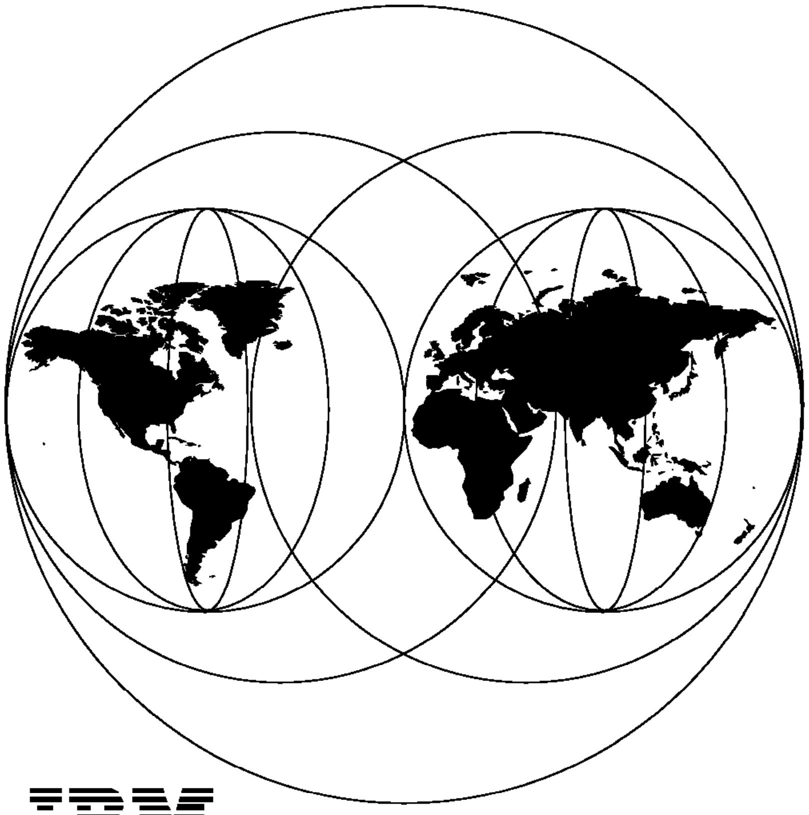
International Technical Support Organization GG24-2563-00

RS/6000 SP System Management: Easy, Lean and Mean

June 1995



International Technical Support Organization
Poughkeepsie Center

International Technical Support Organization

GG24-2563-00

RS/6000 SP System Management: Easy, Lean and Mean

June 1995

Take Note!

Before using this information and the product it supports, be sure to read the general information under "Special Notices" on page xv.

First Edition (June 1995)

This edition applies to PSSP Version 1.2 and AIX Version 3.2.5 for use with the 9076-SP2.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader′s feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 541 Mail Station P099
522 South Road
Poughkeepsie, New York 12601-5400

Abstract

This document is intended for IBM customers, IBM system engineers and third-party professionals concerned with RS/6000 SP system management.

It covers the system-management concepts of the SP, discusses the various hardware and software components involved in the management of the system, illustrates their usage and provides practical experiences.

In addition to hands-on experience with the SP machine, knowledge of parallel processing, AIX and UNIX in general is assumed.

(384 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help IBM customers, IBM system engineers and third-party professionals concerned with SP system management. The information in this publication is not intended as the specification of any programming interfaces that are provided by the PSSP Package. See the PUBLICATIONS section of the IBM Programming Announcement for the PSSP Package 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.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. 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.

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:

ADSTAR	AIX
DB2/6000	ESCON
IBM	InfoExplorer
LoadLeveler	Micro Channel
NetView	Pennant Systems

The following terms reference publicly-available software:

Amd	Berkeley Software Distribution automount daemon
Kerberos IV	Provides authentication for the execution of remote commands
NTP	Network Time Protocol
PVM	Parallel Virtual Machine, available from the Oak Ridge National Laboratory
SUP	Software Update Protocol

Preface

Good system management plays an essential part in the increase of productivity in application development which, in turn, is a major thrust in the objective of enabling customers to load more and more applications on the RS/6000 SP.

The inherently complex nature of large distributed systems, such as the SP, poses some formidable system-management challenges to the customer, however. This is generally given as a reason for deferring, or even rejecting, new application development. In addition, it makes good SP system-management skills almost impossible to find.

The objective of this book is twofold. First, it aims at taking away some of the complexity of SP system management by showing it, at work, in practical situations. Second, it attempts to enhance SP system management by further integrating into it regular RISC/6000 cluster-management tools.

How This Document is Organized

The document is organized as follows:

- Chapter 1, "System-Management Project Environment" describes the environment in which this study was conducted. It highlights the three main types of SP usage in customer installations.
- Chapter 2, "SP System Management Overview" gives a high-level overview of SP system management.
- Chapter 3, "IBM 3270 Host Connection Program" presents the SP in a cross-platform environment. It shows how to connect the SP to a mainframe using the Host Connection Program, the S/370 parallel channel and TCP/IP.
- Chapter 4, "Kerberos" de-mystifies Kerberos, considered a "bete noire" by many SP users. It explains the terminology, the mechanism and the different files essential to managing Kerberos.
- Chapter 5, "The Network Time Protocol (NTP)" describes the Network time Protocol (NTP). It shows the different ways to configure and manage NTP.
- Chapter 6, "The System Data Repository" describes the logical view of the SP. It shows how the SDR works and how the system administrator can interact with it. It discusses the importance of the SDR during SP operations and when it should be backed up.
- Chapter 7, "The High-Performance Switch" describes the HPS hardware, its different components and shows how HPS can be managed to limit faults propagation.
- Chapter 8, "SP Remote Commands" discusses the commands that can be used to access nodes and remotely execute commands in an SP. Sysctl is described extensively. This chapter also introduces the usage of a RISC/6000 cluster-management tool, DSMIT, on the SP. It shows how to set it up and how DSMIT can be used in SP system-management tasks.
- Chapter 9, "Data Management on the SP" is important for users running the Oracle database. It shows how to manage virtual shared disks (VSD) and striped virtual shared disk (HSD). Recoverable virtual shared disk (RCSD) operations are also presented. In addition, it introduces *SYSBACK/6000* as a

convenient system-management tool for backing up the system, user data, and logical volumes (or virtual shared disks) using a remote tape or disk drive.

- Chapter 10, “Job Management (Batch and Interactive)” presents the Resource Manager and the way it reserves SP nodes for exclusive use by a parallel program. Additionally, this chapter shows how to manage groups of SP nodes as a single machine through Interactive Session Support (ISS).
- Chapter 11, “NetView for AIX on the SP” describes the usage of another RISC/6000 cluster-management tool, Netview for AIX, and shows how to implement it on the SP. This chapter also briefly touches the subject of monitoring SP performance with IBM Performance Toolbox/6000.
- Chapter 12, “Print Management” shows how to set up the SP for high-volume printing with a high-speed printer using PSF/6000.
- Chapter 13, “SP User Management” shows how to use the basic components of PSSP, such as File Collections, the Amd automounter and the Network Information System (NIS), to simplify the task of managing a series of tightly-coupled RISC/6000 systems that make up the SP.

In addition, we also provide supplementary information in seven appendixes.

- Appendix A, “Known Command and Script Errors” draws your attention to some known errors that exist in the current release of PSSP. They should be corrected in the next release.
- Appendix B, “New Features For ssp.clients” contains a copy of the README for ssp.clients, informing you of the latest support regarding Kerberos clients.
- Appendix C, “Diagnosing Authentication Problems” discusses different authentication problems when dealing with Kerberos.
- Appendix D, “README Files for Tcl and TclX” contains the README files for Tcl and Tclx, for your convenience.
- Appendix E, “Sysctl Debug Output” presents the debugging environment for sysctl scripts, if you are testing your own.
- Appendix F, “Sysctl and TclX Command Reference” shows the manpages for sysctl and Tclx commands, for your convenience.
- Appendix G, “S/370 Channel Emulator (RPQ 8K1922) Hardware Characteristics” gives specific information regarding the ordering of the S/370 Channel Emulator to support a channel-attached high-speed printer, such as the 3825, when used in conjunction with PSF/6000.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- *IBM 9076 Scalable POWERparallel Systems SP2 Administration Guide*, SH26-2486
- *IBM 9076 Scalable POWERparallel Systems SP2 Installation Guide*, SC23-3865
- *IBM 9076 Scalable POWERparallel Systems SP2 Command and Technical Reference*, SC23-3867
- *AIX Distributed SMIT/6000 Version 1.1 Guide and Reference*, SC23-2561

- *DSMIT V2.1 for AIX, Guide & Reference*, SC23-2667
- *IBM Recoverable Virtual Shared Disk User Guide and Reference*, GC23-3849
- *Oracle 7.13 Parallel Server and Query Option on 9076 SP2*, GC24-4434 (not yet published as of this writing)
- *3270 Host Connection Program Guide and Reference*, SC23-2563

International Technical Support Organization Publications

- *ADSTAR Distributed Storage Manager/6000 on the 9076 SP2*, GG24-4499
- *Using ADSM to Back Up Databases*, GG24-4335
- *AIX Storage Management*, GG24-4484
- *Getting Started with ADSM/6000*, GG24-4421
- *Experiences Using DSMIT in Heterogenous Environment and Visual System Management Under AIX V3.2.5 and V4.1*, GG24-4380
- *IBM LoadLeveler Technical Presentation Update*, SG24-4511 (not yet published as of this writing)

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

To get a catalog of ITSO technical publications (known as "redbooks"), VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order ITSO Redbooks

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their local IBM office.

Customers may order hardcopy ITSO books individually or in customized sets, called GBOFs, which relate to specific functions of interest. IBM employees and customers may also order ITSO books in online format on CD-ROM collections, which contain redbooks on a variety of products.

Acknowledgments

This publication is the result of a residency conducted at the International Technical Support Organization (ITSO), Poughkeepsie Center.

The leader for this project was:

Hung Nguyen	ITSO, Poughkeepsie Center

The advisor for this project was:

Kathleen Schuster	ITSO, Poughkeepsie Center

The authors of this document are:

Peter Kes	IBM The Netherlands

Clive Harris	IBM UK
James Nugen	ITSO, Poughkeepsie Center
Hung Nguyen	ITSO, Poughkeepsie Center

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

Commercial AIX High-End Competence Center, Munich:

Paul Temme

Commercial Parallel Computing
RS/6000 Business Unit, Canada:

John J. Miller

ITSO, Poughkeepsie Center:

Jan Baisden

Pennant Systems:

Bob Peverley

POWERparallel Division:

Jan Badovinatz
Dave Barton
Larry Brenner
Ken Briskey
Michael Browne
Carol Carson
Mike Ferrell
Dave Folsom
Ron Goering
Ted Kirby
Johanna Husta
Kevin Reilly
Skip Russell
John Simpson
Art Wiesen

Solutions Marketing:

Bill Mihaltse

Part 1. Introduction

Chapter 1. System-Management Project Environment

As the need for more computational power grows, the need to manage that power also grows. Nothing is more frustrating for a user than having a new computer with a huge number of Gflops, yet still not being able to get his work done. Now he has the horsepower, but is either not getting adequate information about the system or that information is not straightforward and practical enough to be useful. The intent of this book is to remedy both possible ills as far as the SP is concerned.

Any document on system management that claims to be useful to a user must relate to the conditions under which he uses his system. Toward this end, we have designed the environment for our project based on the results of a survey of the usage types assigned to the SP by customers who already have the machine or are in the process of getting one. Usage is divided into three main areas:

1. Database queries and related disciplines
2. LAN consolidation
3. Unspecified miscellaneous work

Accordingly, our SP configuration incorporates the characteristics of all three of these areas. See below for further details.

Following the description of our environment, we include an overview of system management on the SP and under AIX, in general. Selected topics specific to the SP are further expanded in subsequent chapters.

In addition to the other useful information provided in the appendixes, we wanted to point out a section which is not referenced in other parts of this book. Appendix A, "Known Command and Script Errors" on page 233 describes several minor bugs contained in PSSP commands and scripts, providing work-arounds for each.

1.1 Network Topology

Figure 1 on page 4 shows the environment for this project.

Aside from the SP, the obvious "star" of this project, the environment described in this document includes a network of RISC/6000s and mainframes of interest.

Ring 0 is a token-ring LAN assigned the IP address x.xx.0, hence its name. All of the machines on that LAN can be accessed externally. Two machines of particular interest on ring 0 are:

- The SP used for this project, sp21cw0
- The RISC/6000 that houses the domain name server

The SYSLAB part of our network is in the network domain itsc.pok.ibm.com, whose nameserver is "riscgate." The full name of any machine in the SYSLAB includes the domain name. For example, the full name of the SP is sp21cw0.itsc.pok.ibm.com. However, within the SYSLAB, the short name, sp21cw0, can be used.

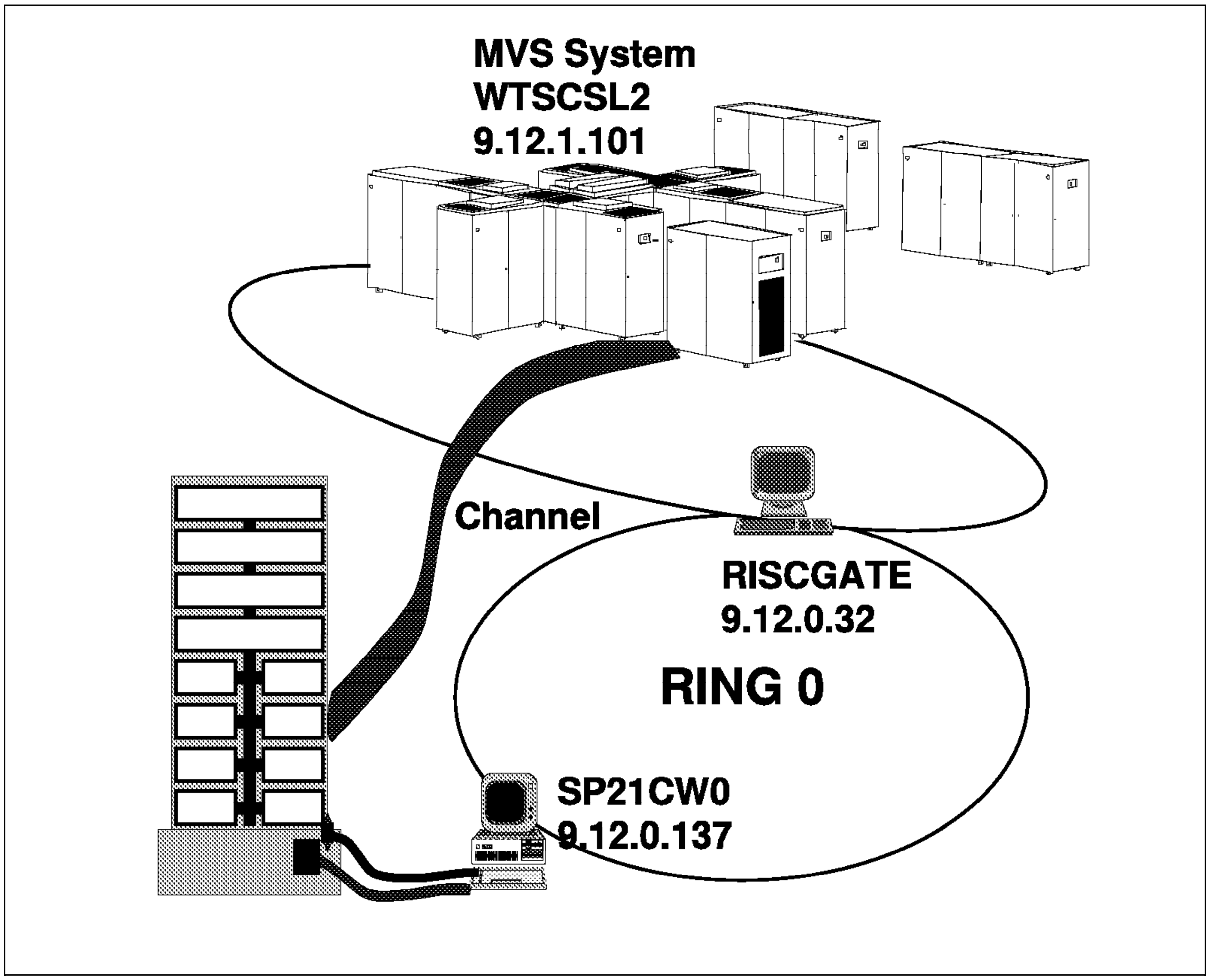


Figure 1. SP System-Management Project Environment

Since, all machines in the SYSLAB use riscgate as the domain nameserver, it is essential that:

- riscgate always be available
- A route exist between riscgate and any defined machine

In addition to running the domain nameserver, riscgate is also, as its name indicates, a gateway. To go from ring 0 to ring 1, riscgate must be specified as the gateway. Machines on rings other than ring 0 are not accessible externally.

This project also focuses on communication with mainframes. Existing documentation concerning this topic is generally deficient. We use an MVS system running on an IBM ES/9021 Model 982. The SP communicates through TCP/IP on an ESCON channel and via token ring (through a 3172 interconnect controller) with the MVS system (called wtscsl2). The LAN on which wtscsl2 is located is ring 1.

1.2 Hardware Characteristics and Addressing

Our SP has eight thin nodes and four wide ones. Following are the characteristics of each thin node:

- 256-MB memory (128 x 2) - no L2 cache
- 1-GB SCSI disk space
- Ethernet
- FDDI
- HPS adapter
- HPSA adapter
- x3270
- S/370 channel emulator

The hardware features of the wide nodes are:

- 256-MB memory (128 x 2) - no L2 cache
- 4/8-GB SCSI disk space
- FDDI
- Ethernet
- HPSA
- HPS
- ESCON control unit

All of the thin nodes are located in the lower part of the SP, making it a 302 model. So, node numbers 1 to 8 are thin nodes. The four wide nodes are all located in the upper part. Their TCP/IP addresses and corresponding interface names are given in Table 1 through Table 4 on page 6.

Table 1. High Performance Switch: css0 - Thin Nodes

Node number	IP address	Interface name
1	9.12.16.38	sp21sw01
2	9.12.16.39	sp21sw02
3	9.12.16.40	sp21sw03
4	9.12.16.41	sp21sw04
5	9.12.16.42	sp21sw05
6	9.12.16.43	sp21sw06
7	9.12.16.44	sp21sw07
8	9.12.16.45	sp21sw08

Table 2. High Performance Switch: css0 - Wide Nodes

Node number	IP address	Interface name
9	9.12.16.46	sp21sw09
11	9.12.16.47	sp21sw10
13	9.12.16.48	sp21sw11
15	9.12.16.49	sp21sw12

Table 3. Ethernet: en0 - Thin Nodes

Node number	IP address	Interface name
1	9.12.30.51	sp21n01
2	9.12.30.52	sp21n02
3	9.12.30.53	sp21n03
4	9.12.30.54	sp21n04
5	9.12.30.55	sp21n05
6	9.12.30.56	sp21n06
7	9.12.30.57	sp21n07
8	9.12.30.58	sp21sn8

Table 4. Ethernet: en0 - Wide Nodes

Node number	IP address	Interface name
9	9.12.30.59	sp21n09
11	9.12.30.60	sp21n10
13	9.12.30.61	sp21n11
15	9.12.30.62	sp21n12

1.3 SP System Structure

To simulate the usage categories we mentioned earlier (and to demonstrate the flexibility of this machine), we have structured the 12 nodes of our SP into three groups, D, S, and T [for Database Query, Servers (LAN Consolidation) and Test and Development (Miscellaneous), respectively].

Note: Since the SP is a shared-nothing computer, resources, except for the HPS, are already architecturally partitioned. Switch partitioning, officially supported in the next release, is not recommended at this time. However, we use it here as a means of separating groups of users to prevent them from interfering with one another. For further information, refer to Chapter 7, "The High-Performance Switch" on page 69.

Figure 2 on page 7 shows the structuring scheme of the SP for this project.

In each of the groups, one node is configured as the boot/install as well as the file-collection server. For group D, it is sp21n10, for group S, sp21n11, and for group T, sp21n12. Software installation and upgrade are done from those servers.

Group D: Group D is the commercial-applications group. We assume a database, perhaps Oracle, is running in all of the nodes of this group. The home directory for users of group D is in node sp2sw10. Our challenge for this project is to install VSD and HSD here and to make recoverable VSD (RVSD) work. All nodes in this group have HPSA adapters. Three 9333 serial disks are used to store the database; they are twin-tailed for RVSD testing.

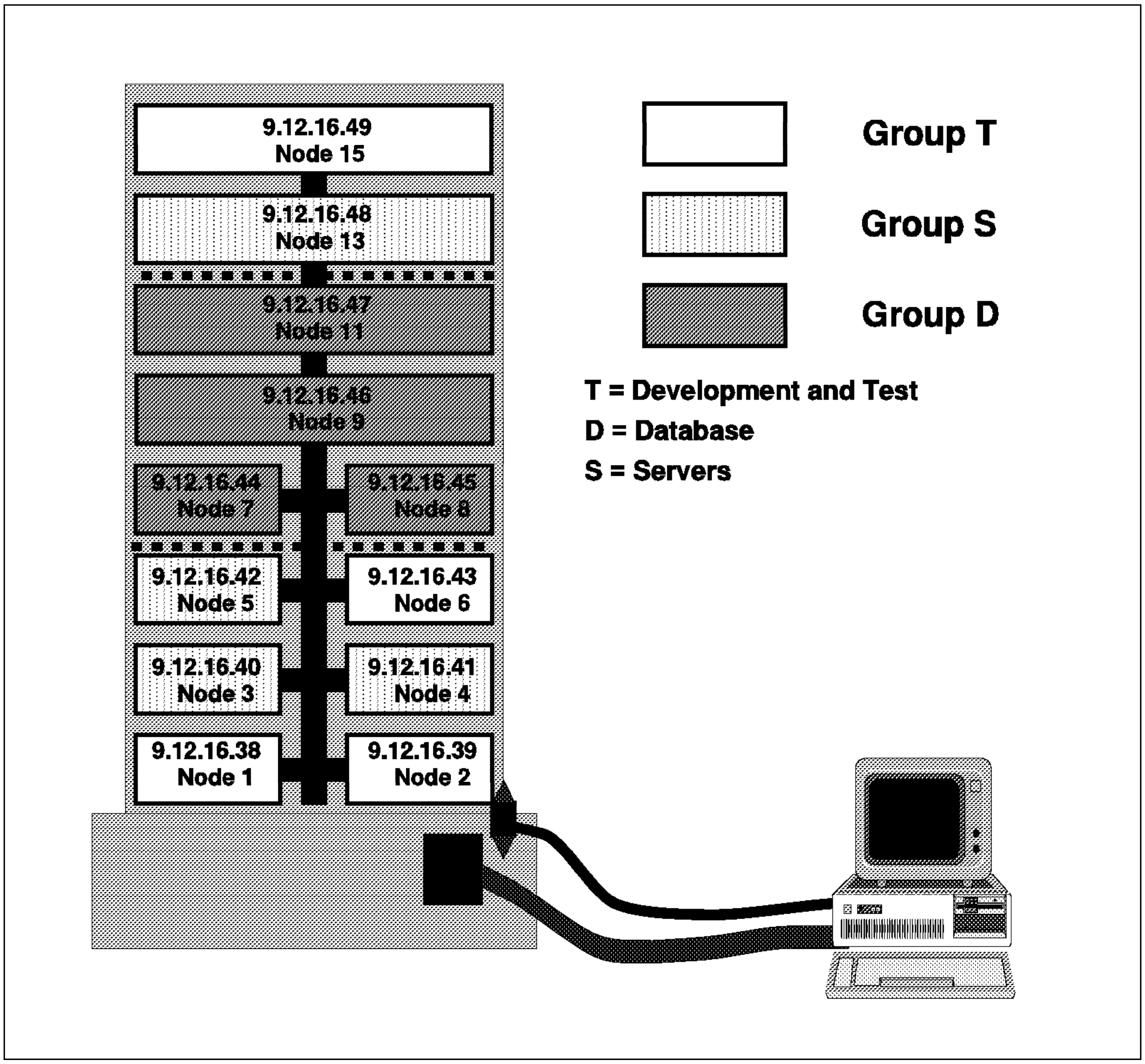


Figure 2. SP Configuration Structure

Group S: Group S is supposed to be a LAN-consolidation type of configuration (that is, all of the servers are consolidated into this group). The home directory for group S is sp21sw11. Our planned work includes:

- Building a nameserver so that ISS can be set up.
- Building an Xstation server so users of group D can use it to access the database.
- Building a print server so all users can print. We use PSF/6000 to drive a 3825 printer, which is normally connected to MVS.
- Installing HCON for host access.
- Installing and configuring NetView/6000 to monitor the SP network.

In the course of setting up servers, we must use Kerberos for authentication.

Group T: This group includes nodes sp21n1, sp21n6, sp21n10 and sp21n12. One of our tasks is to install LoadLever′s ISS here, creating a single IP address to make life easier for the users in that group. The home directory for users of group T is in node sp2sw12. However, if a user logs into any of the nodes within the group, his home directory is the same. In other words, a user logging on node sp21n06 has his home directory mounted for his use from sp21n12. A user in group T cannot log into any node other than those in group T.

The redbook also includes our experiences with using Amd, File Collections, Kerberos, NIS, NTP, SDR, SYSBACK/6000, sysctl and other assorted topics.

Chapter 2. SP System Management Overview

The SP provides a comprehensive suite of management applications that allow you to manage, operate and administer the SP. IBM′s strategy to deliver system-management function is three-pronged:

1. The AIX Parallel System Support Program (PSSP) is supplied as part of the SP hardware. The PSSP includes function required to manage, monitor and operate the SP system.
2. The use of standard hardware and software in the design of the SP system leverages a large number of system and application software products from many areas within IBM [ADSTAR Distributed Storage Manager (ADSM), NetView and Print Services Facility (PSF) to name a few].
3. An aggressive program is in place to facilitate and encourage independent software vendors to enable and port their applications to the SP System. In the areas of system management, these products can provide SP-enabled versions of popular management applications or fill voids in the management function for the system. Among the vendors who have (or will have) system-management products enabled for the SP are Aston Brooke, BGS, CA, Candle, Epoch, Gejac, Legato, STK and Tivoli.

The SP system-management offerings share the following attributes:

- Single Point of Control -- Administrators are able to manage and operate a node or groups of nodes from the Control Workstation using a command-line or a graphical user interface. Remote operations capabilities are provided by accessing the Control Workstation.
- Flexibility -- Flexible mechanisms are provided, allowing you to easily integrate the system into your existing UNIX environment. You can choose those system-management tools that provide value in your environment. If you have management solutions in place, the SP is designed to be integrated into those solutions. A single, consistent environment for users is provided regardless of which node they use; a user′s environment and home directory may be the same on any node in the system.
- Scalability -- System-management solutions for the SP scale in a number of ways. Managing large numbers of nodes requires scalable distributed tools and ways of managing groups of nodes from the single point of control. Large amounts of data require hierarchical storage-management solutions and comprehensive and efficient backup and archival. Large numbers of users and jobs require tools for efficient user management, and workload balancing.
- Openness and Connectivity -- SP systems must participate in and be integrated into a larger computer environment consisting of mainframes, heterogeneous workstations or both. By adhering to open-system standards and providing hardware and software for connectivity, IBM ensures that the SP can be integrated into an enterprise.

System management is not a single, monolithic application. Our view of system management is shown in Figure 3 on page 10.

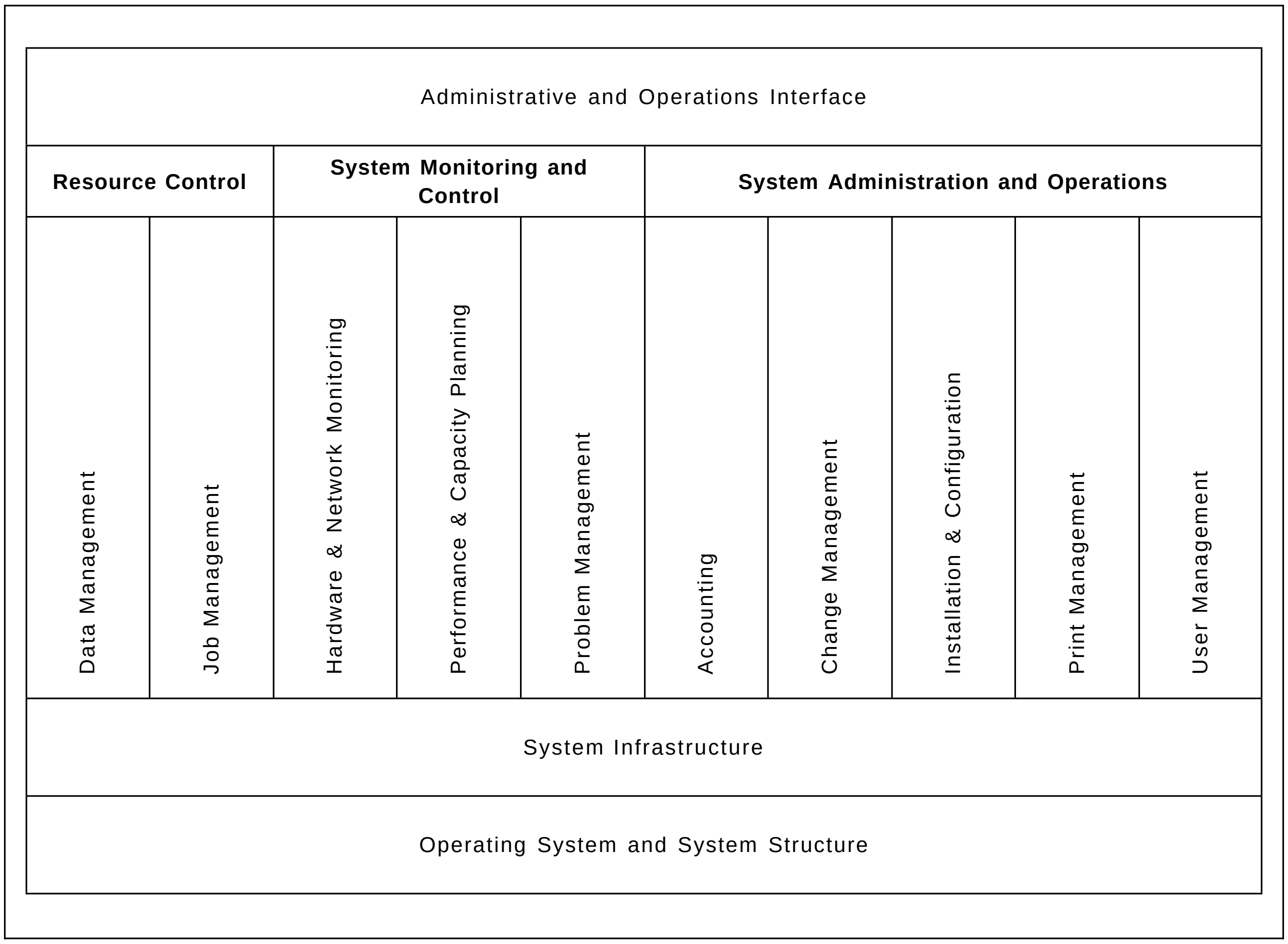

Administrative and Operations Interface
Resource Control
System Monitoring and Control
System Administration and Operations
Data Management
Job Management
Hardware & Network Monitoring
Performance & Capacity Planning
Problem Management
Accounting
Change Management
Installation & Configuration
Print Management
User Management
System Infrastructure
Operating System and System Structure

Figure 3. SP System-Management Structure

The Operating System and System Infrastructure provide the base for a range of system-management applications grouped under three broad disciplines: System Administration and Operations, System Monitoring and Control, and Resource Control. Beginning with discussions of the AIX Operating System and its Infrastructure and the Administrative and Operations Interface, the SP offerings and capabilities in each of the three disciplines are described in the next several sections. We follow the same format throughout the remainder of the book, dedicating a part to the Operating System and System Infrastructure, one to the Administrative and Operations Interface and one each to the three super-disciplines.

2.1 Operating System and System Infrastructure

This section provides a brief overview of the AIX operating system and the system-management function provided by SDR, the security and RAS infrastructures, HACMP, system log files and NTP.

2.1.1 AIX

Each node in the SP system has AIX resident on its local disk (a minimum 1GB disk is required in each node). Table 5 summarizes the node′s filesystem topology.

Table 5. SP Node File-System Topology

Object	Shared	Location
/ (root)	no	local
/usr	optional•	local or server
/var	no	local
/tmp	no	local
boot image	no	local
Paging LV	no	local
Dump LV	no	local
/u	yes	mounted from anywhere in network•

•/usr can be local or can be shared from a server using AIX 3.2.5 /usr client technology.

•Amd (the BSD automount daemon) is optionally used to mount users′ home directories from any NFS server in the network.

The basic AIX filesystems (root, var, tmp) are local to each node. The /usr filesystem can be local on each node or can be mounted from a /usr server using the AIX /usr server technology. Using one or more /usr servers can save significant space on local disks in environments where local access to the /usr files is not necessary (this capability is not available in AIX 4.1).

User home directories and application data can be made available through Network File System (NFS) from anywhere in the network or by running a shared filesystem such as Andrew File System (AFS) or Distributed File System (DFS).

The SP System runs AIX 3.2.5, IBM′s version of the UNIX operating system. AIX is a recognized industry leader in function and performance, providing a number of capabilities for system management that can be exploited on the SP system including:

- Logical Volume Manager (LVM), which improves data-management productivity, enables files to span multiple disk drives and, through disk mirroring, provides a high-availability option for critical data.
- System Management Interface Tool (SMIT), which provides a single, consistent and expandable interface to system-management commands.
- System Resource Controller (SRC), which simplifies the management of subsystems and daemons.
- Security features including a shadow password file and comprehensive auditing facilities.
- AIX device support, allowing devices to be added and deleted at any time without system disruption.

AIX 4.1 will be available for the SP system in 1995.

2.1.2 System Data Repository

The System Data Repository (SDR) is a centralized common-data repository that provides configuration data storage and retrieval from the Control Workstation and all nodes for persistent system data. System and node configuration data, job data and other system data, such as site environment data, is stored in the SDR. Command-line interfaces are provided for the SDR. Typically, data in the SDR is accessed through management applications.

2.1.3 Security Services

PSSP release 1.2 includes a security infrastructure, which provides authentication services for users and servers. Authentication is the foundation for providing other security features such as accountability, access controls and least privilege. The authentication services provided in the SP System are based on MIT-Kerberos version 4. Systems running AFS or an existing Kerberos-version 4 server can integrate the SP System into their existing Kerberos realm.

Kerberos functions as a third party to authenticate the identities of clients and servers. Utilizing Kerberos authentication prevents unauthorized access to system resources. Access Control Lists (ACLs) are used to control access to resources in the SP System. The servers for hardware monitoring and system control use authentication and ACLs to control access to their services. Secure versions of the remote commands (`rsh`, `rcp`) have been provided, eliminating the need for insecure .rhost files.

Because, the use of Kerberos requires changes in existing command libraries, it is expected that a generalized interface, such as the the Generalized Security Services Application Programming Interface (GSS-API), will be exploited in the future. This will allow any authentication server that supports this API to be used, including Kerberos Version 5, DCE Kerberos and AFS Kerberos, among others. A comprehensive system of ACL management should then be provided.

2.1.4 RAS Infrastructure

The objective of the RAS (reliability, availability and serviceability) services is to provide the infrastructure needed to eliminate single points of failure in the system and to enable prompt, efficient detection, diagnosis and repair of hardware and software errors. SP provides a scalable, efficient heartbeat for recognizing node failures. Failing nodes are highlighted in the SP system-monitor GUI. Base AIX provides disk mirroring capability for high-availability access to critical data.

2.1.5 HACMP

AIX High-Availability Cluster Multi-Processing/6000 (HACMP/6000) Version 3.1.1 is a software subsystem that provides high-availability function on the SP platform. It allows up to eight POWERparallel SP nodes to be configured in a highly-available cluster. If one processor node fails, its functions can be assumed by another processor node within the cluster. The standby processor does not sit idle; it can be doing other work until such time as it is needed for backup. Another processor node in the SP highly-available cluster can be designated to back up the standby node.

In addition to existing support when node failure occurs using Ethernet, token-ring or FDDI network adapters, HACMP provides additional support for node failure when using the SP High-Performance Switch.

In a concurrent-access configuration, HACMP/6000 provides the ability to spread workload across multiple SP nodes, sharing the disk and/or CPU resources of the nodes. This clustered approach, together with the capability of applications fallover and recovery/restart of the HACMP-configured machine, provide additional levels of high-availability processing for user-critical applications.

2.1.6 System Logging

Since physical console devices are not attached to each of the nodes, many of the SP subsystems redirect console output to a log file. Rather than being consolidated on the Control Workstation, each of these logs remains on the individual nodes. This scales better, allowing users to examine these logs only when a problem is detected. The logs, collected in a set of directories in /var/adm/ssp, are managed through a nightly cron job that trims large logs and removes old ones.

Software that provides a comprehensive system for managing and viewing the nodes′ error logs and other log files would be a useful addition to the existing collection of PSSP tools.

2.1.7 NTP

Because the SP nodes do not have system batteries, SP uses the Network Time Protocol (NTP) both to set the time at system initialization and to synchronize time-of-day clocks on the Control Workstation and processor nodes. If a site is already using NTP, the SP can be configured to use the existing time server. Or, if NTP is not being used elsewhere on the site, the SP can be configured to keep a consistent time among its nodes. Alternate time-keeping systems, such as TCP/IP′s time server [Domain Time Server (DTS)], may be used as well.

2.2 Administrative and Operations Interface

This section briefly discusses the system-management function provided through the Administrative and Operations Interface.

2.2.1 Centralized Management Interface (CMI)

The SP provides a SMIT-based interface for managing, and a collection of graphical interfaces for operating, the SP system. The SP provides a SMIT-based interface for managing and operating the SP system. All actions available through SMIT also have command-line equivalents. The CMI provides the capability to:

- Enter, change and show system and site configuration data
- Configure node installation parameters (net-install image name, enable reinstall)
- Configure boot/install servers or /usr servers
- Configure the Control Workstation
- Manage the High Performance Switch
- List node hardware (`lscfg`) data

- List node network data (netstat) or node filesystem data (df)
- Manage user accounts

Many of the SP system-management tools, such as user management, print management, accounting, installation-server topology and NTP, can be configured through the CMI. A set of default configuration values is provided, but these tools can be reconfigured through the CMI at any time. Necessary changes are made without reinstalling the system.

2.2.2 System Command Execution

A new command, dsh (distributed shell), is provided to execute remote commands on all nodes or groups of nodes in the SP system. A number of flexible ways to specify these nodes and groups of nodes is provided. The dsh command is based on rsh, hence requiring rsh privileges. With dsh, the user can execute all commands in parallel, with output returning to a single destination. To control the use of system resources, the degree of parallel execution can be specified. As a default, dsh will only operate on those nodes that are "alive," as indicated by the "host responds" indicator. Although this is a relatively simple tool, we have seen many benefits from its use. For example the command `dsh -av ps -fu johndoe` could be used to report all processes belonging to user johndoe on all running nodes.

If available, dsh will use the authenticated rsh.

2.2.3 Sysctl

Sysctl is an authenticated client/server system for running commands remotely and in parallel. It provides:

- Least-privilege capability -- Root authority can be dynamically delegated to non-root users based on their authenticated identities, the task they are trying to perform, access control lists and any other relevant criteria. The root password need not be given out to as many people, thus keeping it more secure.
- Distributed execution -- Sysctl applications can be executed on remote hosts with full authentication and authorization. Sysctl provides a secure, easy-to-program remote-command execution mechanism for arbitrary AIX commands, scripts and programs.
- Parallel execution -- Sysctl applications can be efficiently executed in parallel on many hosts.
- Programmability -- Customized sysctl applications can be coded as shell, Tcl or Perl scripts.

The SP system-administration software provides a set of commands that exploit the sysctl facility. See 2.2.4, "Parallel Management Commands" on page 15. You may also want to design your own sysctl applications. Candidates for such applications are tasks that require root authority, but that administrators would like to delegate (for example, user management, backups, filesystem administration and so on).

2.2.4 Parallel Management Commands

The SP System Administration software provides a set of commands built on dsh and sysctl that are useful for performing common tasks in parallel on multiple nodes. A summary of these commands is shown in Table 6. The current version of these tools has been influenced by an ongoing Parallel Tools Consortium project (led by researchers at Argonne National Laboratory) dealing with scalable UNIX commands. The Parallel Tools Consortium is an industry group (composed of vendors and users) that is exploring portable, usable tools for today′s production parallel systems. The target nodes for these commands can be specified in a variety of ways using the hostlist command. Output from these commands can optionally be filtered to remove duplicate lines, greatly reducing the volume of output.

Table 6. Parallel Management Commands

Command	**Function**
pexec	Execute a command in parallel on specified hosts
pexscr	Execute different commands in parallel on different hosts
pcp	Copy local file or directory to remote hosts in parallel
p_cat	Issue cat command in parallel on specified hosts
pls, pfind. prm, pps, pmv	Issue the ls, find, rm, ps, mv commands in parallel on specified hosts
ppred	Issue a ksh test on each node in parallel and run commands based on the results of that test
pfps	(parallel process find). Takes arguments similar to the find commands, but operates on processes.
pdf	Provides filesystem information from hosts in parallel

2.3 Resource Control

The super-discipline Resource Control encompasses the areas of Data Management and Job Management, described in further detail below.

2.3.1 Data Management

Data Management is a complex topic. Our discussion of data management is broken into two sections. First we define the various functions of managing data, specifying in which environments those functions are important. Second, we discuss the two aspects of SP data management strategy: high-performance data movement and enabling data-management software (with focus on ADSM).

2.3.1.1 Data-Management Functions

We have developed a characterization of types of data and management functions for that data to help categorize solutions for data management. Definitions of terms for data management are shown in Table 7 on page 16.

Table 7. Data-Management Definitions

Term	Description
Online	A file is online when it can be accessed by opening the file. The most common type of online file is one that resides with its data in a filesystem that is mounted on the computing system. In UNIX systems, there is exactly one data tree which represents the online data; this is the tree that starts at the root directory.
Offline	A file is offline when it can be retrieved from some storage system by an explicit user command and placed online. Retrieval techniques include ftp or storage-system-specific commands. Some offline storage systems use NFS as a means of allowing access to offline data. This blurs the distinction between online and offline. We treat storage systems that allow access via NFS as offline if data is typically copied from the storage system for use. Using this definition, UniTree is a server of offline data. There may be multiple offline data trees accessible to a system.
Hierarchy	A storage hierarchy allows data to be moved (transparent to the access interface) among storage media of differing speeds within a class of data. For example, data may move among fast disk, slow disk and tape according to the access pattern observed by the system. The system attempts to keep the data which is most likely to be accessed on the highest-performance medium. There are two types of hierarchy: • A hierarchy whose highest level is a filesystem mounted into the data tree anchored in the root file system, typically called hierarchical storage manager (HSM) or space management • A hierarchy that has its anchor in a backup or archival system
Migration	Migration is the act of moving data from a faster to a slower level in a hierarchy.
Recall	Recall is the act of moving data from a slower to a faster level in a hierarchy.
Tape Management	A tape-management system is not strictly a storage-management system, since it does not manage data objects. Rather, a tape-management system manages removable media, including allocation of tapes and drives, creation of logical tapes that are larger than a physical tape and management of tape libraries. The unit of management is the tape, not the data object contained on the tape. Although a tape-management system may be used in conjunction with other functions to do backups or archives, it is not a backup/archive system.

There are a number of functions that may be important for data management, including:

- Backup: Data objects are backed up to allow recovery from certain events which destroy the primary copy of the data object. These events include failure of the media storing the object and human error. A backup makes a copy of the object at a point in time and places that copy in offline storage. Typically, there is a current backup copy that reflects the latest version of the data and some number of previous backup copies. The previous backup copies are retained according to site policy and overlaid when obsolete. The unit of data is typically the file, but backup systems are also conscious of larger units of failure (a physical disk, for example). Data in a backup file is typically written once, never read and retained until replaced or removed.

- Archive: This is a real-time copy of data into an offline repository. It may or may not be accompanied by deletion of the online copy. Archive is done for different reasons than backup. An archive copy may be made to conserve disk space by relocating specific data to cheaper storage. This is a common usage in the technical-computing community for large objects which need to be online only during processing. Archive copies may also be made to satisfy operational requirements, such as the monthly archival of certain classes of data for record-keeping purposes. Archive data is named differently than online data and is managed according to a separate set of rules. Typically, archive data is retained until an expiration data regardless of how many times the file is archived. Data in an archive is typically written once, read several times and has a long lifetime.

- Space Management: This is the capability of extending, transparent to an application, the online storage capacity through the use of a hierarchy of devices. The image presented to the application is that of disk space beyond that which is installed on the system. This image is achieved by keeping all information about a file online at all times, but keeping the data contained in the file online only if it is perceived to be likely to be referenced soon. A file is created on high-speed disk. It may be migrated to some slower device if its reference pattern suggests that it may not be needed soon and the disk space is required for some other file. The file is recalled from the slower device when referenced by an application.
- Offline Hierarchy: This is really a feature of archive, backup or space management. For economic reasons, an offline data system may consist of several levels of devices. Data enters at the top level and ages towards lower performance-devices. From the user perspective, the interface to the data is the same except for performance.
- Availability: One of the requirements on a computing system is the availability of paths to the data and the data itself. Techniques exist for maintaining access to the data across certain types of access-paths failure.

2.3.1.2 SP Data Management

Data-management functions are among the largest users of the communications links between the nodes. Data is moved among nodes using the fastest communications available while complying with standards. Work is underway to improve the performance of these communications methods.

The second prong of the SP data-management strategy is to enable data-management applications on the SP. With the growth of the SP, data-management solutions have been required for a number of differing types of data and for application use of this data. These uses have been roughly broken into four categories labeled RDB, Huge, Not-so-large and Traditional UNIX.

The RDB category consists of data that is managed by one of a number of relational-database systems, such as IBM′s DB2/6000 Parallel Edition. These systems allow the execution of decision-support, transaction-processing and data-warehousing functions on the SP. The major data-management issues here are the requirements for rapid backup and restore of the relational data. In some cases, archive of a level of the database may also be of interest. Most of the database products either provide or will provide a backup/archive utility. In some cases, third parties also provide this function. The target of these utilities is typically a data-management product, such as IBM′s ADSM. A data-management product adds the capability to manage the backups, track what is backed up and manage tape libraries and other automated hardware. Over time, these data-management products will be enhanced to provide additional disaster-recovery functions and space management.

The Huge category consists of data that is processed by parallel applications and resides in a mass-storage system. Applications using this class of data are the Grand Challenge applications of Astronomy, Biology, Chemistry and Geology and other scientific disciplines. The petroleum industry also uses this class of data. The files are extremely large and the computation performed is complex. The economics of these applications dictates that the data be kept on the lowest-cost media available when not actively being used. Thus, the data is typically kept in tape libraries and recalled to online filesystems when actually needed. The management of these tapes can be relatively complex, and there are three classes of products to aid the user. The tape-management systems, such as ReelLibrarian, allow the user to access tapes from any node of an SP system, to create tape files and to designate that these files span physical tape cartridges. The commercial backup/archive products, such as IBM′s ADSM, free the user from worrying about the specifics of tapes when archiving or recalling data. This class of product provides facilities to manage space on available media, to maintain multiple copies and to manage the performance of the retrieval of data. Future enhancements include transparent retrieval of the data from slower storage upon access. The third class includes specialized high-end technical-computing support products such as UniTree and HPSS. These programs provide storage and retrieval of huge amounts of data and are aimed at the very high end of the technical-computing world.

The Not-so-large category of data deals with large sequential files that reside online. This data encompasses many of the emerging application sets in the parallel-processing world. These include such things as CAD/CAM, image processing and digital libraries. The major requirement here is the backup of data in an effective way within narrow backup windows. The primary solution in this area is commercial UNIX backup/archive products such as ADSM. Such products provide for the automatic backup of data throughout the SP (and beyond) and the safekeeping of these backup copies on media attached to the SP or some alternate backup server. They provide management algorithms for features such as the number of backups to keep, the expiration of data after an administrator-specified period of time and control of which data gets placed on faster media for rapid restore. ADSM exploits the high-speed communications available between SP nodes to provide effective service in this area.

The Traditional-UNIX category consists of the data that would be typically kept for an individual user. Because of the difficulty of managing data on distributed workstations and servers, much of this data is being recentralized onto systems such as the SP. This data requires backup much as the Not-so-large category, but consists of much larger numbers of smaller files. Backup facilities, such as ADSM, are designed specifically for this type of data and perform very well on it. Again, ADSM exploits the high-speed inter-node communications of the SP for backup/retrieval of this data.

2.3.1.3 ADSM

ADSM is a highly-functional storage-management product from the ADSTAR division of IBM. ADSM provides hierarchical file management, backup and restore, as well as archival capabilities, migration, automation, space management and central administration.

SP nodes can participate as ADSM servers for other SP nodes, other computing systems or as ADSM clients for ADSM servers elsewhere in the computing environment.

Additional information on ADSM can be found in the ADSM product publications as well as a series of redbooks on ADSM. Particular attention is called to the publications *ADSM on the SP* and *Backing up Databases Using ADSM*.

2.3.2 Job Management (Batch and Interactive)

Job Management is concerned with the initiation, delivery, management and monitoring of work on the SP system. Many mechanisms are available to accomplish this.

2.3.2.1 LoadLeveler

The LoadLeveler program product is bundled with the PSSP in the US and Europe. It provides facilities for building, submitting and processing both serial and parallel jobs on SP systems and workstation clusters (including RISC System/6000, Sun, SGI and HP). Through LoadLeveler, users can exploit the resources available in a cluster of workstations or SP nodes. LoadLeveler can increase job throughput by taking advantage of unused cycles and balancing the workload among scheduled systems.

LoadLeveler uses information on machine availability and current workload, along with job requirements, such as memory size, disk space and software features, to determine which batch jobs can be dispatched to a particular machine.

Hardware features and availability information are recorded in configuration files. General information that sets policy for the entire LoadLeveler pool is defined in global configuration and administration files.

For parallel jobs, LoadLeveler interfaces with parallel-programming software, AIX Parallel Environment, AIX PVMe or AIX PVM 3.3 to obtain the multiple SP nodes or workstation processors required for the job′s parallel tasks. LoadLeveler maintains the status of the parallel job and reports back to the submitter.

LoadLeveler can also route logins (or other TCP/IP-based application requests) to the best available node using user-specified criteria.

Work is continuing on LoadLeveler in the areas of scheduling, security and authentication, and scalability.

2.3.2.2 Resource Manager

The Resource Manager (RM) is a server that runs on one of the SP nodes. It manages nodes for parallel use by mediating shared or exclusive use of nodes or High Performance Switch adapters between various requesters (such as LoadLeveler or the Parallel Operating Environment component of the Parallel Programming Environment). Nodes are divided by the administrator into pools for shared and dedicated usage (interactive, serial and parallel). The Resource Manager ensures that dedicated nodes are only allocated to a single requestor. It is also responsible for creating authentication tables for the Communication Subsystem.

2.3.2.3 Job Scheduler for AIX

IBM Job Scheduler for AIX manages the complexity of scheduling, initiating and monitoring the regular background workload in distributed RISC-System/6000 and SP environments. Job Scheduler initiates work following policies and rules specified by the systems administrator and monitors the workload through completion. It automatically detects jobs that have failed, can restart a failed job and, optionally, take user-defined corrective action prior to restart. Job Scheduler produces daily and weekly workload-execution plans and various views of historical information on actual processing. In addition, Job Scheduler′s job and execution logs provide an audit trail of processing. (An Oracle or DB2/6000 relational database stores the schedule, execution rules and job-completion data).

Job Scheduler provides a user interface that allows individual users to monitor and control their own jobs based on the level of access given to them by the system administrator. This same mechanism allows the system administrator to perform functions for all Job Scheduler jobs.

Table 8 provides a comparison of LoadLeveler and Job Scheduler function.

Table 8. LoadLeveler - Job Scheduler Comparison

Function	LoadLeveler	Job Scheduler for AIX
Job Arrival	Non-predictive	Predictive
Job Environment	Jobs run once resource requirements are available.	Multiple production jobs run cyclically at predetermined times and events.
Job Submission	As needed by users	According to established schedule
GUI	Lists of Jobs and Machines	Schedule on time line
Schedules	Serial jobs, parallel jobs and interactive sessions	Serial jobs
Agents	Heterogenous UNIX network	AIX/6000 processor nodes
Database	Not required	Required

2.4 System Monitoring and Control

System Monitoring and Control consists of the Hardware and Network Monitoring, Performance and Capacity Planning and Problem Management subdisciplines, each described briefly below.

2.4.1 Hardware and Network Monitoring

The SP provides a number of facilities to aid in system operation.

2.4.1.1 SP System Monitor

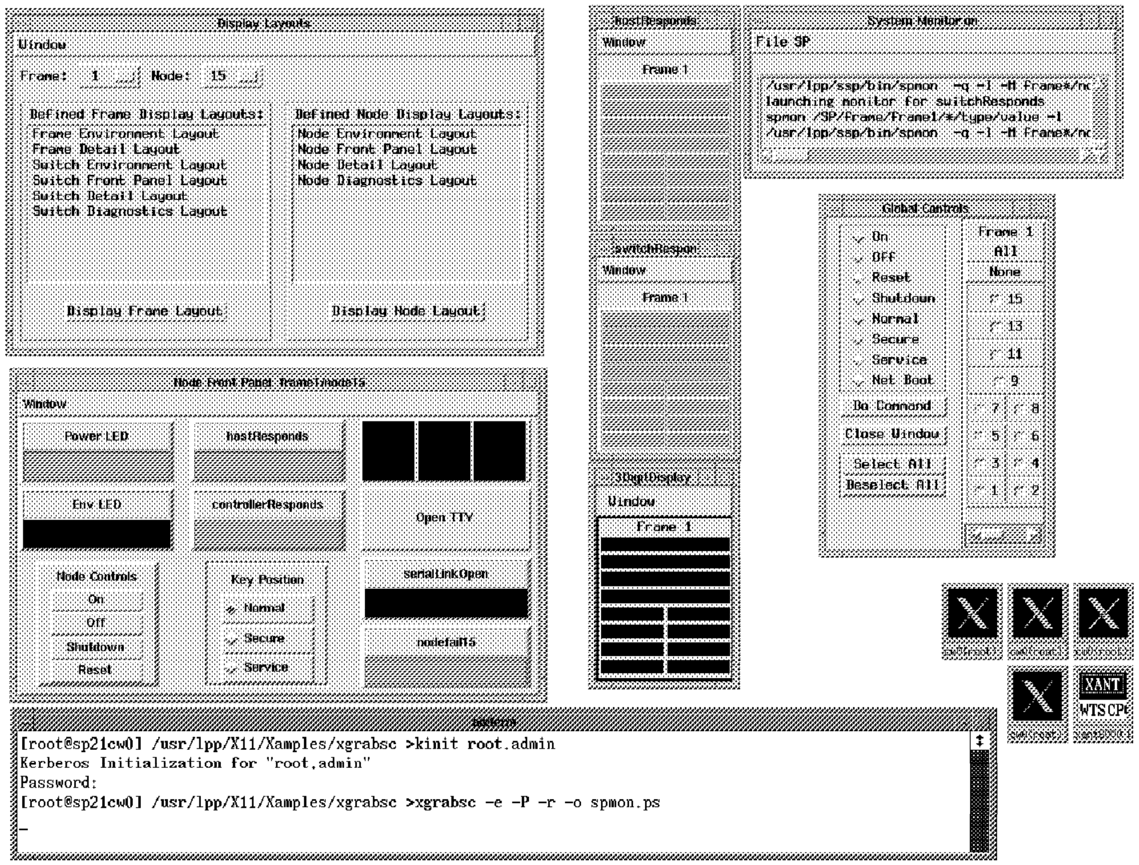
The SP System Monitor (spmon) allows the operator and system administrator to monitor and control frame and node hardware using a command-line interface or a Motif-based GUI. Nodes and frames can be powered on and off; the key-switch position and reset button of the node can be controlled. Hardware data such as node LED state, frame power, temperature and voltage can be monitored. Failure data is logged in the AIX errorlog and syslog.

Note: The AIX errorlog and syslog contain similar information. You may turn off syslog to save storage. This, however, is *not* recommended because information for public domain code goes only to syslog. Also, complementary information can be put in syslog with a pointer to it in the errorlog.

A range of System-Monitor GUI panels are provided with which to oversee system function. Figure 4 on page 21 shows just a few of them:

- A Display Layouts Window
- A Node Front-panel Display

- The Hostresponds, Switchresponds and 3DigitDisplay
- System Monitor Main Window
- The Global Control Window



```
/usr/lpp/ssp/bin/spmon  -q -l -M frame*/nc
launching monitor for switchResponds
spmon /SP/frame/frame1/*/type/value -l
/usr/lpp/ssp/bin/spmon  -q -l -M frame*/nc
```

```
[root@sp21cw0] /usr/lpp/X11/Xamples/xgrabsc >kinit root.admin
Kerberos Initialization for "root.admin"
Password:
[root@sp21cw0] /usr/lpp/X11/Xamples/xgrabsc >xgrabsc -e -P -r -o spmon.ps
_
```

Figure 4. The System Monitor

The hardware-monitoring function has been rewritten in Release 2 of the PSSP to provide greatly-improved performance and reliability.

2.4.1.2 System Startup/Shutdown

In a complex system such as the SP, there are dependencies between nodes such that clients should not be started until server nodes are available, and servers should not be shut down while clients are active. The system shutdown and startup commands allow the administrator to specify these dependencies in a sequencing file. Functions to start up and shut down the entire system or groups of nodes are available from the command-line or the System-Monitor GUI.

2.4.1.3 Enterprise Network Monitoring

Existing AIX/6000 network management tools, NetView for AIX and Systems Monitor for AIX, can be utilized to integrate the SP into an existing enterprise network.

NetView for AIX is a comprehensive management solution that facilitates network management of a multi-vendor TCP/IP network by managing all TCP/IP-addressable SNMP devices, including workstations, PCs, mainframes, routers, bridges, hubs and SP nodes. The NetView for AIX manager polls the base AIX SNMP daemon agents to gather information for display and action by a network control desk.

NetView for AIX performs the functions of:

- Configuration management for automatic discovery of the network (that is, creation and maintenance of topological network maps)
- Performance management for monitoring network statistics and displaying critical network resource status and statistical summaries for analysis and corrective actions
- Fault management for verifying the integrity of the network (using thresholding and filtering algorithms for easier alert notification) and for defining and implementing corrective actions to SNMP traps.

The NetView management software can be installed on any SP node or on a workstation external to the SP. (**Note:** Due to the quantities of polling traffic that can be generated, do not install the NetView manager on the Control Workstation).

NetView for AIX is the basis for a number (hundreds) of NetView applications. Two IBM-provided NetView products that are particularly applicable to the SP environment are Systems Monitor for AIX and Trouble Ticket/6000 (see 2.4.3, "Problem Management" on page 23 for further details on Trouble Ticket).

Systems Monitor for AIX (not to be confused with the SP System Monitor) is a distributed SNMP manager running on heterogeneous UNIX workstations (AIX, HP-UX, Sun Solaris and NCR UNIX versions are available). It performs local SNMP management by monitoring and managing a locally-defined set of SNMP devices, receiving traps of threshold-exceeded conditions on these devices. Intermediate SNMP management is accomplished by off-loading SNMP management tasks from the network manager and forwarding only the most critical information. Systems Monitor also allows for easy supervision of key applications or subsystems.

Systems Monitor for AIX can be installed on any host designated as responsible for monitoring a subset of the enterprise network. This subset may be a single SP, a collection of SP nodes or any other host(s).

2.4.2 Performance and Capacity Planning

Performance Toolbox/6000 (PTX/6000) and the related Performance Aide/6000 (PAIDE/6000) are two products currently supported on the SP to perform performance monitoring at the system level. PAIDE/6000 is a client application gathering AIX performance data on each node. It can perform data reduction using data filtering, thresholding and alert-processing techniques. PAIDE/6000 sends the results to PTX/6000 for consolidated analysis and presentation. PTX/6000 includes easily-customized, real-time color graphic monitors for presentation of performance statistics. This consolidated data may also be passed to AIX character-based performance tools.

2.4.3 Problem Management

In performing Problem Management, the SP System Monitor collects hardware error data in a node′s AIX syslog and error log for alerts and corrective action. This data may be consolidated with other log data from the SP nodes. An operator can monitor this data by scanning the AIX syslog file or using the command `errpt`. This information can also be used by the IBM customer engineer for hardware diagnosis and repair. In addition, the log data may be fed into an enterprise problem- or network-management system, such as Trouble Ticket for AIX, for processing and action.

Trouble Ticket operates alone or in conjunction with the NetView for AIX product to help network personnel manage problems that arise in a distributed network environment. Trouble Ticket detects events and alarms generated by NetView and automatically records incident reports for each condition that matches filtering criteria. These incident reports are logically grouped into “trouble tickets,” which can be tracked and managed to resolution. The network administrator can also create trouble tickets for user-created incident reports.

Trouble Ticket for AIX also allows for automatic notification of service personnel, consolidation of related reports and identification of a resolution strategy. Summary analysis and exception reports are created to manage the problem-resolution process.

2.5 System Administration and Operations

System Administration and Operations consists of the Accounting, Change Mangement, Installation and Configuration, Print Management and User Management subdisciplines, each described briefly below.

2.5.1 Accounting

The SP support for accounting builds upon standard AIX accounting function, providing three main capabilities:

- Accounting-record consolidation

  SP accounting support builds upon standard UNIX System V accounting. 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 can be any node). Nodes are configured into groups called classes. All of the data from a given class is consolidated. Classes can be used to impose different charges for nodes with differing capabilities.

  The administrator uses the CMI or commands to specify the accounting configuration. The SP system-management code automatically configures accounting on the selected nodes. The command `nrunacct` (a modified version of the AIX `runacct`) 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 cluster master, which performs additional processing. The output of this processing is data, in standard format, that can be used for charge-back purposes, usage monitoring or capacity planning, and can be fed into existing accounting applications that you may have today.

- Parallel job accounting

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

- 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 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 that fall 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.

2.5.2 Change Management

The SP software has tools for managing different aspects of change on the system. Change-management activities include the application and testing of new software levels, keeping files and directories updated and altering the configuration of the SP subsystems.

Maintenance can be applied directly to nodes using dsh and standard AIX commands, but for any extensive service, we recommend that a new network install image be created and that the nodes be reinstalled. Data can be preserved during this process by keeping the data in separate volume groups.

New levels of application and system software can be tested by loading them on a test node or group of nodes that has been removed from the switch. When you are satisfied with the test environment, it can be propagated to the rest of the nodes in the system.

File collections are another useful tool for change management. They provide a means of managing files and directories across many systems. For further details, see the discussion on file collections under 2.5.5, “User Management” on page 25.

2.5.3 Installation and Configuration

The installation of SP nodes is based on the AIX net-install capability in which an image of a system is cloned on a target systems(s). This allows nodes to be easily restored to a known level of software. Node configuration data, such as hostname, default route and each communication adapter′s IP address and netmask, is entered into the SDR using SMIT or line commands. This data is used after a net install to automatically customize Object Data Manager (ODM) information. The Control Workstation acts as a net-install server for the nodes acting as boot/install servers (typically one boot/install server for each frame). This two-stage structure allows the install of each frame to proceed in parallel.

When a node is rebooted after installation, a user-supplied firstboot script may be executed, performing additional tailoring such as setting name resolution, enabling AFS or Network Information System (NIS), setting paging space, installing additional LPPs, setting licenses or time zone, and so on.

2.5.4 Print Management

Depending on the size of the SP system and the function required, several options are available for print support. Base AIX support for `lpr`, `enqueue` and related commands is available on each node. For small SP systems, this may be an adequate solution. (Optionally, SP support for printing replaces the AIX print commands with scripts that use `rsh` to shift printing from a node to a print host, eliminating the need to manage print queues and run print daemons on every node).

IBM AIX Print Services Facility/6000 (PSF/6000) extends the base AIX print support to provide production printing support on a variety of printers including:

- Channel-attached printers, generally high-speed, high-throughput devices connected via a S/370 Channel Adapter card in an SP node or RISC System/6000 workstation
- TCP/IP-attached printers connected to an SP node or other system
- AIX-defined printers, normally low-speed printers and plotters attached to a serial port of an SP node (using RPQ hardware) or RISC System/6000 workstation

Additional information is provided in the document *IBM AIX Print Services Facility/6000: Printing with the IBM 9076 Scalable POWERparallel Systems*, available from your IBM Representative (on MKTTOOLS as SP2PSF TERS3820).

2.5.5 User Management

Many systems have user-management strategies in place (NIS or rdist of /etc/passwd, for example); the SP can fit into these schemes. For installations that do not have existing user-management mechanisms, commands to add, change, show and delete users are provided. If NIS is used, these commands modify the password file on the NIS master and push the NIS maps. If NIS is not used, the user-management files are distributed through a file collection that is pulled to each node on an hourly basis. Currently, when using file collections to distribute user-management files, users must log into the Control Workstation to change their passwords.

2.5.5.1 Amd

Optionally, Amd, the BSD automounter, can be used to manage NFS mounting of user home directories and other directories. If configured, an amd daemon runs on each node and mounts directories on demand, unmounting them after a period of time if there is no activity. The directories to be mounted can come from servers within the SP or from any server or workstation in the network. Maps are used to indicate to Amd how the mounting should be done. A map entry for a user′s home directory is automatically created when the user is added to the system.

2.5.5.2 Blocking and Unblocking of login

New commands are provided with the SP to allow the administrator to block a user or group of users from accessing a node or group of nodes using login, rlogin, telnet and so on. Many medium- and fine-grained parallel jobs cannot tolerate timing disruptions caused by other users on nodes that were assigned for exclusive use of a parallel job. The Resource Manager, optionally, can enable only the user whose job is being executed.

2.5.5.3 File Collections

The file-collection technology, introduced on the SP1, was developed as part of project Agora at the IBM Watson Research Center and is based on sup (the software-update protocol from Carnegie Mellon University) and a Perl front end called supper. The sup code is public-domain software, distributed as part of the standard SP software.

File Collections simplify the task of managing duplicated files on multiple nodes. Groups of files can be defined as a collection from a single point. Any changes to files in that collection are propagated to all systems that have that collection installed. File-collection servers are arranged in a hierarchical tree structure to insure scalability.

The SP system comes with predefined collections of user-administration files, such as /etc/passwd, and of root control files, such as /etc/services. The administrator can add any desired files to these collections or define additional collections for applications, tools and data.

Part 2. Operating-System and System-Infrastructure Tools

Chapter 3. IBM 3270 Host Connection Program

This chapter describes the use of the IBM 3270 Host Connection Program (HCON) on an SP. The various topics addressed are:

- Installation of the HCON product
- Using HCON with a distributed function terminal
- Using HCON for TCP/IP connections
- File transfer with HCON
- Practical hints and tips

3.1 HCON Description

The following description is from the book *3270 Host Connection Program Guide and Reference*: The HCON program is a 3270 connectivity application for the AIX environment. The HCON program emulates a subset of 327X functions and features, allowing end users at AIX terminals to connect to an IBM System/370 host and appear to the host as an attached IBM 3270 Display Terminal or printer. HCON facilitates file transfers between the workstation and the IBM System/370 host, allows printer emulation and provides High-Level-Language Application Programming Interface (HLLAPI) support for user-provided workstation applications to communicate with 3270 sessions.

HCON connectivity can be through any of the following:

- System Network Architecture (SNA) connection in a Type 2.1 low-entry networking (LEN) node attachment
- Distributed Function Terminal (DFT) SNA attachment to an IBM 3274/3174 or IBM 9370 Workstation Subsystem Controller
- Transmission Control Protocol/Internet Protocol (TCP/IP) Telnet 3270 connection

3.2 Installing HCON

Because you could not order HCON for the SP at the time this redbook was being written, the software we used for testing was extracted from an 8mm tape ordered for an IBM RISC System/6000.

Usually, software that is to be installed on an SP is first downloaded to the /usr/sys/inst.images/ssp directory. You must traverse the following SMIT panels to do so:

```
└─►Software Installation & Maintenance
     └─►Install / Update Software
          └─►Copy Software to Hard Disk for Future Installation
```

And, enter the following information:

```
* INPUT device / directory for software            /dev/rmt0.1
* SOFTWARE name                                    [hcon]
  DIRECTORY for storing software                   [/usr/sys/inst.images/ssp]
  DIRECTORY for temporary storage during copying   [/tmp]
  EXTEND file systems if space needed?              yes
```

After downloading the software, you can NFS mount this filesystem on the nodes for installation over the network. However, HCON needs two Program Temporary Fixes (PTFs) that provide additional terminal support. These PTFs are part of the Base Operating System (BOS). When downloading BOS to disk for future installation, you get the entire operating system, requiring at least an additional 100MB on your install directory and roughly four hours of tape spooling.

The required PTFs are:

- bos.data.U428204
- bos.data.U429640

So, check first whether these PTFs are already installed on your target nodes. If they are installed, skip this section and head for the next.

3.2.1 Installing PTFs for HCON on an SP

Perform the following steps carefully to get only the PTFs necessary for HCON from tape to hard disk:

1. Enter the command: **smit install_subsystems**. In the resultant panel, select the appropriate tape drive and press F4..
2. Search in the list for U428204 and U429640, both are required.
3. If you cannot find these PTFs on any tape at your office, call IBM Software Support and order them.
4. When you find them, exit SMIT.
5. Change the tape drive′s blocksize to 0: **chdev -l rmtX -a block_size=0**, where X is a number corresponding to your tape drive.
6. Spool the tape forward to image three: **tctl -f/dev/rmtX fsf 2**
7. Read the image from tape to disk: **dd if=/dev/rmt0 of=/tmp/toc**
8. Search this file for the entry: `bos.data 03.02.0000.0000.U429640 En_US Terminal Capabilities database` and for a similar entry describing U428204. The lines preceding these entries should look similar to this:

   ```
   01:0148:000158720 (for U428204)
   01:0149:000031744 (for U429640)
   ```

   The two-digit number represents the number of the tape on which the PTF can be found, 01 in this case. The second number represents the position on the tape. The last number gives the size of that image in bytes.
9. To position the tape correctly for U428204, enter the following command: **tctl -f/dev/rmtX.1 fsf 148**.

   Note: Do not forget the **.1** following the tape drive name. This prevents the tape from rewinding after command completion.
10. To download the PTF, enter: **dd if=/dev/rmtX of=/tmp/bos.data.U428204**.

    Note: Omitting the **.1** rewinds the tape.
11. Repeat steps 9 and 10 for the other PTF.

12. Copy or move these files to /usr/sys/inst.images/ssp and build a new table of contents using the command: **/usr/sbin/inutoc /usr/sys/inst.images/ssp**.

3.2.2 Installing HCON on Target Nodes

To complete the HCON installation, you must mount the install directory on the target nodes. Either log into those nodes and run SMIT install, or use the **dsh** command with the install command and target nodenames as arguments. After committing or applying the software, the install list should look similar to this:

```
hcon.obj at level 1.3.0.0
hconmEn_US.msg at level 1.3.0.0
hconmEn_US.msg at level 1.3.0.0.U491200
hconmEn_US.msg at level 1.3.0.0.U428330
hcon.obj at level 1.3.0.0.U428326
hcon.obj at level 1.3.0.0.U428328
hcon.obj at level 1.3.0.0.U428327
hcon.obj at level 1.3.0.0.U491147
hcon.obj at level 1.3.0.0.U430446
hcon.obj at level 1.3.0.0.U430424
hcon.obj at level 1.3.0.0.U430441
bos.data at level 3.2.0.0.U429640
bos.data at level 3.2.0.0.U428204
```

3.3 Connection Requirements

HCON supports the following network communication adapters:

- IBM 3270 Connection Adapter, f/c 2990, for coaxial attachment to either an IBM:
  - 3174/3274 Control Unit
  - 4361 Workstation Adapter
  - 9370 Workstation Subsystem Controller
- For SNA Node T2.1 attachment:
  - 4-port Multiprotocol Communications Controller
  - Single-port Multiprotocol Adapter/2
  - X.25 Interface Co_processor/2 Adapter
  - Token-Ring Adapter
  - Ethernet Adapter
- For TCP/IP attachment using telnet:
  - Token-Ring Adapter
  - Ethernet Adapter
  - X.25 Interface Co_processor/2 Adapter
  - Fiber Distributed Data Interface (FDDI) Connectivity
  - Block Multiplexer Channel Adapter
  - ESCON Channel Connectivity
  - Serial Optical Channel Converter Adapter

For this redbook, we tested a distributed function terminal (DFT) over a coaxial line connecting to IBM Virtual Telecommunications Access Method (VTAM) running under MVS, as well as a TCP/IP session connecting to an MVS host using Block Multiplexer Channel connectivity. These connections are discussed in more detail in the following section.

3.4 Using and Managing HCON

The HCON installation process adds new panels to SMIT. They can be found under:

```
└─►Communication Applications and Services
   └─►AIX 3270 Host Connection Program (HCON)
```

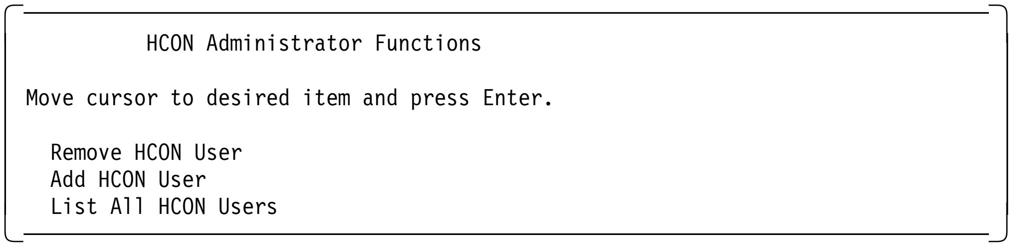
The install process also updates the /etc/inittab file, adding the **hcon** daemon. To use HCON, this daemon must be running.

Managing HCON is a relatively simple task. Only three basic tasks are involved in enabling HCON to serve host sessions:

1. **User Control**: The HCON program will not allow you to add sessions until HCON users are defined. To do so, traverse these SMIT panels:

   ```
   └─►AIX 3270 Host Connection Program (HCON)
      └─►HCON Administrator Functions
   ```

   From the panel shown in Figure 5, select the option Add HCON User. (We do not display the subsequent panel as it contains only a single field, the username).



```
          HCON Administrator Functions

Move cursor to desired item and press Enter.

  Remove HCON User
  Add HCON User
  List All HCON Users
```

Figure 5. Manage HCON Users

2. **Session Control**: Users can have as many sessions defined as desired. The number of concurrent sessions, however, is limited. This is dependent on the host definitions and the connecting hardware (the IBM 3270 Connection Adapter, for example, supports up to five sessions per adapter).

   In Figure 6 on page 33, the SMIT panel for session control is displayed. It was reached through the following panels:

   ```
   └─►AIX 3270 Host Connection Program (HCON)
      └─►HCON User Functions
   ```

   Examples of the definitions we used to add the DFP and TCP/IP display sessions are shown in 3.4.1, "Distributed Function Terminal" on page 33 and 3.4.2, "TCP/IP Connection" on page 35.

```
          Add an HCON Session

  Move cursor to desired item and press Enter.

  Reconfigure an HCON Session
  Remove an HCON Session
  Show Characteristics of an HCON Session
  Add an HCON Session
  List All HCON Sessions
```

Figure 6. Add an HCON Session

3. **HCON Program Control**: The SMIT panel shown in Figure 7 is used to monitor and control HCON. This panel can be accessed via the following SMIT selections:

```
└─►AIX 3270 Host Connection Program (HCON)
   └─►HCON Control
```

```
      HCON Control

Move cursor to desired item and press Enter.

  Start the hcondmn Subsystem
  Stop the hcondmn Subsystem
  Display HCON User Status
  Display SNA LU Pool Statistics
  Display SNA LU Pool Information
  Start an HCON Printer Session
  Stop an HCON Printer Session
  Display an HCON Printer Session Log
```

Figure 7. HCON Control

3.4.1 Distributed Function Terminal

In this section, we explain the steps needed to establish a DFT session under HCON. In addition, we provide a sample VTAM definition to aid in configuring your communications port.

3.4.1.1 Establishing a DFT Session

Our DFT session is running over a IBM 3270 Connections Adapter connected to a IBM 3174 Control Unit. This session can be defined by going through SMIT as follows:

```
└─►Add an HCON Session
   └─►Add DFT Display Session
      └─►* HCON USER name      [peter]
```

After entering the user name, you reach the screen to set the characteristics of your session. Unless you specify a particular session name, HCON will automatically choose the next consecutive available session (that is, if you already have a session “a,” HCON uses “b”).

Figure 8 on page 34 shows the entry fields that can be set (see below for an explanation of the most commonly-used fields).

```
[TOP]                                             [Entry Fields]
  HCON user name                                 peter
  SESSION name                                   a
  Session USE                                   [PROFS]
* Communication DEVICE                          [3270c0]
  3270 MODEL (rows x columns)                    2 (24 x 80)
* LANGUAGE                                       English (U.S.A.)
* KEYBOARD table                                [/usr/lib/hcon/e789_ktb>
* COLOR table                                   [/usr/lib/hcon/e789_ctb>
* File used by SAVES key                        [/u/peter/e789_saves]
* File used by REPLS key                        [/u/peter/e789_repls]
* Local printer used by PRINT key               [lp0]
  Host TYPE                                      TSO
  Host LOGIN ID                                 [kes]
  Autolog NODE ID                               [tso]
  Autolog TRACE                                  yes
  Autolog TIMEOUT (seconds)                     [0]
* Host file transfer PROGRAM                    [IND$FILE]
  File transfer DIRECTION                        down
* File transfer WAIT period (minutes)           [0]
* File transfer RECOVERY time (minutes)         [0]
* Maximum I/O buffer SIZE (bytes)               [2048]
```

Figure 8. Configuring an HCON Session

- **Session USE** defines the string that will display in the lower-right corner of the screen.
- **Host LOGIN ID** is the string used for automatic login by HCON utilities, such as fxfer.
- **Autolog NODE ID** is the representation of the file stored in your home directory as **SYS**id. In this example, the file appears as SYStso.
- **Autolog TRACE** enables autolog tracing when setting up your own Autolog nodeid.

3.4.1.2 Sample VTAM Definition

Here is a sample definition showing how to configure VTAM to understand your communications port. For a more detailed explanation of how to set up your host environment to serve the HCON facilities, please refer to the *3270 Host Connection Program Guide and Reference*.

```
* THIS SWITCHED MAJNODE CONNECTS THE RISC/6K ON THE SYSLAB TOKEN      * 00010000
* RING TO THE MVS HOST. CROSS DOMAIN SESSIONS ARE ESTABLISHED BY MVS. *
*                                                                       00030000
         VBUILD TYPE=SWNET
SLRISC   PU    ADDR=C1,                                                X
               IDBLK=071,                                              X
               IDNUM=06000,                                            X
               IRETRY=YES,                                             X
               MAXPATH=2,                                              X
               MAXDATA=1033,          MAX PIU SIZE INC TH + RU         X
               MAXOUT=7,              MAX PIU SENT BEFORE RESP REQ     X
               PACING=0,              SECONDARY RECEIVES               X
               VPACING=0,             (V) VTAM TO NCP PACING           X
               PUTYPE=2,              PHYSICAL UNIT TYPE TWO           X
               DISCNT=(NO),           NO DISC AFTER SESSION END        X
               ISTATUS=ACTIVE,                                         X
               MODETAB=MODDPFN,                                        X
               USSTAB=ISTINCDT,                                        X
```

```
                SSCPFM=USSSCS
SLRISC1  LU     LOCADDR=0,DLOGMOD=MODDPFC2                LU6.2 SESSION
SLRISC2  LU     LOCADDR=0,DLOGMOD=MODDPFC2                LU6.2 SESSION
SLRISC3  LU     LOCADDR=0,DLOGMOD=MODDPFC2                LU6.2 SESSION
SLRISC4  LU     LOCADDR=0,DLOGMOD=MODDPFC2                LU6.2 SESSION
SLRISC01 LU     LOCADDR=2,DLOGMOD=SNX32702,USSTAB=USSSNA  3270 SESSION
SLRISC02 LU     LOCADDR=3,DLOGMOD=SNX32702,USSTAB=USSSNA  3270 SESSION
```

3.4.2 TCP/IP Connection

Our second connection is set up using a Block Multiplexer Channel Adapter running TCP/IP to an MVS host. You must go through the same SMIT panels as when setting up the DFT terminal, only you will choose the option to add a TCP/IP session:

```
└─►Add an HCON Session
   └─►Add TCP/IP display session
      └─►* HCON USER name      [peter]
```

The SMIT panel shown in Figure 9, contains the same fields as that for DFT, with one exception. The Communication DEVICE field for DFT has been replaced with The HOSTNAME field. Specify the dotted decimal representation of your host in this field:

```
[TOP]                                           [Entry Fields]
  HCON user name                              peter
  SESSION name                                b
  Session USE                                [TCPIP]
* HOSTNAME (name or dotted decimal)          [9.12.8.101]
  3270 MODEL (rows x columns)                 2 (24 x 80)
  LANGUAGE                                    English (U.S.A.)
* KEYBOARD table                             [/usr/lib/hcon/e789_ktb>
* COLOR table                                [/usr/lib/hcon/e789_ctb>
* File used by SAVES key                     [/u/peter/e789_saves]
* File used by REPLS key                     [/u/peter/e789_repls]
* Local printer used by PRINT key            [lp0]
  Host TYPE                                   TSO
  Host LOGIN ID                              [kes]
  Autolog NODE ID                            [tso]
  Autolog TRACE                               yes
  Autolog TIMEOUT (seconds)                  [0]
* Host file transfer PROGRAM                 [IND$FILE]
  File transfer DIRECTION                     down
* File transfer WAIT period (minutes)        [0]
* File transfer RECOVERY time (minutes)      [0]
* Maximum I/O buffer SIZE (bytes)            [2048]
```

Figure 9. Configuring a TCP/IP Connection

Once the session is defined, you can invoke the host emulator by entering the command **e789 a** or, for a windows version of the emulator, **xhcon a**.

Note: When using TCP/IP to make a host connection, the host must also be running the TCP/IP software. Having TCP/IP on the host, allows the use of standard TCP/IP tools (like telnet and ftp) in conjunction with HCON.

3.5 File Transfer with HCON

This section describes the use of file transfer between hosts and RISC System/6000 nodes.

3.5.1 File Transfer During Session

When logged into the host, you can do file transfer in two ways:

- From the emulator subshell: To invoke the emulator subshell, enter <Ctrl-C> from your host session window. This returns an AIX prompt from which you can enter the file transfer command. For example, to transfer your stored mail from the host to your workstation, enter:

  ```
  fxfer -v -t -HCMS ″all notebook a″ notes.host
  ```

  After completion of this command, enter **exit** or <Ctrl-D> to return to your host session.

  Note: You do not need to specify **-d** (the download flag) as it is the default file-transfer direction (refer to the SMIT panel in 3.4.2, “TCP/IP Connection” on page 35). Nor do you need to specify your session name, since you are in it already.

- From a separate window or program: Open a separate AIX window or execute a program from which you will invoke the file transfer command. For example, to transfer a C program from your workstation to the host, enter:

  ```
  fxfer -na -u -v -t -HCMS $HOME/bin/myprog.c ″myprog c a″
  ```

  In this case, the session name and upload flag must be specified.

3.5.2 Automatic File Transfer (AUTOLOG)

It is not necessary to log into your host to perform file transfer. When using the DFT type of connection, you can ask the file-transfer program to log in for you automatically and to perform the transfer. For example, the following command will perform the the download after prompting you for your host user-ID password:

```
fxfer -na -d -v -t -x harris -HCMS ″all notebook a″ notes.host
```

For this to work, you must create a file in your home directory with the keystroke sequence that is normally executed when logging into the host system. Examples of such a file can be found in the directory /usr/lib/hcon . Samples of keystroke files to TSO, VM1 and VM2 are provided as **SYStso, SYSvm1** and **SYSvm2** . You can update, create and verify your own keystroke file by using the **hconutil** command with the option **Configure AUTOLOG profiles** (**genprof**).

3.6 Summary

The HCON program is well-suited to running on an SP. When consolidating your workstation environment to an SP, HCON is a useful tool to have in your collection.

The following list gives you a few hints and tips from our testing of HCON:

- End a session with <Ctrl-D> twice. Should your session hang, use the **e789cln** command to kill those sessions.

- Use telnet to connect to the node serving the HCON facilities. The rsh and rlogin programs do not handle the <Ctrl/Act> key properly.
- When using the AUTOLOG facility, use file collections to handle the update of these files.
- If file transfer is not used regularly, do file transfer from a logged in session rather than using AUTOLOG.
- Issue **fxfer -h** for a description of how to use the transfer utility.

Chapter 4. Kerberos

Kerberos is used for services authentication on the SP. Kerberos is basically the watchdog of the system. Its name is derived from Greek mythology, where Cerberus guarded the entrance to the underworld.

Kerberos provides authentication services that allow certain distributed services within the SP system and between it and other workstations to securely control access to their facilities. It is used to provide secure remote command capbility, used by the SP system installation programs and available for general use in place of the standard AIX remote shell and remote copy commands, which are notoriously insecure. The root user must use Kerberos when installing the SP system, taking on the role of Kerberos administrator, because the installation process includes the creation or modification of the Kerberos security database. The System Monitor that administrators use to monitor and control the SP hardware resources uses Kerberos to ensure that only authenticated users can invoke its functions through either the graphical or command-line interfaces. Generally a small number of administrative users would expect to use Kerberos for that purpose. More general use of Kerberos by other users is possible but not required. It is required for the exploitation of the authenticated distributed command execution facilities: sysctl, dsh, the "p*" commands and the enhanced rsh and rcp commands

Kerberos can best be compared with going to the motor races, say the Grand Prix of Monte Carlo. Going to Monte Carlo requires you to show your passport. This is, in fact, equal to logging into the system. Now, in order to enter the race track, you must be a member of the Grand Prix Racing Club. In terms of Kerberos, you must be a known member of the Kerberos database to get access to the Kerberos services. Having a seat in the grandstand, getting access to restricted areas such as the pits, require additional permits. You can get those permits by buying tickets. These tickets, however, are only valid for a certain amount of time. In terms of Kerberos, you must ask Kerberos to give you a ticket granting you permission to use certain services. As with the races, where you can only get a pit ticket when you are a member of the Grand Prix Racing Club, with Kerberos, you can only get a ticket when you are known to Kerberos.

Kerberos is delivered with the SP as a vehicle to enable the highest SP security. The main reason for implementing Kerberos on the SP, and on UNIX workstations in general, is to avoid the use of the **.rhosts** file of the root user. This does not mean that SP users are not allowed to use this file anymore; users who still want to have all network services available can do so. Kerberos is a method of authentication that is not related to any AIX authentication system nor is it used as an additional login user password verification. Guarding your (root) password is still critical to having a secure environment. As soon as someone other than the system administrator can log into the system as root, he or she can destroy the Kerberos database anyway. Rather, Kerberos is used as a system for use of authenticated services within the SP.

The authentication services provided with the SP are based on MIT′s Kerberos version 4. These services use the Data Encryption Standard (DES) algorithm. Due to export regulations, DES is restricted for use to the United States only. Therefore, a subset of DES is implemented in the SP. This means that no interfaces or libraries are provided to implement additional security (authenticated) services on the SP.

The commands developed with Kerberos subroutines and functions are:

- spmon, the System Monitor
- rsh
- rcp
- sysctl

All other authenticated Kerberos commands are built on top of one of the commands mentioned in the list and therefore also authenticated through Kerberos.

4.1 Kerberos Terminology

Understanding Kerberos authentication is aided by being familiar with the following terms:

Realm	A Kerberos domain that can consist of a number of machines providing authentication services. The default name of a realm on an SP is the TCP/IP domain name converted to uppercase. For example, `ITSC.POK.IBM.COM` is the realm name derived from the domain name itsc.pok.ibm.com.
Principal	A user or a service that uses authentication services and is identified in the authentication database. For example, `root.admin@ITSC.POK.IBM.COM`, where `root` is the user identity and `admin` the instance.
Instance	In the case of a service instance, it represents the occurrence of the server. In the case of a user, the instance represents the Kerberos authority granted to the user. Example 1: `root.admin`, where admin represents a Kerberos authorization for administrative tasks in Kerberos. Example 2: `hardmon.sp21cw0`, where `hardmon` represents the service and `sp21cw0` represents the node providing the service.
Authentication Database	A set of files containing the definitions of the Kerberos authentication information. The authentication database on an SP is stored in `/var/kerberos/database`.
Ticket	An encrypted message containing the identity of a user. A ticket is passed from a client to a server as soon as a Kerberos service is requested. Tickets are stored in the `/tmp` directory in file `tktuid` (where `uid` is the client user ID)
Key	An eight-byte form of a user or service password stored in the authentication database. This password is associated with a Kerberos user or service principal, not a user ID.
Ticket-granting ticket	A ticket that is generated by the Kerberos authentication database as proof that Kerberos recognizes the user as an authorized user.

4.2 Kerberos Authentication — A Picture

Figure 10 depicts, step by step, the activities triggered by the two Kerberos commands kinit and rcmd.

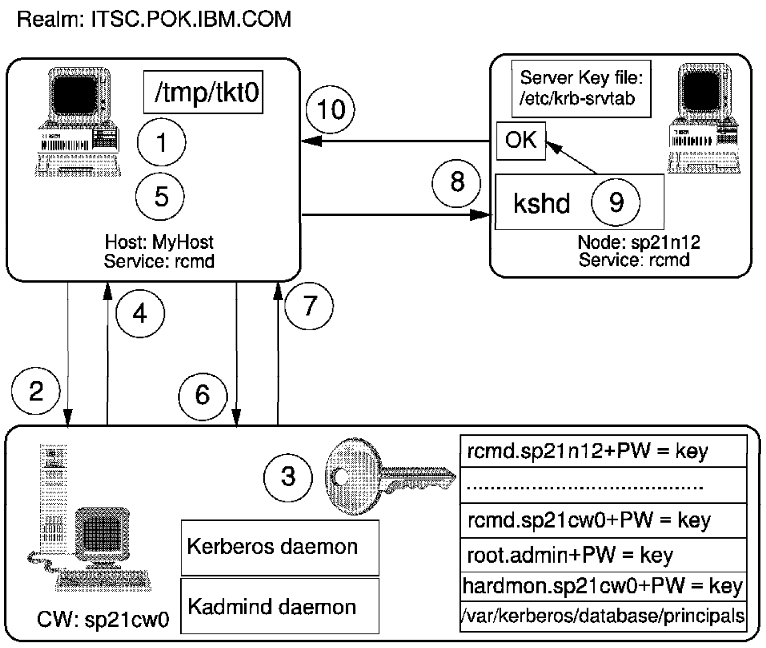


Figure 10. Kerberos In Action

Step 1. The user enters the kinit command. Communication is set up with the Kerberos Master machine. The username is passed to the Kerberos server.

Step 2. A request is sent, by kinit, to the authentication server, containing the user′s (principal) name and the name of a special service, known as the ticket-granting service.

Step 3. The authentication server checks that it knows about the client. If so, it generates a random session key, which will later be used between the client and the ticket-granting server. It then creates a ticket for the ticket-granting server, which contains the client′s name, the name of the ticket-granting server, the current time, a lifetime for the ticket, the client′s IP-address, and the random session key just created. This is all encrypted in a key known only to the ticket-granting server and the authentication server. The principal′s password is used for encryption. In a formula, the authentication server sends the following information back:

```
{{TGT}Key(krb), Key(tgt)}Key(password)
```

Note: In most cases (and as shown in Figure 10) the authentication server is the same as the ticket-granting server.

Step 4. The authentication server then sends the ticket, along with a copy of the random session key and some additional information, back to the client. This response is encrypted in the client′s private key, known only to Kerberos and the client, which is derived from the user′s (principal) password. Once the response of the authentication server has been received by the client, the user is prompted for a password. The password is converted to an encrypted key (using DES) and used to decrypt the response from the authentication server. If authentication succeeds (that is, if the Kerberos password matched), the ticket sent by the server is stored in the user′s ticket cache file, then known as a Ticket-Granting-Ticket. The ticket cache file looks like **/tmp/tkt**uid file, in the case of root /tmp/tkt0.

Step 5. The user now enters, for instance, a rsh command to obtain information from a server. The rsh command will build a packet, a so called authenticator, containing the client′s IP address and the current time.

Step 6. This authenticator (which will be or is encrypted), along with the previously obtained ticket-granting-ticket, is sent to the ticket-granting server. Once the authenticator and the ticket have been received by the server, the server decrypts the ticket, decrypts the authenticator, compares the information in the authenticator with the information of the ticket, derives the IP address from which the request was received and the present time. If everything matches, and the ticket has not expired, the ticket-granting server will search the Kerberos database for a rcmd-command ticket for sp21n12.

In a formula, this request would look like this:

```
{auth}Key(tgt) {TGT}Key(krb) rcmd-service
```

Step 7. The Kerberos Master (the ticket-granting server) returns an sp21n12-rcmd ticket. This ticket will also be stored in the user′s ticket cache file /tmp/tktuid (entering **klist** will show all tickets).

In a formula, the server sends back the following packet:

```
{{rcmd-ticket}Key(rcmd) Key(session)}Key(tgt)
```

Step 8. The client creates an authenticator, auth2, and sends it, with the ticket received in Step 7, to the server sp21n12.

The offered packet looks like:

```
{auth2}Key(session) {rcmd-ticket}Key(rcmd)
```

Step 9. The kshd daemon wakes up, decrypts the ticket and validates the client.

Step 10. If all of the tests are passed, the command is executed and the results returned.

4.3 Security Levels

Customers have various requirements as to what must be implemented to make their networks secure. Different levels of security impose different implementation methods and policies. In the following two sections, we will discuss the two extremes in security implementation, that is, limited and strict. We are providing guidelines for security levels; anyone is free to choose an implementation that is a compromise between the two.

Whatever implementation you are going to use, the Kerberos server (and possible backup servers) must be physically secure in order to meet the security requirements of a Kerberos environment. This means that the existence of a *.rhosts* file and remote logins must not be allowed on the Kerberos database server. But, you are free to choose otherwise on the SP′s Kerberos server , in most cases the Control Workstation.

4.3.1 Limited SP Security

Because the Systems Monitor and the dsh command are built on top of Kerberos, once the Kerberos software has been installed, the root user must use Kerberos when configuring the SP nodes. Customers making use of the .rhosts utilities can still do so. Be sure, however, to use the **rcp** and **rsh** commands from the `/usr/bin` directory. The only task that needs to be performed is to set up the Kerberos database with the basic principals defined. This means that setting up Kerberos with the *root.admin* principal at installation time is sufficient. The program that defines and initializes the Kerberos database is **setup_authent**. This program prompts the user to set the following attributes:

1. **Kerberos Master Key**

   Note: Do not forget this password. There is no way to recover from forgetting this password, except reconfiguring the Kerberos database. In 4.4, "Managing Kerberos" on page 45, this will be discussed in more detail. However, you are unlikely to ever have to use the master key, unless you specifically want to change the master key. Making sure you have a backup of the */.k* file is as good as remembering the master key.

2. **The Administrative principal**

   This principal is called *root* and belongs to the instance *admin*. The Kerberos administrator can elect to use another AIX user ID in place of or in addition to root; in which case, the administrative principal′s name must be the same as the AIX user name (see the note below).

   We recommend that you create the *root.admin* principal at installation time. This principal carries a few attributes that can be set, shown in the example below. However, we recommend that the default values, which will be prompted, be used. The Kerberos administrator can set this instance′s password to anything he or she chooses.

   ```
   Principal: root, Instance: admin, kdc_key_ver: 1
   New Password:
   Verifying, please re-enter
   New Password:

   Principal′s new key version = 1
   Expiration date (enter yyyy-mm-dd)
   [ 1999-12-31 ] ?
   Max ticket lifetime (*5 minutes) [ 255 ] ?
   Attributes [ 0 ] ?
   Edit O.K.
   ```

   Note: The *IBM 9076 Scalable POWERparallel Systems SP2 Installation Guide* shows an example where the administrative principal is called **poobah**. This could cause a problem. The **kadmin** command expects a user ID to be the same as the principal′s name. In this case, you should either add an AIX user called poobah or add the root.admin principal after setup_authent. If your Kerberos administrator is defined as poobah.admin and you try to add a principal with kadmin when logged in as root, the following message will appear:

```
Principal root.admin@ITSC.POK.IBM.COM does not exist.
```

This is as designed. If you want to use another administrative principal than the root.admin principal, you must make sure this principal is also defined as a user in the /etc/passwd file and, as root, issue kadmin as follows:

```
kadmin -u poobah
```

3. **The Service Principals rcmd and hardmon**

   These principals are created automatically by the next step of the setup_authent command. The user must authenticate him or herself here for the first time to Kerberos. The principals **rcmd** and **hardmon** are configured into the Kerberos database with random passwords. We do not recommend that you change these passwords since it is not necessary to know them. They are maintained by the Kerberos system.

4. **The Service Principal rcmd for the Nodes**

   The setup_server command completes the initialization of Kerberos by adding the rcmd principals to the Kerberos database. It also creates the **.klogin** file for the root user.

After completion of this task, the Kerberos environment is ready for use.

Note: An APAR (IX49179) is nearing shipment, or is already available as this book is published, that fixes an Estart failure when a switch primary node′s hostname is set to the switch interface name. The fix also removes the prior restriction that was built into the MIT Kerberos package that multi-homed hosts were not supported. Prior to the fix, servers had to identify to Kerberos as a single instance of a service, where instance is the short hostname. With the application of this change, instances or the "rcmd" services are created for all (short) network names for each node and the Control Workstation. Refer to Appendix B, "New Features For ssp.clients" on page 237 for more information.

4.3.2 Strict SP Security

This section describes how to set up the SP in the most secure way. Setting up the initial Kerberos database is the same as described in 4.3.1, "Limited SP Security" on page 43. In addition, the following tasks must be done:

1. Because network logins cause the passwords of users go over the network in an unencrypted form, you must disable all TCP/IP network login facilities that require a user ID and password for connecting to that service. Edit the file /etc/inetd.conf and comment out the following services:

   - telnet
   - exec
   - shell
   - login
   - ftp

   After changing the /etc/inetd.conf file, run **inetimp** followed by **refresh -s inetd**. This activity nullifies the existence of the .rhosts file.

2. We recommend that you limit the use of **tftp** to the root user only. The install process of the PSSP package uses tftp to copy Kerberos files from the Control Workstation or assigned install server to the node being installed.
3. If you require more than one administrator, create additional admin (Kerberos) users with unique passwords.
4. Register all SP users to the Kerberos database. User principals carry the same names as the users′ login names. If you add "normal" users to Kerberos, you must be aware that certain restrictions exist for non-root users. As also discussed in 4.4.1.2, "Adding Principals to the SP Realm" on page 47, the maximum lifetime of a ticket is 21.25 hours. This means that users (except for root) cannot run a batch jobs that will invoke remote services after more than 21.25 hours. After expiration of the user′s ticket, the user must invoke **kinit** again. Also, only the root user can run **rcmdtgt** to non-interactively create a new Ticket-Granting Ticket for remote commands. Normal users cannot do so because this command requires read access to the `/etc/krb-srvtab` file. This is a known restriction of the SP Kerberos implementation. Other Kerberos restrictions will be discussed in 4.5, "Kerberos Restrictions" on page 57.
5. Ensure that every workstation or PC connecting to the SP is part of the Kerberos realm and has Kerberos services implemented.
6. By disabling the remote login services, users can only log into a workstation or PC. Any command executed on a remote machine must (and can only) be done by **rsh, rcp, sysctl, dsh** or **pcp**.
7. Use of Xstations is not recommended in a strictly-implemented secure environment. As with the network login utilities, an Xstation requires a remote login over the physical network.

4.4 Managing Kerberos

This section discusses how to manage the various functional parts of Kerberos.

As mentioned in 4.3.1, "Limited SP Security" on page 43, after completing the configuration tasks in the System Data Repository (SDR), the Kerberos database is ready for use on the Control Workstation. To test the Kerberos commands, try invoking the **rsh** or **rcp** command from the `/usr/lpp/ssp/rcmd/bin` directory. For example, issue: `/usr/lpp/ssp/rcmd/bin/rsh CW-hostname ls`. Notice that the behavior of this rsh command has changed. The command no longer allows you to log into a node. So, users who used to issue rsh `<nodename>` without arguments will get the following message:

```
rsh: 0041-001 No arguments listed for rsh.
usage: rsh host [ -l login ] [ -n ] command
```

The rcp command will behave as before.

Note: One of the best tools for performing systems-management tasks is the **sysctl** command. This command is discussed in the *IBM 9076 Scalable POWERparallel Systems SP2 Administration Guide*.

4.4.1 The Realm

The *realm* in Kerberos is a set of systems that is served by one authentication database. The configuration setups may vary:

- A realm may contain only one system.
- One realm may serve an entire network.
- Multiple realms may occur in one network.
- A system may belong to only one realm.

4.4.1.1 Initializing the SP Realm

The **setup_authent** command is used by the SP systems-management software to initialize the Kerberos database. In general, this command is executed on the Control Workstation, but could also be executed on any AIX system client if you wish to run remote and monitor commands from those systems. However, any system that is part of the realm may be configured as the Kerberos database server. The setup_authent command also creates the three initial principals:

- root.admin
- hardmon.<CW-name>
- rcmd.<CW-name>

Note: With APAR IX49179, this list of service pricipals is no longer complete and will depend on the number of network interfaces on the system.

The setup_authent command performs the following functions:

kdb_init This command initializes the Kerberos database. This command can be given an argument. If an argument is omitted, the TCP/IP domain name is chosen as the realm′s default name and converted to uppercase.

```
[root@sp21cw0] / >kdb_init
Realm name [default  none ]:
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.

Enter Kerberos master key:

[root@sp21cw0] / >
```

kstash This command prompts for the master key again and stores it in encrypted format in the **.k** file in the root (/) directory.

update inittab To enable Kerberos serving of a network, the Kerberos daemons must be invoked and added into the /etc/inittab file. Two entries are added to the /etc/inittab file:

- kerb, to start the **kerberos** daemon. This daemon is responsible for providing Ticket-Granting Tickets to clients. This process listens to port number 750/udp, unless defined differently in the /etc/services file. With the command **netstat -a**, you can verify whether TCP/IP is indeed listening.
- kadm, to start the **kadmind** daemon. This daemon is responsible for serving the Kerberos administrative tools, such as changing passwords and adding principals. This daemon listens to port number 751/tcp.

- When you are using a backup Kerberos server, you would also have an entry for the **kpropd** in /etc/inittab. This daemon will listen to port number 754/tcp.

4.4.1.2 Adding Principals to the SP Realm

These steps are also performed by the setup_authent command.

kdb_edit This utility is used for adding the root.admin principal. When invoked without a flag, the administrator is asked to enter the Kerberos master key. The **-n** flag is used to indicate that the master key can be found in the **.k** file. Principals must be given a password and lifetime characteristics. See also 4.3, “Security Levels” on page 42.

Set Kerberos ACLs The only user that is enabled to perform administrative tasks is the root.admin principal. This is recorded in the administrative Access Control List (ACL) files in the /var/kerberos/database directory. The authorization levels that Kerberos provides are:

- Adding principals
- Getting principals (retrieve for reference)
- Modifying principals and passwords

add_principal This command is basically the same as the ank subcommand of kadmin, but is particularly useful when adding large numbers of users or for use in batch files. The add_principal command allows the users password (unencrypted) as input and always takes the defaults values for expiration date and maximum ticket lifetime. The add_principal command can add principals to any realm with the **-r** flag. When not specified, the principals are added to the local realm. Typically, this command is used in the following manner:

```
/usr/lpp/ssp/kerberos/bin/add_principal <filename>
```

The format of the file containing the principals is:

```
name[.instance][@realm] password
```

Note: Be careful with this file, since it contains the unencrypted passwords. Ensure that access to the file is restricted, or remove the passwords from the file after adding the principals.

kinit The **kinit** command is invoked with your principal′s name. For example:

```
kinit root.admin[@realm]
```

By entering the password, you identify yourself to the Kerberos database. When Kerberos recognizes you as a valid user, Kerberos generates a Ticket-Granting Ticket (TGT) that is stored in the file **/tmp/tkt**uid. This TGT has a maximum lifetime, unless otherwise specified, of 21.25 hours (255 * 5 minutes). This means that users must be authenticated by Kerberos on a daily basis.

There is, however, a batch-type command that does a kinit without prompting for a password: **rcmdtgt**. The rcmdtgt command is particularly useful for applications that run for longer than the maximum lifetime of a Kerberos ticket. The rcmdtgt command, however, only creates a ticket-granting-ticket for the remote commands and not for the system monitor, spmon. This command should be used for background execution where you would assign a different ticket cache file than the default, /tmp/tkt0, since the rcmdtgt command does a kdestroy like action unless you do the following in your script:

```
export KRBTKFILE=/tmp/tkt$$
/usr/lpp/rcmd/bin/rcmdtgt
/usr/lpp/ssp/bin/dsh -a <command>
rm $KRBTKFILE
unset KRBTKFILE
```

The commands **kinit, rcmdtgt, ksrvtgt, kdestroy** and **klist** always use the environment variable KRBTKFILE if defined, defaulting to /tmp/tkt<uid> if it is not defined. In addition to use with background processes and scripts, interactive users who share the root (or other) user ID, should use the KRBTKFILE environment variable to avoid destroying each other′s tickets.

4.4.1.3 Miscellaneous

When configuring the SDR, setup_authent is invoked by the **setup_server** command. After the completion of setup_authent, the setup_server command performs a few additional tasks:

1. Creation of the nodes′ **rcmd** services. The setup_server command searches the SDR for all defined nodes in the SP and adds a **rcmd.<nodename>** principal to the realm for each node. The passwords for these principals are generated randomly.
2. Creation of the Server key files for all defined nodes in the SDR. Each node in the SP must have a Server key file, stored in the node′s `/etc/krb-srvtab` file. The setup_server command will prepare this file in the `/tftpboot` directory if the node is not configured for **DISK** with regard to the **Response from Server to bootp Request**. The files created are of the format: **<nodename>-new-srvtab**.
3. When installing a node, the setup_server command will create the local node root user′s .klogin file, copy the Server key file from the install servers `/tftpboot` directory to /etc/krb-srvtab and copy the files **/etc/krb.conf** and **/etc/krb.realm** from the install server to the node.

4.4.2 Managing the Kerberos Database

Maintaining the Kerberos database requires only one task of the Kerberos administrator, that is keeping a flat-file dump of the contents of the Kerberos database. The command that enables him to do so is **kdb_util dump** `filename`. Preferably, the Kerberos administrator should maintain a daily dump of the Kerberos database. This can either be done manually or automatically (using cron). Make sure that the dump files are given restricted access permissions. You can do so by issuing:

```
kdb_util dump /var/kerberos/database/dump_<datestamp>
```

Another way of saving the Kerberos database regularly, is adding an entry of the **/usr/kerberos/etc/push-kprop** command to cron. This command is typically used in environments where a backup server is used but is also useful when you do not have a backup server installed in your realm. This command will dump the

Kerberos database in the same format as that created by **kdb_util dump** command, to a file /var/kerberos/database/slavesave. This will provide a dump of the database in a secure directory. When the backup server exists, so does a file called slavelist, containing all the "slave" servers, to where the dumped Kerberos database will be sent. An explanation of why a dump of the database is useful to have is explained in 4.4.2.3, "Recovering from Corrupted Kerberos Database" on page 50.

4.4.2.1 Changing the Master Key

The Master Key of the Kerberos database (that is, the password required to modify the Kerberos database) can be changed with the **kdb_util** command. Perform the following steps to change the Master Key:

1. **kdb_util new_master_key** filename, where filename represents the file in which to dump the Kerberos database contents.

   ```
   [root@sp21cw0] / >kdb_util new_master_key Kerby.dump

   Enter the CURRENT master key.
   Enter Kerberos master key:

   Now enter the NEW master key.  Do not forget it!!
   Enter Kerberos master key:

   Don′t forget to do a kdb_util load Kerby.dump to reload the database!
   [root@sp21cw0] / >
   ```

2. **kdb_util load Kerby.dump** This command reloads the database with the contents of the file Kerby.dump.

3. **kstash** This command updates the **.k** file in the / directory, the Master Key cache file. This file is particularly useful for entering Kerberos database commands without entering a password, like **kdb_edit -n**. The **.k** file is convenient for not having to enter the password for database commands, but is necessary for starting the Kerberos daemons from ./etc/inittab without operator interaction.

   Note: When the **.k** file is deleted by accident, or corrupted, the **kstash** command can be used to recreate it.

4.4.2.2 Editing the Kerberos Database

The Kerberos database can be manipulated and changed using the **kdb_edit** command. Characteristics of principals can be changed, and principals can be added to the Kerberos database. The kdb_edit example given in the *cmdref.* is not correct. It should look like this:

```
[root@sp21cw0] / >kdb_edit -n
Opening database...
Previous or default values are in [brackets] ,
enter return to leave the same, or new value.

Principal name: rcmd
Instance: mroz

<Not found>, Create [y] ? y

Principal: rcmd, Instance: mroz, kdc_key_ver: 1
New Password:
Verifying, please re-enter
New Password:
```

```
Principal's new key version = 1
Expiration date (enter yyyy-mm-dd) [ 1999-12-31 ] ?
Max ticket lifetime (*5 minutes) [ 255 ] ?
Attributes [ 0 ] ?
Edit O.K.
Principal name:
(press enter to exit kdb_edit)
```

4.4.2.3 Recovering from Corrupted Kerberos Database

There are several reasons why a Kerberos database could be corrupted. While it is not very useful to discuss all of these possibilities, it is important to discuss how to recover. As discussed in 4.4.2, "Managing the Kerberos Database" on page 48, is is very useful to maintain a dump of the Kerberos database. If you are certain that the Kerberos database has been corrupted and that no changes have been made to the Server Key files (/etc/krb-srvtab) on the nodes and the Control Workstation, this procedure might help you to recover.

1. Empty the ticket cache file (/tmp/tkt0) using the **kdestroy** command. Issuing the command **klist** should show you that no ticket file is found.
2. Edit the /etc/inittab file and alter the entries for **kadm, kerb** and **hmon**, changing the respawn value to off. Or, if you are familiar with the **chitab** command, issue:

   ```
   chitab "kadm:2:off:/usr/lpp/ssp/kerberos/etc/kadmind -n"
   chitab "hmon:2:off:/usr/lpp/ssp/bin/hmon.start"
   chitab "kerb:2:off:/usr/lpp/ssp/kerberos/etc/kerberos"
   ```

   After this, issue **telinit 2** and kill (-15) the hardmon daemon.
3. Destroy your Kerberos database with the **kdb_destroy** command. The command will prompt you with the following message:

   ```
   You are about to destroy the Kerberos database on this machine.
   Are you sure you want to do this (y/n)?
   ```

   When successful, the command response will be:

   ```
   Database deleted at /var/kerberos/database/principal
   ```
4. Delete the **/.k** file.
5. Reinitialize the Kerberos database with the **kdb_init** command. If no arguments are submitted, the **kdb_init** command assumes no realm name. When a TCP/IP domain is defined, the realm name will be the same as the domain name, but converted to uppercase.

   ```
   Realm name [default  none]:
   ```

   Press Enter to continue.
6. Recreate the **/.k** file with **kstash**.
7. Reload the database with the dump file created earlier by the Kerberos Administrator. Issue the command: **kdb_util load** filename.
8. Restart the Kerberos daemons. Either edit the /etc/inittab file and change the values of the **kadm, kerb** and **hmon** entries to respawn or use **chitab**. After this, issue the command: **telinit 2**.
9. Create a new ticket for the root user by entering the command: **kinit root.admin**.

After completion of these tasks, the Kerberos database should be up and running again. Test this by starting the Systems Monitor using **spmon -g**.

Note: A more detailed and drastic recovery script is provided in 4.4.6, "Sample Recovery Script" on page 53.

4.4.3 Managing Principals

This section details how to manage Kerberos principals.

4.4.3.1 Adding Principals

Adding principals can be done in one of three ways:

1. With the use of the **kdb_edit** command. This command can only be issued by root. If the **.k** file exists, kdb_edit can be invoked with the **-n** flag. This allows the root user to enter this command without specifying the master key. Following is an example in which the user's entry will expire after June 1st:

```
[root@sp21cw0] / >kdb_edit
Opening database...

Enter Kerberos master key:

Previous or default values are in [brackets] ,
enter return to leave the same, or new value.

Principal name: peter
Instance: admin

Principal: root, Instance: admin, kdc_key_ver: 1
Change password [n] ?
Expiration date (enter yyyy-mm-dd) [ 1999-12-31 ] ? 1995-06-01
Max ticket lifetime (*5 minutes) [ 255 ] ?
Attributes [ 0 ] ?
Edit O.K.
Principal name:
Press <enter> to exit kdb_edit
```

2. With the use of the **kadmin** command. This administrative tool is typically useful for adding user principals to the Kerberos database interactively. By default, the root user is the only user that is granted to change, add and view principals. These permissions are registered in the Kerberos ACL files in the /var/kerberos/database directory. Any user known to Kerberos may be granted to any of these three administrative tasks; however, the user must be a member of the Kerberos admin instance. Adding another user for any of these permissions can be done by editing one of the respective ACL files. Following is a sample interation of the **kadmin** command:

```
[root@sp21cw0] / >kadmin
Welcome to the Kerberos Administration Program, version 2
Type "help" if you need it.
admin:  ank
Usage: add_new_key user_name.
admin:  ank peter
Admin password:
Password for peter:
Verifying, please re-enter Password for peter:
peter added to database.
admin:  quit
```

```
Cleaning up and exiting.
[root@sp21cw0] / >
```

3. In batch mode by the **add_principal** command. The **add_principal** command uses a file as input. This file contains the principals that need to be added to the Kerberos database. The format of the file should be: name[.instance][@realm] password.

4.4.3.2 Deleting or Changing Principals

Changing the instance of a principal or deleting the principal is not supported by a command. For these two actions, you must do the following:

1. Make a dump of the Kerberos database with: **kdb_util dump** `filename`.
2. Edit the `filename` and change or delete the instance of the user or service. The instance entry can be found in the second column of the dump file.
3. Save this file and quit from it.
4. Load this changed dump into the Kerberos database with: **kdb_util load** `filename`.

The instance can now be either removed or changed.

4.4.4 Use of the .klogin File

The **.klogin** file is used in the same manner as the .rhosts file. However, the files differ in that .klogin does not specify remote hosts allowed to log in without passwords, but allows remote principals to use a local user account (that is, the user account that owns the .klogin file). If the client (originating remote user) is authenticated to one of the principals named in the .klogin file, access is granted. The owner of the account is granted access if there is no .klogin file. However, when a .klogin file is present, the owner must also have a principal listed in order to be granted access. The .klogin must be of the following format:

```
name[.instance]@realm
```

For instance,

```
peter@ITSC.POK.IBM.COM
```

- or -

```
rcmd.sp21n12@ITSC.POK.IBM.COM
```

In the description of the .klogin file given in the *SP Command and Technical Reference*, the .klogin example is incorrect. You cannot specify principal names without a realm name.

Note: The root user must always have a .klogin file and, therefore, be listed in the .klogin file. This is as designed.

There is, however, one particular occasion when the root user should be very careful. Should the root user decide to add another user's principal to the .klogin file, that user could run every command as root. If entry `peter@ITSC.POK.IBM.COM` is added to the .klogin file, the user `peter` can do everything that root can do on the node where this .klogin resides. For example, if prior to his addition to the .klogin file, user `peter` were to issue the command:

```
[peter@sp21cw0] /u/peter >cat /etc/security/passwd,
```

the system would respond with the following error message:

```
cat:0652-050 Cannot open /etc/security/passwd.
```

However, once added to the .klogin file, peter could successfully access the /etc/security/passwd file with the following:

```
[peter@sp21cw0] /u/peter >rsh sp21cw0 -l root "cat /etc/security/passwd"
```

Note: For this reason, adding user principals to the .klogin file is not recommended. However, should it be required to delegate root responsibilites without giving out the root password, one should use **sysctl**. Refer to 8.3.5, "Sysctl" on page 84, for a detailed explanation on sysctl.

The success of the above command would only be possible if peter is able to authenticate himself to the Kerberos database with kinit. So, although someone might be able to log in as peter, he must still know the Kerberos password for the principal peter.

For most users, you can say that Kerberos introduces another level of security. For the root user, however, this is not the case. As you will see in 4.4.6, "Sample Recovery Script," the root user is able to delete and recreate the Kerberos database without any Kerberos authentication. The root user only needs to know the root user ID password.

4.4.5 Using the .k File

As mentioned in 4.4.3.1, "Adding Principals" on page 51, **.k** is the file that contains a copy of the Master key. This file is used by the Kerberos database and is useful with the kdb_edit command. Specifying this command with the **-n** flag does not prompt for the Master key when the **.k** file exists. At installation time, the Kerberos administrator is asked to set this password and is reminded not to forget it. All of this precaution is not necessary. As soon as someone (with bad intentions) is able to log in as root, he or she can run any Kerberos commands, delete and reinitialize the database and, worse yet, change the password for the Kerberos root.admin principal. In that case, the system is open. So, for the root user, the old rules for safeguarding the password apply as much now as before using Kerberos. For the root user in particular (but also for other Kerberos users), there are only two benefits of Kerberos protection:

1. No need for the .rhosts file. This means that no machine can possibly manifest itself as a "known" system. Kerberos really knows who is using network facilities and will allow no one else to do so.
2. No unencrypted passwords going over the network, provided that no standard TCP/IP network services are used. Should the need arise to do so, use "Kerberized" versions of the network services.

Note: The Kerberos implementation on the SP does not provide any service with which to log in remotely. In the rsh command shipped with the PSSP software, the ability to log in (that is, issue rsh) without arguments has been disabled.

4.4.6 Sample Recovery Script

In case no standard command works to recover the Kerberos database, you might find the following script (Kerby.Fixit) useful.

Note: You may also refer to Appendix C, "Diagnosing Authentication Problems" on page 241 when you experience authentication problems.

When you are working with a default-configured Kerberos database, this script basically repeats the SDR configuration scripts. After completion of this script, the Kerberos database is configured as if it were installed just after completing the SDR configuration steps. If you have any other user or services entries to add to the Kerberos database, that must be done in additional steps after Kerby.fixit completes. This script does not provide a means of seeing which principals are different from those of the default installation process. Because your database is corrupted, the commands used by this script may not be able to read it, so the script starts again from scratch.

Note: If your environment uses strict security, the rcp command in the following script will fail because it relies on the existence of the .rhosts file in the home directory of root. However, there is a circumvention that avoids having to “CUSTOMIZE” the SP nodes. Create an entry in the /etc/inittab file that performs the copy of the Server Key file each time the SP is rebooted. Preferably, the already-created Server Key files, of format <**nodename**>**-new-srvtabs**, should then be permanently stored in a directory other than /tftpboot. This directory is used by the setup_server command. In so doing, setup_server might remove the nodes′ Server Key files. To avoid this, change the TMPDIR variable in the following script to an existing directory on the Control Workstation. In addition, create an entry in /etc/inittab (before the **sp** entry) that looks like this:

```
updksrv:2:wait:/usr/local/update.SrvKey
```

Following is an example of the above-mentioned update.SrvKey script:

```
#!/bin/ksh
exec > /tmp/update.SrvKey$$ 2>&1
# Determine My HostName First
MYNAME=hostname -s
# Determine IP address of install server
INSTSERVER=cat /etc/ssp/server_hostname | awk ′{print $1}′
mv /etc/krb-srvtab /etc/krb-srvtab.orig

# This command basically can only fail if Server Key
# file does not exist or if network is down.

tftp -o /etc/krb-srvtab $INSTSERVER \
        /YourKeySrvDir/$MYNAME-new-srvtab

# If command fails, put it back the way it was.

if [ “$?” -ne “0” ]
then
    echo ″Log into node with a console...″
    echo ″that is safe...over PRIVAT RS232 line...″
    echo ″and see what is going on...″
    echo ″Check the /tmp/update.SrvKey$$ file for errors...″
    mv /etc/krb-srvtab.orig /etc/krb-srvtab
fi
exit
```

Recovery Script: **Kerby.Fixit**

```
The following script provides a way to recover from a corrupted
Kerberos environment.
At the end of the script, where the Kerberos service key files are
copied to the nodes, the script expects an .rhosts file to succeed in
the copy.
If you do not use this file, you have to define all nodes to
```

customize in the SDR and network boot the nodes, in such a way that install servers are customized first and their dependencies after that. The other method is to put the above mentioned script (update.SrvKey) in the /etc/inittab file, so that customizing is not necessary.

```
#!/bin/ksh
#
# Remove old references to this script first
#
rm -f /tmp/update.srvtabs*
#
SSPDIR=/usr/lpp/ssp/bin
TMPDIR=/tftpboot
RCPCMD=/usr/bin/rcp
KERDIR=/usr/kerberos/etc
RCMDIR=/usr/lpp/ssp/rcmd/bin

# It may not work, but give it a try...
$KERDIR/kdb_util dump /tmp/Kerby.srvdump$$

echo
echo "Dumped contents of Kerberos database in /tmp/Kerby.srvdump$$"
echo

# Destroy corrupted database

$KERDIR/kdb_destroy

rm -f /.k
rm -f ~root/.klogin

sleep 2

# Stop Kerberos activity and stop the Systems Monitor
# to make sure nothing of the old configuration is there.

chitab "kadm:2:off:/usr/lpp/ssp/kerberos/etc/kadmind -n"
chitab "kerb:2:off:/usr/lpp/ssp/kerberos/etc/kerberos"
chitab "hmon:2:off:/usr/lpp/ssp/bin/hmon.start"
telinit 2

# Kill the hardmon

KILLID=ps -ef | grep hardmon | grep -v grep \
         | awk '{print $2}' > /dev/null 2>&1
kill -15 $KILLID > /dev/null 2>&1

# Get all hosts from SDR.

HOSTS=$SSPDIR/SDRGetObjects Node initial_hostname \
     | awk '{FS="."};{print $1}' | grep -v initial

# Get all node numbers from SDR, for spbootins.
NODES=$SSPDIR/SDRGetObjects Node node_number \
     | grep -v node_number | awk '{print $1}'

# Need to change spaces by comma's between nodenumbers...

NODELIST=echo $NODES | sed 's/ /,/g'
```

```
# Invoke setup_authent to recreate Kerberos database

setup_authent

# Invoke setup_server to complete Kerberos tasks, on all install
# servers

echo ″Executing setup_server, this may take several minutes...″
$SSPDIR/spbootins -r customize -l $NODELIST >/dev/null 2>&1

# Configure the Systems Monitor to respawn.

chitab ″hmon:2:respawn:/usr/lpp/ssp/bin/hmon.start″
telinit 2
# Add rcmd service for CW to .klogin file.

REALM=head -1 /etc/krb.conf
KERSERVER=cat /etc/krb.conf | grep admin | awk ′{print $2}′

# Need shortname...
KERADMSER=echo $KERSERVER | awk ′{FS=″.″};{print
$1}′

echo ″rcmd.″$KERADMSER″@″$REALM >> ˜root/.klogin

# Execute script to distribute srvtab files, need to do this
# outside this script, because it replicates itself.
# I assumed this script to be stored in /usr/local/bin.
# The script is called distr.srvtab, and is provided as
# an example following this script.

/usr/local/bin/distr.srvtab

echo ″Executing setup_server again to set bootp_response back to disk″
$SSPDIR/spbootins -r disk -l $NODELIST
exit
```

The following script, in this example assumed in /usr/local/bin, is first executed on the Control Workstation, where it searches the /tftpboot directory for new-hostname-srvtab entries. If it finds any, the script will copy the respective files to the corresponding (hostname) host/node. This script will then copy itself to the same node to also search the remote host′s /tftpboot directory: this node may be a install-server. If the remote host is an install-server, the script will then copy the new-hostname-srvtab entries on the install-server to the corresponding hosts served by that install-server. After that the script will copy itself to the nodes served by that install-server and so on down the line. If the remote host is not an install-server, the script exits and will continue with the next host served by the install-server.

Note: As you will see, this script relies on the existence of an .rhosts file or entries in the /etc/hosts.equiv file. If Kerberos is corrupted, the Kerberized commands cannot be used. The way to circumvent this is to define all nodes to customize in the SDR and network boot them all in such order that first all install-servers are network booted, and then all other nodes.

```
#!/bin/ksh
cp $0 /tmp/distr.srvtab
NAME=/tmp/distr.srvtab
SRVTAB=/etc/krb-srvtab

cd /tftpboot
if [[ -z ls -1 /tftpboot | grep srvtab ]]
then
   echo "hostname not an install-server"
else
   for i in ls -1 /tftpboot | grep srvtab
   do
# Look for nodes to receive srvtab files
     NODE=echo $i | awk '{FS="-"};{print $1}'
# Copy srvtab file to node
     /usr/bin/rcp /tftpboot/$i $NODE:$SRVTAB
# Change ownership and mode
     /usr/bin/rsh $NODE chmod 400 $SRVTAB
     /usr/bin/rsh $NODE chown root.system $SRVTAB
# Copy THIS script to next node and change mode
     /usr/bin/rcp $NAME $NODE:/tmp
     /usr/bin/rsh $NODE chmod 700 $NAME
# Execute this script on following node.
     /usr/bin/rsh $NODE $NAME
   done
fi
```

4.5 Kerberos Restrictions

This section describes the known restrictions of Kerberos as implemented on the SP. It is meant as a short overview.

1. Ticket life only 21.25 hours

   The reason for this limitation is that only eight bits were used for the expiration time, with only five minutes for each tick. This results in 255 times 5 or 1275 minutes (21.25 hours).

2. Command rcmdtgt only available for the root user

   For running remote commands continuously, only the root user can use this command to non-interactively create a new ticket. Due to the permissions on the /etc/krb-srvtab file, only root can access it. There is no reason why rcmdtgt should not work for normal users. Perhaps ACLs could provide a solution. Unfortunately, our team did not have sufficient time to investigate this further.

3. No Kerberized network login facility

   As discussed earlier in this chapter, the rsh command is restricted for use without argument (that is only the Kerberos version). Technically, there is no reason why this cannot be done. We think this might be caused by the restictiveness of the rcmdtgt command.

   Note: In the Public Domain, there are Kerberized versions of telnet available.

4. Limited Kerberos recovery possibilities, other than those described in this chapter.

5. The current version of Kerberos, as described and tested in this book, only supports one network interface as a target for the Kerberized commands. However, there is a PSSP APAR available that enables support for multiple interfaces. Refer to Appendix B, “New Features For ssp.clients” on page 237 for more details.

Chapter 5. The Network Time Protocol (NTP)

The Network Time Protocol is a time-service protocol that attempts to synchronize clocks with other known clocks. This synchronization occurs in a hierarchical, master-slave configuration, as shown in Figure 11.

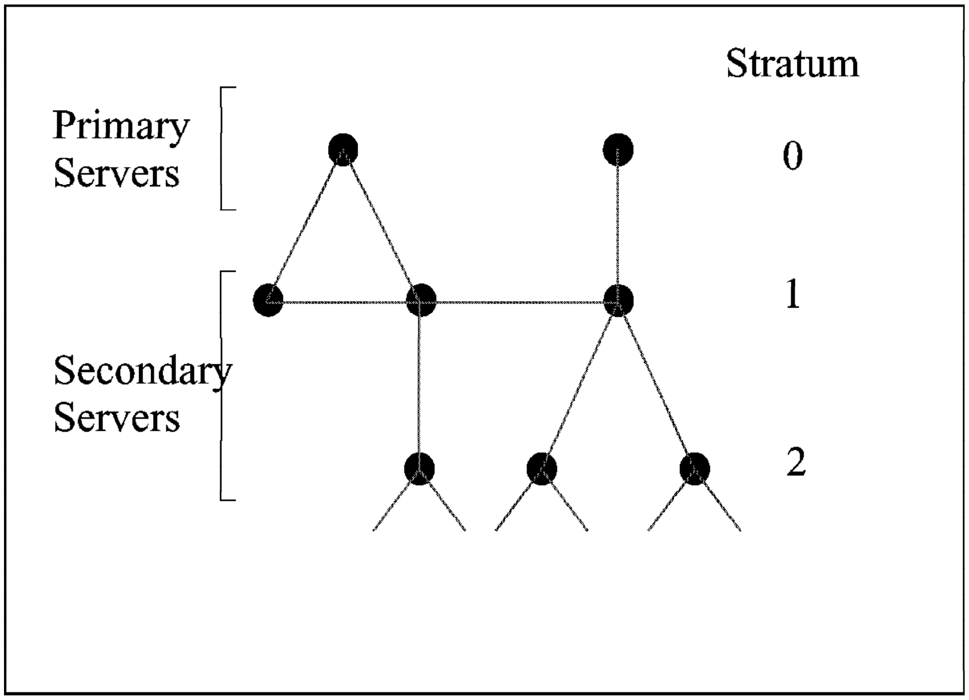


Figure 11. Sample NTP Hierarchy

At the top level of the hierarchy are the primary servers. Primary servers maintain their own, presumably highly-accurate, clocks. At lower levels of the hierarchy are secondary servers, which synchronize to the primary servers, possibly via other secondary servers. The various levels of the hierarchy are referred to as the stratum. The stratum of a server tells exactly how many synchronization paths lie between a secondary and a primary server. The larger the stratum number, the more hops exist to the main clock reference, therefore, the smaller the degree of accuracy.

Secondary servers have identical function. They can operate simultaneously as both clients of lower-stratum servers and servers to higher-stratum servers. Configuration information determines which hosts a server may take time from or offer time to. When a host is configured to have multiple time servers, a selection algorithm attempts to determine the most accurate and reliable server with which to synchronize. This is a dynamic process in order to overcome such problems as network drops.

NTP aims at maintaining accurate synchronization, even in the face of factors such as network transmission delays and heterogeneous computing resources. NTP does this by synchronizing both the time of day and the tick frequency. The accuracy of the synchronization is directly affected by two factors: the stratum and the duration of synchronization. The duration of synchronization refers to the number of measurements used to adequately resolve the skew, or frequency difference, among clocks. The more measurements used, the more accurate the clocks will be.

UDP/IP messages are used to transfer information between servers. So, any IP-capable network can be used as the network medium. In fact, host configuration is based on IP addresses.

5.1 NTP on the RS/6000 SP

The SP nodes do not have system batteries. So, some mechanism must be chosen to synchronize their clocks upon system startup. NTP is the default mechanism. If you do not choose NTP, you must provide another time service, such as the `timed` daemon or `Digital Time Service`.

To see how or whether NTP is configured on the SP, on the Control Workstation or any node enter:

```
# splstdata -e | grep ntp_config
```

If this command returns a value, then NTP is installed and configured. To check if NTP is running on the SP, enter the following from the Control Workstation or any SP node:

```
# ps -ef | grep ntp
```

If the command returns `/usr/lpp/ssp/bin/xntpd`, then NTP is running.

5.1.1 Standard Configuration Options

Four standard options for using NTP are provided in the SP configuration software, corresponding to four different values for `ntp_config`:

consensus	Run NTP locally on the RS/6000 SP to generate a consensus time. This configuration uses the Control Workstation as the time master.
timemaster	Use your site′s existing NTP time server to synchronize the RS/6000 SP system clocks.
internet	Use an NTP time service from the Internet to synchronize the SP system clocks.
none	Do not use NTP on the SP.

If configured to use NTP in one of the standard setups shown above, the file `/etc/ntp.conf` will reside on the Control Workstation and all nodes. Note that the file may not necessarily be identical on all nodes (see Table 9 on page 61). One or more lines within this file may begin with the word `server` or `peer`. *Servers* are hosts from which this host can request time. *Peers* are hosts which this host can give time to or request time from. The selection algorithm, which puts all hosts into a minimum-weight spanning tree, determines whether a peer gives time or requests time. Consult the manpages for other options that can be specified in configuration file.

Note the server entry on the Control Workstation when operating in consensus mode. IP address 127.127.1.10 is, of course, not a valid machine address. NTP uses a set of reserved IP addresses to denote special meanings. Specifically, an address of the form 127.127.1.u tells the NTP daemon to synchronize to a local clock and to run the server at the stratum designated by *u*.

Table 9. SP NTP Configurations

ntp_config	ntp_server	Control Workstation′s ntp.conf	Boot/Install Servers′ ntp.conf	Processor Nodes′ ntp.conf
timemaster	hostname of your current NTP server	SERVER: your time server PEER: all boot/install servers	SERVER: your time server PEER: Control Workstation PEER: all boot file servers	SERVER: your time server SERVER: Control Workstation SERVER: all boot file servers
internet	hostnames of the time servers on the Internet	SERVER: Internet servers	SERVER: Control Workstation PEER: other boot file servers	SERVER: Control Workstation SERVER: all boot file servers
consensus		SERVER: 127.127.1.10	SERVER: Control Workstation PEER: other boot file servers	SERVER: Control Workstation SERVER: all boot file servers
none	″″	no configuration	no configuration	no configuration

Note: NTP synchronizes time across time zones. It is up to the system administrator to ensure that all nodes are operating in the same time zone.

5.1.2 Runtime Administration

A program called xntpdc allow users to query and control the NTP daemon. With this program, most startup configuration options can be changed at runtime. Also, state and statistical information can be gathered.

For example, to print a list of the peers known to a host, in this case node sp2sa02, and some state information, enter:

```
# xntpdc -c peers

     remote           local      st poll reach  delay   offset   disp
=====================================================================
*sp2eacw        155.56.169.11   11  128  377  0.0014 -0.000194 0.0024
```

This output tells us that the node is currently synchronized to the host sp2eacw, which happens to be the Control Workstation. Also indicated are the local interface, the stratum of the remote peer, the polling interval (in seconds), the reachability register (in octal) and the current estimated delay, offset and dispersion of the peer (all in seconds). The last three values are the products of NTP: the delay is the roundtrip delay to the remote host, the offset is the difference between the two clocks and the dispersion is the maximum error of the local clock relative to the reference clock. Consult the manpages for more detailed information. (See 5.2.2, “Installing the Manpages” on page 62 and 5.2.3, “Installing Additional NTP Commands” on page 62 for instructions on how to install NTP manpages and commands).

Note: The default permissions on xntpdc are set to 755 and, so, allow normal users to control the NTP daemon, possibly altering established operating parameters. If this is not desired, reset the permissions; for example:

```
# chmod 700 /usr/bin/xntpdc
```

This assumes that you have installed `xntpdc` in `/usr/bin` as indicated in 5.2.3, “Installing Additional NTP Commands” on page 62.

5.2 Hints and Tips

In this section, we present our experiences with NTP and what we think may help you in making its administration easier.

5.2.1 Setting Up NTP

When you set up NTP on the Control Workstation, either through the SMIT interface or the command line (spsitenv), you are, in fact, setting up NTP for the entire SP “complex” (that is, the Control Workstation included). If, for example, you select *timemaster* in the **NTP Installation** field and define an NTP server, that server is the server for the processor nodes as well as for the Control Workstation (refer to Table 9 on page 61). So, it does not make a lot of sense to specify the Control Workstation or any of the SP nodes as an NTP server. (In fact, should you try to do so, you will get a message saying that it is not allowed to specify the Control Workstation or a node as an NTP server). You must select a host that is external to the SP complex.

Using the SMIT interface, you can specify more than one server in the field NTP Server Hostname(s). Simply separate the hostnames by a comma with no space in between. It is recommended that you specify three servers reachable via different networks to ensure that one is always available.

5.2.2 Installing the Manpages

The installation of PSSP does not automatically install the manpages of many software components, NTP included. To install the NTP manpages:

1. `cd /usr/lpp/ssp/public`
2. Extract the manpages:

   ```
   zcat ntp.tar.Z | tar -xvf - ntp/doc/xntpdc.8
   zcat ntp.tar.Z | tar -xvf - ntp/doc/xntpd.8
   ```

3. Install the manpages:

   ```
   install -c /usr/man/man8 ntp/doc/xntpdc.8
   install -c /usr/man/man8 ntp/doc/xntpd.8
   ```

Now you are able to get the manpage for xntpdc or xntpd by entering, for example: **man xntpdc**.

5.2.3 Installing Additional NTP Commands

All of the NTP commands needed for SP operation have been installed. However, there are other commands that may help to better manage NTP. To install these commands:

1. `cd /usr/lpp/ssp/public`
2. Extract the source: `zcat ntp.tar.Z | tar -xvf -`
3. `cd xntpdc`
4. Compile the program: **make**
5. Install the program: `install -c /usr/bin xntpdc`

Now you can issue NTP queries through **xntpdc**. However, set the permission, as indicated in 5.1.2, “Runtime Administration” on page 61.

Chapter 6. The System Data Repository

The System Data Repository (SDR) is an SP subsystem that stores SP configuration and some operational information. The SDR stores:

- Information about the frames, nodes and switches and how they are configured
- Job Manager operational data
- VSD configuration information
- The current values of host_responds and switch_responds, two indicators of the health of a node

This information is stored on the Control Workstation, but is made available through a client/server interface to other network-connected nodes.

6.1 SDR Data Model

The SDR logical model consists of classes, objects and attributes. Classes contain objects. Objects are made up of attributes. Objects have no unique ID, but their combined attributes must be unique among other objects. Attributes can be one of three types: strings, floating-point numbers or integers.

Most SDR interaction is performed using the SDR command-line interface. Some clients, like Job Manager, communicate directly with the SDR.

6.2 Setup

During installation, the install_cw script is run, causing the following to occur:

1. /etc/inittab entries are added for the SDR daemon (sdrd).
2. Port definitions are added to /etc/services for sdrd.
3. The SDR_dest_info file is built.
4. **telinit q** is run, prompting /etc/inittab to be read again.

Invoking **telinit q** causes the SDR daemon to initialize. The SDR daemon runs the script, SDR_init, which creates the classes necessary for the SP. SDR_init also creates some default objects, like the SP object.

In addition, **telinit q** causes the hardware monitor daemon (hardmon) and the SP logging daemon (splogd) to start. The hardmon daemon will not begin reporting hardware status until **spframes** is run. At that point, hardmon “discovers” SP hardware from the serial connections to the frame controller and reports to splogd. The splogd daemon calls SDR_config, which adds new hardware (frames, nodes and switches) to the SDR.

6.3 Locking

The following sections discuss the SDR access rules.

6.3.1 Authorization

SDR permits only two types of authorization, read-only and read-write. Read-write authority is granted only if the following are *both* true:

- The user is running as root.
- The user is running from the Control Workstation or from a node whose adapter is in the SDR Adapter class.

If both of the above are not true, the user gets read-only permission.

Sometimes the above two conditions are met, yet an SDR command returns the message:

```
0025-001 A read lock has been obtained.
Operations that create or change data are not allowed.
```

This is usually caused by a routing problem on an SP node. The route from the SP nodes to the Control Workstation should be set up such that they go over the SP Ethernet. If an SDR client request is tagged with the address of an adapter that is not in the SDR Adapter class, the SDR assumes that the client is not on an SP node.

6.3.2 Holding Locks

SDR clients (Job Manager, the SDR command-line routines and so on) can lock an entire class to get exclusive write access. While the client holds the lock, other clients can read the data as it was before the lock, but cannot write to the class. If the client that holds the lock fails to unlock the class before it exits, the server releases the lock without making any changes to the class. The only cases where a class may be locked, yet the lock is not released are when:

- The client that has the class locked is stuck in an infinite loop.
- The node on which the client is running crashes or is powered down. The SDR server is not notified of the broken connection until the connection times out. This occurs after the time indicated in **tcp_keepidle**. The value of **tcp_keepidle** can be checked using the **no -a** command on the Control Workstation. Time is designated in half-second increments. The default is 14400, or two hours.

Only when an administrator is sure that one of the above has happened should he use the **SDRClearLock** command-line routine.

The **SDRWhoHasLock** command can be used to determine if a class is locked and by whom. The return value (if locked) contains a transaction ID of the format: <hostname>:<pid>:<session_id>, where <hostname> is the name of the node the client holding the lock is on and <pid> is the client′s process ID on that node. The <session_id> is not significant.

With the SDR code on the SP, deadlock cannot occur. The clients either write a class exclusively (only Job Manager writes the Job Manager classes, for example) or the client uses the command-line interface. It is impossible to cause deadlock with the command-line SDR routines because they only hold a lock (those that change data) for the duration of the call. So, no process can hold a lock using the command-line interface while also requesting another lock.

6.4 Shadow Files

The SDR stores class data in the /var/sdr directory on the Control Workstation. One file exists in that directory for each class. The name of a file that represents a class has the same name as that class. Though in ASCII format, these files should not be edited. Editing them may produce unpredictable behavior in the SDR.

When a class file is unlocked, the sdrd moves the current class to a file named <class>.shadow (where <class> is the name of the class being written). The modified class is then written from the daemon′s memory to <class>. When the daemon has successfully written the new class definition to <class>, the <class>.shadow file is erased.

If the SDR daemon is killed or the Control Workstation is powered down abruptly, the sequence detailed above may be interrupted. This will be obvious because of the presence of a <class>.shadow file. If one of these files is discovered, the system administrator should remove the file named <class> and move the file named <class>.shadow into <class>. The SDR daemon should then be killed. The daemon will restart automatically and will recognize the new <class> file. The only exception to this is if the administrator is sure that the contents of the original <class> file are intact. In this case, the <class>.shadow file can be removed.

Often, this situation will be discovered because a call to retrieve information from an SDR class fails. The message may indicate either that the class does not exist or that the class is corrupted. Messages of this type should prompt the search for the <class>.shadow files.

6.5 Client/Server Communication

SDR clients communicate with the SDR daemon using TCP/IP. If an SDR client is started on a node, it finds the name of the Control Workstation from either the SP_NAME environment variable or from the *primary* record of the /etc/SDR_dest_info file. The client tries to connect to that address on either the port named *sdr* in the /etc/services file or, if the *sdr* entry is not in /etc/services, by using port 5712.

If an SDR client exits abnormally, tcp notifies the SDR daemon. The daemon removes any locks held by the client. Any changes made to classes that the client had locked at the time of the crash are not saved. The integrity of the SDR data is at the class level. If the client saved a class, then the whole class was saved. If the client was in the middle of updating a class at the time of the crash, then all changes up to that point are lost.

If the node on which the client is running goes down, the above paragraph still applies, except that the tcp notification is delayed by the connection timeout value. (Check the **tcp_keepidle** value using the **no -a** command). If the Control Workstation goes down, all locks are lost. If a class was locked, the data in the class reverts back to the state it was in before the class was locked.

6.6 The host_responds and switch_responds Classes

A very important class in the SDR is the host_responds class. It has two attributes, *node_number* and *host_responds*. If the host_responds attribute for a node is “1,” the node is operational. If it is “0,” the node is down.

The host_responds attributes are updated by the hr daemon. The hr daemon connects to the heartbeat daemon (named ccst) on the Control Workstation. When the heartbeat daemon notifies hr that a node has come up or gone down, hr changes the appropriate attribute in the SDR.

The *switch_responds* class is updated when a node′s HPS adapter no longer responds.

The host_responds and switch_responds attributes are used by the Job Manager to determine if jobs can be submitted to a particular node. These attributes are also reported in the System Monitor.

6.7 Backing Up The SDR

The proper way to back up the SDR is with the provided utility, **SDRArchive**. Following is an example of its use:

```
# mkdir /tmp/sdr.backup
# SDRArchive /tmp/sdr.backup
```

This puts a file, `SDRArchive.tar`, into the directory /tmp/sdr.backup.

Use the **SDRRestore** command to restore the backup, as in the following:

```
# cd /tmp/sdr.backup
# SDRRestore SDRArchive.tar
```

The SDR daemon (sdrd) should be killed at this time, allowing it, when it restarts, to recognize the new files.

Note: The directory containing the SDR archive to be restored should not include a file named **SDR_Archive_temp**.

Partial archives can be done on a class-by-class basis. Class files can be copied out of the /var/sdr directory and stored elsewhere. To restore them, simply copy them back into /var/sdr, replacing files of the same name. Then, kill the SDR daemon (sdrd) and allow it to respawn.

Warning

Partial restore is not recommended. There are dependencies between classes in the SDR; those classes must be archived and restored together. Partially restoring the SDR can produce unpredictable results.

The system administrator should back up the SDR before doing any tasks that reconfigure frames, nodes, switches or VSDs. In addition, the SDR should be backed up periodically, such as once a week. The SDR should only be restored if it is known to be corrupted. If a reconfiguration command does not work properly or is interrupted, it is usually sufficient to rerun the command.

6.8 The SDR Log File

Serious SDR problems are reported in /var/adm/SPlogs/sdr/sdrdlog.<pid>, where <pid> is the process ID of the SDR daemon. Often, this log contains little more than messages showing a return code of 110, indicating that a client exited without closing its session with the SDR. These messages have no adverse effect on the SDR daemon and can be ignored.

Figure 12 shows a sample log file containing messages with the return code of 110.

```
Invocation: /usr/lpp/ssp/bin/sdrd
  pid  = 4759
  time = Thu Feb 23 17:45:06 1995

Client sp21cw0:23456:2 disconnected.  RC=110 from SDRRead (line 416).  Errno=0
Client sp21cw0:96732:2 disconnected.  RC=110 from SDRRead (line 416).  Errno=0
Client sp21cw0:19853:2 disconnected.  RC=110 from SDRRead (line 416).  Errno=0
```

Figure 12. Sample SDR Log File

There may be many of these SDR log files. A new one is created every time a new SDR daemon is invoked. Once the contents are checked, the files can be either archived or discarded, *except* for the file that corresponds to the daemon that is currently running. To identify the current log file, find the process ID of the SDR daemon using **ps -aef | grep sdrd**. The current log file has that process ID as its extension.

6.9 Dependencies on the SDR

The following subsystems rely on the SDR:

- The Job Manager.
- VSD configuration tasks.
- SP system configuration, customization and install tasks.
- The hardware monitor clients (spmon, hmmon, hmcmd, s1term and nodecond).
- The hardware monitor daemon (hardmon), only when invoked. Once running, SDR access in not required.
- The SP nodes, when they are booted.
- The High Performance Switch, when the switch is started.

Chapter 7. The High-Performance Switch

The High-Performance Switch (HPS), an optional feature, tightly couples the SP nodes. It is HPS that makes the SP into a parallel machine.

The SP is, in fact, a general-purpose machine and can be used for consolidating servers on a LAN, as well. In that mode, the LAN can be a token ring, Ethernet or FDDI network; there is no need for HPS. But, if the requirements are for fast communication between nodes and large bandwidth to move large amounts of data, as in parallel databases, HPS is the ideal network because it is scalable. The HPS component described in this section is the one supported by PSSP 1.2.

7.1 HPS Hardware

The HPS hardware components include:

- The switch assembly, containing switch chips (or ports)
- Adapters cards, located in the nodes
- Internal and external data cables (for multi-frame)
- Internal clock cables
- Clock interposers (for multi-frame)
- Power cable
- Wrap plugs

Note: Internal clock cables and clock interposers may not be required because they exist only on early SP models.

In his daily work, it is very unlikely that the SP system administrator will have to deal with the switch hardware. However, it is useful to know the meaning of the switch LEDs located on the front of each switch assembly.

Yellow LED status	**Meaning**
Off	Supervisor card detects no environmental problem.
On	Supervisor card detects a problem. Schedule the switch for service.
Flashing	Supervisor card detects a serious problem; power to the switch is shut off.

Green LED status	**Meaning**
Off	No power.
On	Power is on, but HPS is not running.
Flashing	Normal status.

7.2 Configuring HPS

HPS is configured at installation time. Configuration files are provided for this purpose.

7.2.1 HPS Configuration Files

The configuration files for HPS are located in the /etc/SP directory on the Control Workstation. The content of that directory is shown in Figure 13.

```
[root@sp21cw0] / >ls /etc/SP/expected.*
Eclock.top.1nsb.0isb.0       expected.top.1nsb.0isb.0
Eclock.top.1nsb_8.0isb.0     expected.top.1nsb_8.0isb.0
Eclock.top.2nsb.0isb.0       expected.top.2nsb.0isb.0
Eclock.top.3nsb.0isb.0       expected.top.3nsb.0isb.0
Eclock.top.4nsb.0isb.0       expected.top.4nsb.0isb.0
Eclock.top.4nsb.2isb.0       expected.top.5nsb.0isb.0
Eclock.top.5nsb.0isb.0       expected.top.5nsb.4isb.0
Eclock.top.5nsb.4isb.0       expected.top.6nsb.4isb.0
Eclock.top.6nsb.4isb.0       expected.top.7nsb.4isb.0
Eclock.top.7nsb.4isb.0       expected.top.8nsb.4isb.0
Eclock.top.8nsb.4isb.0
```

Figure 13. Contents of /etc/SP Directory

The naming convention for those files is of the format:

`expected.top.XXnsb.YYisb.Z`

where:

XX is the number of node switch boards (NSB).

YY is the number of intermediate switch boards (ISB). For up to 80 nodes, the number of ISB is 0. ISBs exists only in type-IV fr   ames.

Z is the topology type, typically 0 (the default).

The content of expected.top.XXnsb.YYisb.Z may look like that shown in Figure 14.

```
####################################################################
#
#   Initial version - 7 July 94
#
#   NOTE : This is a 8-port HPS switch topology file. It requires
#          a "aux 8-port" entry at the end of the file. So that
#          proper mis-wire detection can be done.
#
####################################################################
#
format 1
8 11
# Node connections in frame L01 to switch 1 in L01
s 10 4  tb2 0 0     L01-S00-BH-J18 to L01-NXX
s 10 5  tb2 1 0     L01-S00-BH-J16 to L01-NXX
s 10 3  tb2 2 0     L01-S00-BH-J20 to L01-NXX
s 10 2  tb2 3 0     L01-S00-BH-J22 to L01-NXX
s 10 6  tb2 4 0     L01-S00-BH-J14 to L01-NXX
s 10 7  tb2 5 0     L01-S00-BH-J12 to L01-NXX
  .. .  .......     .........................
```

Figure 14. Content of Topology File /etc/SP/expected.top.1nsb_8.0isb.0

Aside from the numbers 8 and 11, the rest of the file is self-explanatory. Eight and 11 represent two bounds in terms of nodes and switch chips.

Note: It is recommended that the system administrator not alter the content of these configuration files. Having said that, we are about to suggest an instance where altering these files may be beneficial (that is, for annotatation of topology files and when grouping switches within HPS).

Annotating a topology file is not necessary, but it helps when reading switch diagnostics. The customer engineer will thank you and may be able to fix your HPS a bit faster. Annotating a topology file adds the physical locations to the connection labels shown in Figure 14 on page 70. To annotate a topology file, you can either use SMIT or the command `Eannotator`. The format of the command is as follows:

```
Eannotator -F input_file -f output_file -O yes -I no
```

where:

> `input_file` is the name of the topology file to be annotated.
>
> `output_file` is the annotated file.
>
> `-O yes` if you want to store this file in the SDR.
>
> `-I no` if you do not want to retrieve the input file from the SDR.

Note: Eannotator generates connection data only for the nodes that are defined in the SDR. These are actual nodes that exist in the frame and are “discovered” by *hardmon* during installation after `spframe` is run. This also explains why topology files are not shipped pre-annotated: the physical locations are not known until after installation.

Figure 15 shows the annotated version of the file presented in Figure 14 on page 70.

```
#
####################################################################
#
#   Initial version - 7 July 94
#
#   NOTE : This is a 8-port HPS switch topology file. It requires
#          a "aux 8-port" entry at the end of the file. So that
#          proper mis-wire detection can be done.
#
####################################################################
#
format 1
8 11
# Node connections in frame L01 to switch 1 in L01
s 10 4  tb2 0 0           E01-S17-BH-SC to E01-N1
s 10 5  tb2 1 0           E01-S17-BH-SC to E01-N2
s 10 3  tb2 2 0           E01-S17-BH-J27 to E01-N3
s 10 2  tb2 3 0           E01-S17-BH-J29 to E01-N4
s 10 6  tb2 4 0           E01-S17-BH-SC to E01-N5
s 10 7  tb2 5 0           E01-S17-BH-SC to E01-N6
s 10 1  tb2 6 0           E01-S17-BH-J31 to E01-N7
 ..................................................
```

Figure 15. An Annotated Topology File

The directory `/etc/SP` also contains *clock topology* files. These are used to set up the clock distribution. With a single switch board or SP frame, the clocking signal is obviously the same for the entire configuration. Eclock is not used in that case. But, with multiple switch boards and frames, the clocking signal must be distributed over the entire configuration. The clock topology files provide information to the Eclock command with which to complete the process.

The clock setting can be checked with *spmon*. For a one-frame SP machine, the only choice is `internal clock`.

Warning

When you add a new frame to your configuration, you must select a new topology file if a new switch board is also added. Otherwise, HPS will fail.

7.2.2 Selecting and Storing the HPS Configuration Files

Configuration files exist for all possible configurations of the SP. You need only select the one that is appropriate for your situation. In our center, the SP has 12 nodes, eight thin ones and four wide. So, the configuration we should select is the one with a one-node switch board and no intermediate boards, that is `expected.top.1nsb.0isb.0`. This file should be stored in the SDR because the switch initialization code searches in the SDR for it when executing the `Estart` command. To store this file in the SDR, use the `Etopology` command:

```
[root@sp21cw0] / >Etopology /etc/SP/expected.1nsb.0isb.0
```

You can verify that the topology file is stored in the SDR by displaying the SDR content for HPS with `splstdata -s`. Ours is shown in Figure 16.

```
switch_part          topology          primary        arp  switch_node
     number          filename             name    enabled   nos._used
-----------------------------------------------------------------------
          1   SDR.expected.to  sp21n01.itsc.pok        yes          no
```

Figure 16. Partial Output Showing SDR Content for HPS

If there is a need for it, the stored configuration can be overridden. This can be done by storing a topology file in the `/etc/SP` of the primary node.

Note: You should run Eannotator against your topology file before storing it in the SDR.

7.2.3 HPS Primary Node

Although routing information for the HPS adapter exists in every node of the SP, HPS logic requires that a specific node be designated as the repository for that information. This node, the primary node, is indicated using the command:

```
[root@sp21cw0] / >Eprimary node_number
```

where node_number is the node number of a processor. If you use the number 5, for example, the command tells the switch logic that routing information is to be found in node 5.

The primary node initializes the entire switch network and recovers when errors, reported as “switch faults,” are detected. With the information on switch connections, stored in the topology file, the primary node dynamically generates routes from the actual connections. So, using `Eprimary` to change the primary node without actually doing an `Estart` only passes the “title” to the new node.

The existence of only one primary node without a backup results in a single point of failure for HPS. Should node 5 fail, HPS will fail and a new primary must be selected. This impact can be lessened by setting up switch groups.

7.2.4 Grouping Nodes within HPS

By default, there is only one node group for the entire HPS. It is, however, understandable that this might not be the ideal configuration in many cases. For instance, suppose a company receives a four-frame SP machine defined with only one HPS, intending it for multiple uses. One frame may run a parallel database, another one may be used for LAN consolidation, the remaining ones for production departmental work and testing. Each time the test people manage to create a switch fault, the database users and other users doing production work will experience an interruption in services. Grouping nodes within HPS is a way of localizing the faults. The physical connections between each group of nodes are actually removed by hiding them in the topology file. A node in one group cannot communicate with a node in a different group. So, in the previous case, you might want to divide the switch into three groups so that the testers would be the only ones enjoying the consequences of their mistakes.

In our project, we created three groups for our 12 nodes. To do this, we had to modify the expected.top.1nsb.0isb.0 file (the second instance where altering a configuration file might be beneficial). (Future releases of PSSP will support HPS partitioning, so you will not have to do this manually). We created three topology files, one for each group, by carefully pruning expected.top.1nsb.0isb.0. This is what we previously referred to as “hiding” the physical connections; they are not actually removed.

The nodes that share the same topology file form a group or partition on the HPS. Each partition has a primary node and is independent of the other partition. So, each of them must be started separately using Estart.

Note: Although the switchresponds panel does not show the three groups, if one fails, you would see the fault is localized to only one partition.

7.2.5 Switch IP Address

In PSSP 1.1, switch IP addresses were related to switch node numbers. This is because the Communication Subsystem did not support ARP (the Address Resolution Protocol). The switch network was defined as a subnet, giving the user only the choice of the starting node IP address. Subsequent addresses were created based on the switch node number. With ARP, the user is free to decide the IP address for all of the nodes.

7.3 Starting the HPS

The switch can be started with the Estart command from the Control Workstation or the primary node. The *Worm* must be running already on all SP nodes. Otherwise, the command would fail for that node. It will also fail for all nodes if Worm is not running on the primary. It is easy to see why (if the primary fails, the entire set fails).

```
    root 12404     1   0   Apr 05      -  0:51 /usr/lpp/ssp/css/fault_service_Wo
rm_RTG -r 12 -s 100014 -p 1 -a TB2 -d SW
```

Estart causes a switch fault. If the switch is already running, applications that use an unreliable protocol, such as UDP, will lose data. Although it does not use UDP, VSD is one such application. But VSD will recover through the sequence number scheme. Estart accepts the partition number as a parameter. If not specified, by default, it uses partition 1.

7.4 Resolving a Switch Problem

We take this opportunity to show how to “solve” a common switch problem. Occasionally, when you do an Estart command, nothing happens. The switch times out with a message like the following:

```
# Estart
Switch initialization started on sp2sw01.itsc.pok.ibm.com.
Estart:  0028-007 Switch initialization 180 second time limit exceeded.
```

And, if you look in the log, you see this message:

```
# cat /var/adm/SPlogs/css/out.top
  ⋮
s 17 3  tb2 15 0    L01-S00-BH-J34 to L01-N16 -16 R: device might not be driven
by the global oscillator (may be driven by internal clock)
  ⋮
```

The message gives the physical location of the problem. It tells you that the link from jack34 at the base switch assembly in logical frame 1 to switch adapter node 16 of the same frame did not initialize properly because of a clock-synchronization problem. The -16 shown at the end of the physical location is the error number. Figure 17 on page 75 shows the Diagnosis Guide′s explanation of the problem.

```
# errpt -a | more

ERROR LABEL:    HPS_FAULT6_ER
ERROR ID:       89F34F83

Date/Time:       Fri Nov 11 13:54:33
Sequence Number: 1803
Machine Id:      000028707500
Node Id:         sp2sw01
Class:           S
Type:            PERM
Resource Name:   Worm

Error Description
HPS Fault Service Daemon Terminated

Probable Causes
SYSTEM I/O BUS
HPS adapter failure
Switch clock signal missing

Failure Causes
SYSTEM I/O BUS
HPS switch cable
        Recommended Actions
        Run adapter diagnostics

Detail Data
Software ID String
LPP=PSSP,Fn=fs_daemon_init.c,SID=1.14,L#=527,
```

Figure 17. Error Text for HPS Fault

This indicates a clock problem. But, to make sure the hardware is not under the spell of some malicious spirit, run diag on node 16 to test HPS, as is recommended. As in our case, it is likely not a hardware problem. Just invoke the following from the Control Workstation:

```
[root@sp21cw0] / >sysctl -h sp2n16 exec /usr/lpp/ssp/css/css_restart_node
```

Then, reissue the Estart command. This time it should succeed and all nodes should initialize properly.

Part 3. Administrative and Operations Interface

Chapter 8. SP Remote Commands

This chapter discusses the commands that can be used to access nodes and remotely execute commands in an SP. Obviously, Kerberos will impact the way in which you can perform remote access. Kerberos must to be set up correctly and additional access permissions must be established at various levels. Refer to Chapter 4, "Kerberos" on page 39 for detailed information.

We cover the basic concepts of remote access related to AIX in general and, more specifically, the remote commands delivered with the PSSP package. Commands like **dsh** were made available with the initial release of the SP product line, so they are only discussed briefly. The main focus is on the latest additions to PSSP, in particular **sysctl**.

A short section covers the features of DSMIT. Although it is an unsupported product, we are of the opinion that DSMIT is a very suitable product for the SP.

8.1 Remote Execution

Remote commands under AIX enable users to log into a remote host or run commands on remote hosts, provided they have remote authorization. Nodes in the network that provide a service are considered servers; machines asking for service are clients. The AIX Operating System is the base of each SP node and, by default, AIX is installed with TCP/IP. TCP/IP provides commands such as **rsh**, **rlogin**, **rexec** and **ftp**. All of these commands can only run when the remote host provides the specific service configured in TCP/IP (/etc/inetd.conf and /etc/services). Each user has access to all of these commands, but can be granted or denied access on server machines. Traditionally, authorization can be set by configuring, for example, the **.rhosts** file or the **/etc/hosts.equiv** file. The authorization is based on machine names and/or usernames. Refer to InfoExplorer for more detailed information on how to configure these services and on other services TCP/IP provides for remote access and execution.

With the introduction of Kerberos, the authorization method of TCP/IP is no longer necessary. Actually, it is not recommended for a secure environment. The mechanism used by Kerberos to grant or deny access to services is accomplished by authentication rather then authorization. Kerberos provides an authentication service, but an authorization policy is still necessary. The **rcmds** commands use **.klogin** files, for example.

8.2 PSSP Remote Commands

The PSSP comes with a set of new, remote and parallel commands. The objective of the parallel commands is to execute commands on multiple hosts in parallel by specifying only one command. This allows you to manage SP nodes from a single point of control.

8.2.1 Kerberos Commands

Each SP ordered with PSSP Version 1.2 is delivered with the Kerberos Authentication services. The implementation on the SP provides three new remote, authenticated commands:

- **/usr/lpp/ssp/bin/rcmd/rsh**
- **/usr/lpp/ssp/bin/rcmd/rcp**
- **/usr/bin/sysctl**

This list may seem small, but all other remote commands on the SP are based on one of the commands listed. These commands are discussed later in this chapter; we point out on which of the above-listed commands they are based.

Note: Refer to Chapter 4, "Kerberos" on page 39 for more information on Kerberos.

8.3 Parallel Commands

The parallel commands shipped with the PSSP package are all scripts written in Perl and built on top of one of the "Kerberized" remote commands (rsh or rcp) or sysctl. If Kerberos is not functioning properly (ticket expired, database corrupted or user not part of Kerberos), the new rsh tries to execute the "normal" remote command, that is, the non-Kerberized version in /usr/bin. Successful execution then depends on permissions set by TCP/IP configuration files like **.rhosts**. For example, most parallel commands use dsh, which uses `/usr/lpp/ssp/rcmd/bin/rsh`. This section describes the various parallel commands, in particular:

- dsh
- pcp
- sysctl
- Other parallel commands

Note: Perl (Practical Extraction and Report Language) is a language that combines some of the features of C, sed, awk and shell. A Nutshell Handbook published by O′Reilly & Associates can provide additional information.

The picture shown in Figure 18 on page 81 might help you to understand the relationship of all of the commands a bit better.

8.3.1 The Dsh Command

Dsh (Distributed Shell) is the parallel implementation of the **rsh** command [executes the specified command at the remote host(s)]. The dsh command is build on top of the Kerberized rsh command; so, to use dsh, you should be an authenticated user to Kerberos, or you should work with the .rhosts file. The command specified with the dsh command is performed on all the nodes specified in the *working collective* (with an rsh on those nodes).

The working collective can be specified in the following ways:

- **WCOLL**: WCOLL is an environment variable that specifies the filename of the file containing the nodes with which you wish to work. As an example, say you want to work with the nodes in group_d. The file `/usr/local/groupd` contains a list of the nodenames in that group.

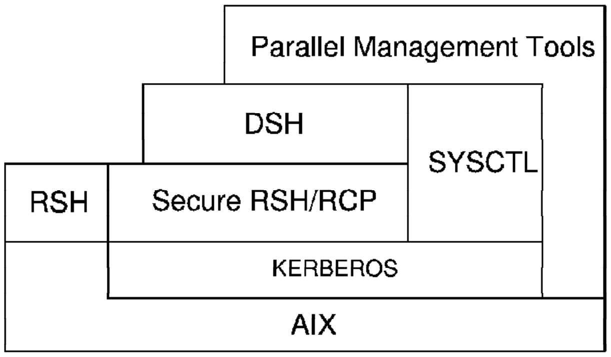


Figure 18. Remote and Parallel Commands

The WCOLL variable should be set to:

```
WCOLL=/usr/local/groupd
export WCOLL
```

Now, you can specify:

```
dsh -v <command>
```

- **-w flag**: Interactively, you can specify a number of nodes using the -w flag:

  ```
  dsh -vw sp21n01,sp21n02 <command>
  ```

- **All nodes**: You do not need to specify anything special when you choose to run a command on all of the nodes:

  ```
  dsh -av <command>
  ```

8.3.1.1 Using Dsh Without a Command

Dsh has an interactive mode. It runs multiple rshs on the local host to drive commands on remote hosts in parallel. You are prompted for each command. To do so, invoke dsh as follows:

```
dsh -av
```

This gives you a dsh prompt:

```
dsh>
```

From this prompt, you can now issue commands, which, in effect, will be executed on all of the nodes you have specified in the Working Collective. From the dsh prompt, you can now issue every command you wish except those that are full-screen driven, for example: `vi`, `pg` and `more`.

8.3.2 The Pcp Command

The **pcp** command is the parallel implementation of rcp (tranfers files between a local host and a remote host or between remote hosts). Because the pcp command is built on top of the Kerberized rcp command, it understands the normal rcp flags, like -r and -p. You must be an authenticated Kerberos user to use this command or implement the .rhosts file. The nodes affected by this command can be indicated using one of the following:

- **Stdin**: You can use standard input for passing nodenames to pcp, for example:

  ```
  hostlist -av | pcp -w - /etc/filesystems /etc
  ```

- **Hostlist flags**: You may specify the hostlist flags as arguments for pcp:

  ```
  pcp ″-av″ /etc/filesystems /etc
  ```

- **Working collective**: You may also use the **-w** flag, followed by the hosts to which you want to copy a specific file:

  ```
  pcp -w sp21n01,sp21n12 /tmp/config /usr/local/config
  ```

The pcp command uses the **pexscr** command (Parallel EXecute SCRipt) to run multiple rcp commands in parallel on the local node. Each rcp command copies the local file to a remote host. Pexscr uses stdin (only) for input of commands and nodenames. Pexscr is, in a way, similar to dsh, but is more general.

An example of how to use pexscr follows:

```
pexscr << ENDIT
sp21n07: command1
sp21n07: command2
sp21n08: command3
sp21n09: command4
ENDIT
```

All four commands run in parallel simultaneously; the first and second command on sp21n07, the third on sp21n08 and the fourth on sp21n09.

8.3.3 Other Parallel Commands

This section gives you an overview of all of the parallel commands provided by the PSSP package. The commands (shown in Table 10 on page 83) are listed under three columns: dsh, sysctl and other. The commands listed in the respective columns are either built on top of dsh, sysctl or another remote command, like rcp. All commands are truly executed in parallel, which is established by a commandloop, looping as many times as there are hosts to be served. In the loop, for all of the scripts, the remote command is forked. The sysctl commands are explained further in 8.3.5.1, “Sysctl Environment” on page 87. Please refer to the *IBM 9076 Scalable POWERparallel Systems SP2 Command and Technical Reference* for detailed information on the other commands.

8.3.4 Practical Experiences with Parallel Commands

As you can see, there are a number of commands available to perform all sorts of tasks in parallel. It is quite difficult to point out to you which commands to use for which tasks. This is dependent on what you are accustomed to using and, of course, how comfortable you are with using one of the new commands.

Table 10. Overview of the Parallel Commands

dsh	sysctl	other
p_cat pexec pfind pls pmv ppred pps prm	pdf pfck pfps	pcp pexscr

In most cases, you can use either the particular command or use a dsh session to perform the single task in parallel. For example:

```
hostlist -av | prm -w - /tmp/trash
```

instead of:

```
dsh -av rm /tmp/trash
```

There is no difference in the results either way you execute the command, because prm invokes dsh to perform the remote command. In some cases, there will be a slight difference in the command response, particularly in the case of command failure.

However, if you use the dsh command to remotely copy files from the Control Workstation, a problem may arise.

For example, say you want to copy the /etc/security/limits file from the Control Workstation to all nodes. For our example, we rely on the Kerberos permissions and not on the permission distributed through the .rhosts file. Working from the Control Workstation, you would normally execute a **kinit root.admin**. This allows you to use the Kerberized remote commands from the Control Workstation. A file must be copied to all nodes, thus “pushing” the file from the Control Workstation to all nodes, and cannot be done by “pulling” the file to the nodes from the Control Workstation. Following is an example of a command that “pulls” from the Control Workstation:

```
dsh -av rcp sp21cw0:/etc/security/limits /etc/security
```

This command would fail with:

```
sp21n09: rcmd protocol failure: 0041-004 Kerberos rcmd failed:
rshd: 0041-005 Kerberos rsh or rcp failed: %s
sp21n09.itsc.pok.ibm.com: trying normal rsh (/usr/bin/rcp)
sp21n09.itsc.pok.ibm.com: rshd: 0826-813 Permission is denied.
```

It fails because you have not issued a **kinit** or otherwise obtained a Kerberos ticket on all nodes. You are not allowed to remote copy/execute from the nodes.

The right way, in this example, to remote copy the file to the nodes is to use the **pcp** command, thus “pushing” the file from the Control Workstation to the nodes:

```
hostlist -av | pcp -w - /etc/security/limits /etc/security
```

Obviously, had you used **.rhosts** permissions, you would not have experienced the above error: rshd: 0826-813 Permission is denied. In this case, the only difference is that the Kerberized commands would not work.

Practical experience will determine which command to use in a given situation.

8.3.5 Sysctl

The *sysctl* utility is a client/server system for running tasks and commands on remote systems. The command is authenticated through Kerberos, so users of sysctl require Kerberos authentication, although sysctl allows non-authenticated users to have access to sysctl services, as well.

A sysctl environment is another method with which to allow system administrators to execute root commands on local and remote hosts without having to distribute the root password. Essentially, sysctl allows authenticated and authorized access throughout the SP system to servers running as userid 0. Authenticated users can run "sysctl commands" as root on those local and remote nodes where they are authorized. These commands are run in parallel on the target nodes.

Sysctl especially provides added value in the following areas:

Security: Sysctl is based on the Kerberos Authentication mechanism. Although not mandatory, generally a sysctl user must be a "known" Kerberos user before getting access to sysctl services. The default configuration of the SP does not allow normal users to access sysctl.

Authorization: As discussed earlier in this chapter, most of the authorization schemes are based on machine names and user IDs.

TCP/IP authorization is usually "all or nothing." You either give a user access to a particular machine or do not. Sysctl, however, provides a multilevel authorization mechanism for not only granting user access for sysctl use, but for providing authorization levels for commands or groups of commands, identifying which are executable or usable and which are not. Sysctl initially provides four levels of authorization discussed in 8.3.5.1, "Sysctl Environment" on page 87.

Existing Services: In general, system administrators have methods and procedures to perform day-to-day operations on their computers. These procedures exist to ease repetitive operations. It would be quite inconvenient to do these tasks interactively, command by command.

Sysctl is a new way of doing remote systems management. It would be very inconvenient if you needed to "reinvent the wheel" for sysctl. Being able to use what you already have would be ideal. Also, system-administrative tasks normally require high-level authority, usually root, to be able to execute them. Sysctl provides the ability to integrate your existing commands, scripts or programs into the sysctl environment without requiring a distributed root password. In order to integrate your existing commands into sysctl without recompiling them, you must write small procedures in the sysctl command language. This language is called Tcl or TclX (Extended Tcl) and is discussed in Appendix D, "README Files for Tcl and TclX" on page 251. Your sysctl commands can be simple wrappers around commands that normally require root authority or can be full-blown applications.

Authorization Levels: Sysctl provides a mechanism to grant access to sysctl commands at various levels. The default implementation provides four authorization levels, but you are able to create additional ones. The term for an authorization level in sysctl is *callback*. An authorization callback is the procedure that is called by the receiving sysctld daemon, which checks the authorization level assigned to a user against the required authorization defined for the requested service. The authorization levels or callbacks supplied with sysctl are:

- **NONE**: This means that the command always returns results. This is the lowest authorization level in sysctl.

  Note: In the default Sysctl configuration shipped with the PSSP package, no commands with NONE authorization can work, since you must be a Kerberos user (AUTH) to use sysctl. That can be changed, as is discussed in 8.3.6.3, "Adding Procedures to Sysctl" on page 95. So, when using sysctl as a non-authenticated user, you can only run commands/scripts that are authorized with NONE.

- **AUTH**: The user should be a Kerberos authenticated user to be able to execute the specified command. The typical error message when issuing an AUTH command without being authenticated is:

  ```
  sysctl:  2501-122 setauth: Insufficient Authorization.
  ```

  Provided that sysctld is able to resolve your authentication through Kerberos, you can run all commands with authorization level AUTH.

- **ACL**: The command will only run if the user (principal) is specified in an ACL file and, thus, is also an authenticated user. The default sysctl ACL file can be found in `/etc/sysctl.acl`. The error message for not being authorized is the same as that for AUTH.

  When connected to sysctl, if the user is authenticated, the user can only run those commands for which he or she is granted in any ACL file. Full authorization is provided by adding the user's principal to the global ACL file: `/etc/sysctl.acl`.

  Note: Be careful when adding users to the global ACL file, since it has the same effect as adding users to the root user's .klogin file. Any user (principal) mentioned in the global ACL file has **FULL** root authority on that node. This is because the Tcl "system" and "exec" commands, which allow the issuance of any AIX command or script, can be issued, as root, by users in this ACL file.

- **SYSTEM**: This type of command can only be used within the context of a predefined sysctl procedure. Such a command will fail to run if you try to run it from an interactive sysctl session, again with the error message:

  ```
  sysctl:  2501-122 setauth: Insufficient Authorization.
  ```

  Typically, these commands must be run from sysctl scripts. To prevent them from being executed in interactive sessions, you want functions and procedures developed for use in a script only to have SYSTEM authorization for these commands.

To implement an even greater access granularity, you can also define your own callbacks. An example is shown in 8.3.6.3, "Adding Procedures to Sysctl" on page 95.

Predefined Procedures: A procedure made up of sysctl and Tcl commands can be run on multiple nodes in parallel. All commands provided in sysctl can be executed from within a predefined procedure. As you will see later, some commands cannot be executed in interactive mode; they are only suitable for "SYSTEM" execution, that means, from within a predefined procedure. Consequently, these commands carry a default callback of SYSTEM. The reason for this protection is that some commands impact the entire sysctl environment, something you probably do not want to do in an interactive session. The default authorization callback can be changed, but this is not recommended for some commands.

Interactive: Although an environment with strict security no longer allows you to log into a node directly, sysctl provides a (single) line-mode interface to nodes (one at a time). This makes you feel you are logged into the connected host when, in fact, you are communicating (via RPC) to the remote sysctld daemon with a line interpreter.

Figure 19 on page 87 depicts, step by step, the activities triggered by invoking a sysctl session or sysctl command.

Let us go through the steps shown in Figure 19 on page 87. (We assume sysctl is configured as delivered; in other words, no changes in callbacks or authorization have been implemented):

1. Invoking the sysctl command prompts a call to the Kerberos database for user authentication. When authenticated, the user receives as many tickets for remote nodes as specified in the sysctl session (in the picture, only one). That is, when specifying only one node, you receive one ticket; when specifying multiple nodes, you receive a corresponding number of tickets. The sysctld daemon relies on the *rcmd* Kerberos principals.
2. The sysctl command is sent over to the sysctld on the server side.
3. The sysctld receives the request and performs a few steps before executing the request:
    - The user is authenticated to determine whether or not he or she is a Kerberos user. This check, in fact a sysctld-connection callback, determines whether he or she can use sysctl. The procedure invoked is *svcconnect* (by default, an AUTH-type command), so only Kerberos users can connect to the sysctld server. (If authenticated, the user can then connect and is able to run AUTH commands; if not, the connection is refused).
    - The rcmd ticket is decoded to verify if remote execution is allowed.
    - The session′s environment variables are defined, as designated in the server′s sysctl configuration file.

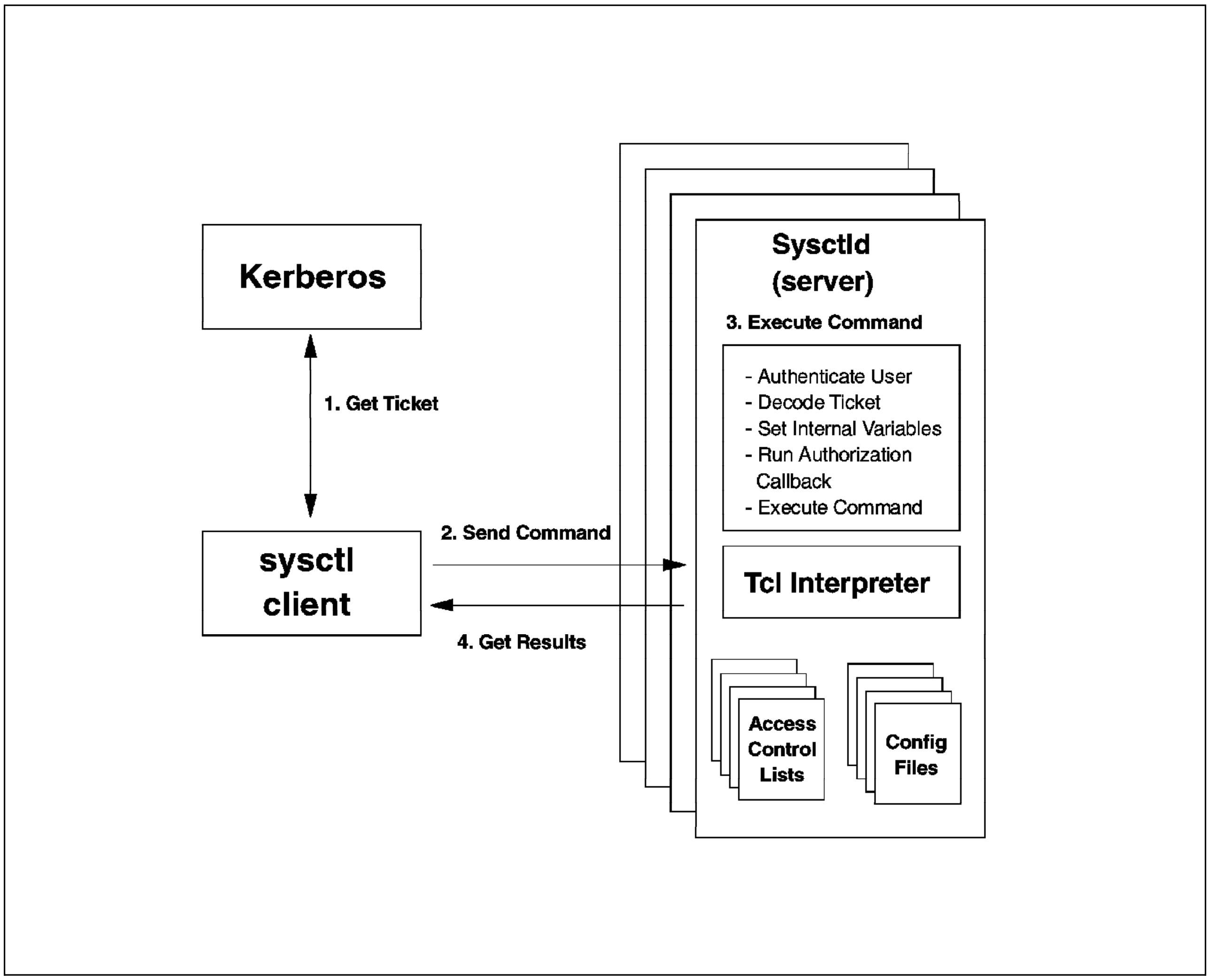


Figure 19. Sysctl in Action

- The callback procedure is executed for the client′s requested procedure or command. The authorization of the user is matched against the authorization of the procedure or command. Authorization checks are verified for AUTH, ACL and SYSTEM. Accordingly, execution is granted or denied.
- The Tcl or TclX code is interpreted. In this step, the actual work is performed. The code is interpreted and executed as root, unless you have specified that the code should be executed as another user with the **id** tcl-command.

4. The command responds.

8.3.5.1 Sysctl Environment

The *sysctl* environment is composed of the following:

- **The Sysctld daemon**: This daemon runs on every node, providing the sysctl service. Sysctl clients send their requests to this (server) daemon. The sysctld daemon runs commands as root. In Tcl scripts, you can ensure that commands are executed as the calling user (an example is provided in 8.3.5.3, “The /etc/sysctl.acl File” on page 93).

In the default configuration, sysctl can only be used by the root user, who is made a Kerberos user during installation of the Control Workstation. To make sysctl available to other users, you must change the sysctl environment so that non-Kerberized users can connect to the sysctl daemon. Do so by changing the default callback of the *svcconnect* sysctl procedure. Adding users to the Kerberos environment allows them to connect to sysctl and execute the AUTH-callbacked commands (AUTH meaning Authenticated). Examples are provided in 8.3.5.3, "The /etc/sysctl.acl File" on page 93.

The characteristics of the sysctl environment can be configured in the default /etc/sysctl.acl file.

The sysctld syntax options are:

sysctld [-s], security audit mode, logs each user request
[-d], debug option
[-a ACL-file], where the default is /etc/sysctl.acl
[-f ACL-config-file], default /etc/sysctl.conf
[-k Kerberos-key-file], default /etc/krb-srvtab
[-l Log-file], default /var/adm/SPlogs/sysctl/sysctld.log
[-P Port-number], default Portnumber 6680

Refer to Appendix E, "Sysctl Debug Output" on page 261 for an example of a debugged sysctld daemon.

- **The Sysctl command**: This command can be used either for one node interactively or for multiple hosts in batch mode. When using sysctl in an interactive session, specify one node only; for example:

```
[root@sp21cw0] / >sysctl -h sp21n07
Sysctl (Version 1.1) on sp21n07.itsc.pok.ibm.com
sysctl>
```

As a result of this command, it will appear as if you are logged into the node sp21n07. You will get a sysctl prompt from which you can enter sysctl-type commands.

The sysctl syntax options are:

sysctl [-c Collection of Nodes], specify a file with nodenames.
[-f Number-of-Nodes-Concurrent], default 8, maximum 128.
[-h Host-to-Connect], default localhost, multiple -h flags allowed.
[-L], makes HOSTNAME precede output, like in dsh.
[-m], use message authentication.
[-n], no authentication information sent.
[-P Port-number], default Portnumber 6680.
[-q], quick mode, do not wait for "slow" nodes.
[-r File or - (stdin)], replay option, sources sent to sysctld.
[-s], Server will send results back over TCP socket.
[-t Seconds], timeout time for connection, default 10 seconds.
[-T Seconds], time to wait for result, default 30 minutes.
[-v], prints sysctl version and exits.
[-x], the sysctl ping. Sends NULL RPC to server.
command

Note: The -f flag specifies the maximum number of nodes served in one cycle. If, for example, your configuration has 24 nodes and your -f flag is set to 8, sysctl will contact eight nodes at a time for three cycles.

Refer to Appendix F, “Sysctl and TclX Command Reference” on page 273 for a complete list of all sysctl commands, their descriptions and syntax.

- **Additional Sysctl commands on SP**: The sysctl implementation on the SP provides several extra commands to perform remote tasks. A brief description of each of these commands is provided in this section.

  These commands are added during sysctl startup through the sysctl.conf file. The files containing the commands′ source code can be found in the following directories:

  – /usr/lpp/ssp/sysctl/bin
  – /usr/lpp/ssp/samples/sysctl
  – /usr/lpp/ssp/samples/sysctl/rcmds

  These commands are written in the Tcl or TclX command language.

  Note: You can find more information on Tcl and TclX in Appendix D, “README Files for Tcl and TclX” on page 251.

  The additional commands are:

  – **fscheck** number: This command returns all filesystems that have a percentage used of more than number.

    Default authorization: AUTH

    Here is an example of the output from this command:

```
sysctl> fscheck 80
/ ==> 86.78% Used, 1624 KB Free
/usr ==> 99.70% Used, 3508 KB Free
/home ==> 85.56% Used, 5324 KB Free
/home/ftp ==> 82.41% Used, 73508 KB Free
```

  – **df**: Works similar to the AIX **df** command, with a slightly different format.

    Default authorization: AUTH

  – **pdf**: Similar to df, but provides more information. You may suppress the header line with the **-g** option.

    Default authorization: AUTH

  – **pstat**: Works as the AIX **ps** command does.

    Default authorization: AUTH

    In this case, however, you can also use process names; for instance:

```
sysctl> pstat amd
CMD          PID   PPID USER       INVOCATION
amd        20050     1  root       /etc/amd/amd -t 16.120 -x all -l -p /u / ...
```

  – **pkill**: Works similar to the AIX **kill** command.

    Default authorization: ACL

    In this case, however, just as with **pstat**, you may specify the command name:

```
sysctl> kill -15 amd
```

  – **rcopy**: The sysctl version of **rcp**. We could not get this command to work as it was provided in the default configuration. The command must be rewritten (see the discussion on rewriting the rexec command in 8.3.6.3, “Adding Procedures to Sysctl” on page 95).

– **rexec**: The authorized version of the **system** command. As with rcopy, this command will not run with the default configuration. We rewrote it to make it work (see 8.3.6.3, "Adding Procedures to Sysctl" on page 95).

– **pfck** option *number*: The sysctl filesystem checker.

  Default authorization: AUTH

  Valid options for this command are:

```
-s   : file system size is        > number (KB)
-f   : file system free space is   < number (KB)
-u   : file system used space is   > number (KB)
-pf  : file system free space % is < number %
-pu  : file system used space % is > number %
-is  : file system inodes is       > number
-if  : file system free inode is   < number
-iu  : file system used inodes is  > number
-pif : file system free inode % is < number %
-piu : file system used inode % is > number %
```

– **pfps**: The sysctl process handler.

  Default authorization: AUTH

  Valid flags for pfps are:

```
-n  name        fully-qualified command name of process
-tn name        command name of process
-o  owner       user ID that owns the process
-pty name       process controlling terminal
-rtime hh:mm    total execution time of process
-stime dd:hh:mm start time of process
-r state        run state of process
-cpu percent    true for processes %cpu > specified %cpu
-mem percent    true for processes %mem > specified %mem
-or             logical or arguments
-kill signal    kill process with specified signal if
                flags evaluate to true
-nice n         nice process with specified value, n, if
                flags evaluate to true
-print          print all processes that evaluate to true
                in the "ps guw" format
```

  Note: You must have an ACL for the nice option.

If you have additional requirements, you can code your own Tcl programs and add them to sysctl. Specify the additions in the file /etc/sysctl.conf.

Note: The README file for sysctl, in /usr/lpp/ssp/README, contains more helpful information on how to use the SP implementation of sysctl.

8.3.5.2 The /etc/sysctl.conf File

The /etc/sysctl.conf file contains the definitions and configuration characteristics of a node's sysctl environment. This file is present on all nodes running the sysctld. Basically, you can specify the following definitions:

Global Variables: Global variables are variables that can be referenced throughout the sysctl environment. In the default configuration file provided with the installation of sysctl, the variable buildTop is defined. The general syntax for defining variables is:

```
create var varName varValue [varAuth]
```

An example in /etc/sysctl.conf would appear as:

```
create var buildTop /usr/lpp/ssp AUTH
```

Creation of Classes: A class in sysctl is a collection of commands and/or variables that you can choose to group together. Organizing your system commands and variables into particular collections or classes in this way allows you to:

- Assign authorization callback for that group.
- As a result, assign permissions to sets of users.
- Create an administrative hierarchy for commands and tasks in a very efficient and productive manner.

Note: You cannot create subclasses. Defining a class within a class will result in a new class.

The general syntax to define classes is:

```
create class className classFile [classAuth]
```

An example in /etc/sysctl.conf would appear as:

```
create class help $buildTop/help/help.cmds
```

Creation of Procedures: It is probably best to define procedures in separate files. Use the *include* command to define the new procedure to sysctl.

Within the body of a procedure, you can specify sysctl commands of any callback type. Once a user has been authorized to use a procedure, the server disables authorization checking during execution of the body of that procedure. This means a user can run higher-authorized commands from within a procedure than he could normally run directly.

General syntax to define procedures:

```
create proc procName procArgs procAuth procBody
```

Refer to one of the procedure files for examples.

Including Procedure Files: Procedures can be included in the sysctl environment by specifying the *include* command (followed by the filename) within the configuration file; for example:

```
include $buildTop/sysctl/bin/pdfpfck.cmds
```

Set: Setting values for ACL, LOG and KEY variables and for the **env**() array can be done with the *set* command. The ACL, LOG and KEY variable can be used to assign different filenames for a(n):

- ACL (other then the default /etc/sysctl.acl); does not override the -a flag of sysctld daemon:

  ```
  set ACL /etc/other.acl
  ```

- LOG; does not override the -l flag of the sysctld daemon. Either specify the **sysctld -l** logfile or specify:

  ```
  set LOG /tmp/mysysctl.log
  ```

- KEY (other then the default /etc/krb-srvtab file). Again, does not override the -k flag of the sysctld daemon.

  ```
  set KEY /etc/my-kerby-key
  ```

The env() array contains the environment variables of the sysctld environment. You may compare this array with the AIX /etc/environment file. This example is extracted from the sysctl.conf file:

```
set env(PATH) /bin:/usr/bin:/usr/ucb:/etc:/usr/etc
```

The following variables have the same meaning as in an AIX shell session and may be set accordingly:

```
   EDITOR          NLSPATH
   LANG            ODMDIR
   LOCPATH         TZ
```

The following variables are set by the sysctl daemon as soon as a user connects to a sysctl server. These variables are set from the authentication information sent by the sysctl client at the moment a connection is established. In this way, the procedures invoked in the sysctl session know about the identity of the user making the connection. The sysctl daemon uses these variables to make decisions based on the (authenticated) identity of the client.

```
   SCHOST          SCMODE
   SCINSTANCE      SCPRINCIPAL
   SCLHOST         SCPRIVATE
   SCLPRINCIPAL    SCREALM
   SCLREALM        SCUSER
```

It is not recommended to change these variables or to set them explicitly.

Changing Callbacks: There are a few commands in the default setup of sysctl that may require a change in the default callback or require a separate ACL file.

Changing the default callback for svcconnect allows non-Kerberos users to make use of sysctl. In the default configuration, there is only one command available for non-authenticated usage: **return**. So, additional programming is required before this becomes of much use. The command to enable non-Kerberos users is:

```
setauth -cmd svcconnect NONE
```

Associating ACL files with sysctl commands might be very useful for commands like *exec* and *system*. These commands are basically exits to AIX, where, if authorized, the calling user has root access. The command to do this ACL association is:

```
setauth -cmd exec {ACL /etc/sysctl.exec.acl}
setauth -cmd system {ACL /etc/sysctl.system.acl}
```

These files contain the user principals, giving certain users authenticated rights to use the specific commands.

Following is an example of an entry from the `/etc/sysctl.exec.acl` file:

```
#acl#
_PRINCIPAL peter.@ITSC.POK.IBM.COM
```

In this particular example, the only principal having access permission to the *exec* command is PETER; no one else has it, not even the root.admin principal. Within this ACL file, you can also define the other parameter, _ACL_FILE, for defining concatenated ACL files.

The advantage of using special ACL files for individual commands is:

- You only assign authority for execution to one command instead of authority for all ACL commands by defining users in the global ACL file.
- Granting root access to users without spreading the root user′s password.

8.3.5.3 The /etc/sysctl.acl File

The sysclt.acl file is comparable to the Kerberos *.klogin* file. It allows principals to access commands with an ACL callback from remote systems.

There are two different types of ACL files:

Global ACL file: The global ACL file is the ACL file defined by the sysctld daemon. Any principal configured in the global ACL file can run any ACL sysctl command as *root*.

The sysctl global ACL file determines sysctl access permissions for *All* ACL-authorized commands in the global ACL file. The global ACL file is set by either:

- Not specifying the -a flag with the sysctld daemon, thus taking the default `/etc/sysctl.acl`.
- The *-a* flag of the sysctld daemon, specifying another global ACL file.
- A *set ACL* `<my-acl-file>` command in the sysctl configuration file.

Command ACL file: A command ACL file controls individual access to an ACL command for user principals. This is, so to speak, a way of sub-defining access control for ACL callback commands. Sub-defined access control must be configured in the sysctl configuration file using the syntax:

`setauth -cmd` **`ACL-command`** `{ACL command-ACL-file}`

As soon as you decide to implement granularity in your ACL definitions, you have to be aware that explicit ACL definitions, as shown in the above example, overrule the global definitions for that specific sysctl command. This means that the users mentioned in the global ACL file have access to all ACL files; but as soon as you have defined an individual ACL file, only the principals mentioned in that ACL file have access to the command associated with that ACL file (as defined with the setauth command). By default the root user is defined in the global ACL file, which gives the root.admin principal access to all commands, but if you decide to add an ACL file for a particular ACL command, the root user principal must also appear in this additional ACL file to keep access for that particular command.

As already shown in examples in other sections, the format of any ACL file must look like this:

```
#acl#
#
#Commented Lines
#_PRINCIPAL clive.@BEDFONT.LAKES.UK
#For denied users.
#
_PRINCIPAL <username.>[instance]@REALM
#
#Sub-ACL definitions, node dependencies
#
_ACL_FILE <file-name>
#
```

Note: Do not forget the period (full stop) after the username when no instance exists for the principal.

8.3.6 Using and Configuring Sysctl

The sysctl configuration, as it is when first installed, is basically only suited for the root user. The root user′s principal is configured in the global ACL file, and the root user is the only authenticated user. In other words, with the default configuration, only authenticated users (thus, only root) can use sysctl.

This section discusses two ways of changing the default configuration for different needs.

8.3.6.1 Adding Normal Users to Sysctl

There are two ways of enabling access to sysctl for non-root users:

- **Adding Users to Kerberos**: Defining users in Kerberos authenticates them (a requirement for access under the default sysctl configuration). Once authenticated in sysctl, the user has access to all commands with an AUTH callback. This method is the recommended approach for allowing users to connect to sysctl. In this way, you can limit and control the access of remote users to nodes on the SP. (Refer to Chapter 4, “Kerberos” on page 39 for details on Kerberos and how to add users to the Kerberos database).
- **Changing the default callback of svcconnect**: As discussed in 8.3.5.2, “The /etc/sysctl.conf File” on page 90, the **svcconnect** command connects a sysctl client to the sysctld server. In the default configuration, svcconnect has an AUTH callback. This means you must be known to Kerberos in order to use the command. You can change the default callback of svcconnect in the sysctl configuration file:

  ```
  setauth -cmd svcconnect NONE
  ```

  Now, non-authenticated users have access to sysctl. However, only commands with an authorization level of NONE can be run by these non-authenticated users.

  Note: When changing the default callback of svcconnect, every user in the network can connect to the sysctl server. As a consequence, any user can now “chew up” resources on those servers having svcconnect set to NONE.

8.3.6.2 Granting Access to ACL Commands

We illustrate granting access using an example. Let us say that we would like a particular user to have the ability to kill commands on one or a number of remote hosts. This user must, for instance, be able to stop and restart the **amd** daemon and to refresh Amd maps. We do not want to grant access to any other ACL command. To accomplish this, each sysctld daemon on the relevant nodes must be reconfigured with the steps described below:

- Make this user (Clive) an authenticated user in Kerberos.
- Add the following line to the sysctl configuration file:

  ```
  setauth -cmd pkill {ACL /etc/sysctl.pkill.acl}
  ```

- Create the file /etc/sysctl.pkill.acl and add the following lines:

  ```
  #acl#
  _PRINCIPAL clive.@ITSC.POK.IBM.COM
  ```

- Restart the **sysctld** daemon with:

  ```
  kill -15 <sysctl-pid>
  ```

  Most likely, your sysctl environment looks the same on all SP nodes. In that case, you would distribute the new sysctl definitions to all nodes (with pcp).

  Restarting the sysctl daemon can be done in a more convenient way using the **svcrestart** sysctl command:

  ```
  [root@sp21cw0] / > hostlist -av > /tmp/nodes
  [root@sp21cw0] / > sysctl -c /tmp/nodes svcrestart
  ```

  or,

  ```
  [root@sp21cw0] / > hostlist -av | sysctl -c - svcrestart
  ```

The following illustrates what user Clive can (and cannot) do once granted access:

```
Sysctl (Version 1.1) on sp21n01.itsc.pok.ibm.com
sysctl> exec ls
sysctl:  2501-122 exec: Insufficient Authorization.
sysctl> pstat aixterm
CMD          PID   PPID USER       INVOCATION
aixterm    38372 32738  root       aixterm
sysctl> pkill -9 38372
sysctl>
```

8.3.6.3 Adding Procedures to Sysctl

Sysctl is far from complete. It provides the basic commands for trivial operations, yet system-management tasks consist of a complex combination of those commands. Adding procedures requires the following steps:

- Writing the procedure in Tcl or TclX
- Defining Callbacks for the commands
- Adding the commands file to the sysctl configuration file

As mentioned in 8.3.5.2, "The /etc/sysctl.conf File" on page 90, there are two commands that are defined in the default configuration but are not functional: rexec and rcopy (part of the sysctl remote commands).

In this section, we provide an example that shows how to:

- Add a new callback to sysctl

- Add a new procedure to sysctl (rexec, rlog and raix) that allows users to exit to AIX and execute a command

Adding a New Callback to sysctl: (Use the following example as a basis for creating your own callback procedure). In this example, we create the procedure **rcmd_auth**, which will be invoked when a command is accessed with the **rcmd_auth** authorization callback. This callback′s purpose is to check whether a user is authenticated or not. This procedure in itself is a SYSTEM-callback command, since we do not want it to be executed interactively.

```
create proc rcmd_auth {cmdName} SYSTEM {
        global SCPRINCIPAL
        if [catch AUTH] {
           svclog ″Denying \″$cmdName\″ access for user $SCPRINCIPAL″
           error ″Authorization denied″
        } else {
           svclog ″Authorizing \″$cmdName\″ access for user $SCPRINCIPAL″
           return ″Authorization ok″
        }
}
```

Adding a New Procedure: This example shows you how to grant "safe" access for all authenticated users to the **system** command in sysctl. If we were to add normal users to a command ACL file, those users would have root access to the server nodes. What we really want is to enable users to exit to AIX using the system command, but with their own identities.***:*** The first step is to create the procedure. Our sample procedure does the following:

1. The **rcmd_auth** callback is assigned to the command.
2. The **svclog** command is used to count the number of times this command is accessed (to monitor the usage of this command).
3. The user′s identity is assigned to the rexec command body, ensuring the command is executed as the calling user and not as root.
4. The system command joins all arguments passed to rexec and executes the command.
5. For convenience, the command is renamed in sysctl. The real format of the command name would be:

   `[class-name]:[command-name]`

   Renaming the command `rcmds:rexec` to simply `rexec` provides a short cut.

Our example is shown in Figure 20 on page 97.

Note: Put both the callback and the procedure in one file; for example, `/usr/local/sysctl/remcmds.cmds`.

To ensure that the sysctl environment recognizes the new **rexec** command and callback, we must include this procedure in the sysctl configuration file by assigning a new class for this command, class rcmds. Add the following line to the `/etc/sysctl.config` file:

```
create class rcmds /usr/local/sysctl/remcmds.cmds
create proc rexec {args} {CLASS:rcmd_auth rexec} {
        global SCUSER

        # Log, for good measure
        svclog "rexec: user=$SCUSER, command=\"$args\""

        # Assume the remote user's identity
        id user $SCUSER

        # Issue the command and return the exit status
        system [join $args]

}
#To make sure we can invoke the new command with rexec
#and not with rcmds:rexec:
#
rename rcmds:rexec rexec
```

Figure 20. Sample sysctl Procedure

Distribute these definitions to all SP nodes with pcp, then restart the **sysctl** daemon:

```
[root@sp21cw0] / > hostlist -av > /tmp/nodes
[root@sp21cw0] / > sysctl -c /tmp/nodes svcrestart
```

or,

```
[root@sp21cw0] / > hostlist -av | sysctl -c - svcrestart
```

As you can see, the rexec command joins all arguments of the rexec command together and presents them to the **system** command. The system command exits to AIX and executes the arguments. Normally, only an ACL-authorized user is able to execute the **system** command, but this is executed from the body of a script and, as you will recall, authorization checking in the body of a script is switched off. So, users with a lower authority can now execute the system command through this rexec command, but cannot execute AIX administrative commands nor can they execute the system command separately. To illustrate this last point, we use two examples:

- Assume Peter, a non-root user known to Kerberos, uses the following rexec command:

  ```
  sysctl> rexec ls /u/clive
  ```

  Peter will only be successful if he has read permission in the /u/clive directory on the server host. (And, we do not have a warm feeling that Clive will let Peter have a peek in his directory!)

- The following will fail because of AIX permissions (user peter is not root). Note that we use the Kerberized rsh command.:

  ```
  sysctl> whoami
  peter.@ITSC.POK.IBM.COM
  sysctl> sys:host_name
  sp21cw0.itsc.pok.ibm.com
  sysctl> rexec /usr/lpp/ssp/rcmd/bin/rsh sp21n05 rm /etc/passwd
  ```

  The error message that results is:

  ```
  rm: 0653-609 Cannot remove /etc/passwd
  The file access permissions do not allow the specified action.
  0.
  sysctl>
  ```

8.3.6.4 Exits to AIX from Sysctl

In our experience, using **rsh** to remotely check a node or browse through system log files on a remote host is not very efficient. One way of doing this more conveniently is using dsh interactively, by issuing dsh without a command. This allows you to run the "rsh"ed commands on several nodes at once.

We found a way to use sysctl in the same manner, but only to one node. Note that any commands you run are run as root on the remote system. Also, note that only users authorized to run the "system" command can use this technique. Normally, this will be only those users with principals in the global sysctl ACL file (/etc/sysctl.acl).

You must perform the following to get a full (almost) AIX interactive session in sysctl:

1. Enter the sysctl shell by inputting **sysctl**. For example, to connect to the sysctld on node sp21n12:

   ```
   [root@sp21cw0] / >sysctl -h sp21n12
   Sysctl (Version 1.1) on sp21n12.itsc.pok.ibm.com
   sysctl>
   ```

2. Enter the system command to exit to AIX and invoke an interactive ksh shell:

   ```
   sysctl>system ″/bin/ksh -i″
   #
   ```

   Note: The straight double quotes present the command and command option as one string to the system command.

3. Once in the ksh shell, you have no environment variables at your disposal. This exit does not take care of any of the usual login sequence, wherein the environment variables are set and the user profile is executed. So, within this "bare" interactive shell, you must set these parameters explicitly. You could create a dummy profile program, which is shown is Figure 21.

```
#!/bin/ksh
PATH=/usr/lpp/ssp/rcmd/bin:/usr/bin:/etc:/usr/sbin: \
     /usr/ucb:/usr/bin/X11:/sbin: \
     /usr/lpp/ssp/bin:/usr/lpp/ssp/kerberos/bin: \
     /var/sysman:/usr/lpp/ssp/kerberos/etc:.
PS1=″hostname -s > ″
LOGNAME=whoami
HOME=cat /etc/passwd | grep ^whoami \
            | awk '{FS=″:″};{print $6}
export PATH PS1 LOGNAME HOME
```

Figure 21. Dummy Sysctl Session Profile

Name the file `sysctl.prof` and store this file on the remote host in the user′s home directory.

Execute the following from within this "bare" ksh shell and you become "ksh interactive" on a remote host:

```
# . /sysctl.prof
sp21n12 >
```

At this point, you are not logged in as the root user. You still have the interpreter interface on the local host through sysctl, but it looks and feels like being logged into the remote host. You will notice slower response times since the commands must go a longer way (through sysctl). Remember, too, that this interactive session is a command-line interface, not a full-screen one. So editing a file in full-screen mode is not possible in this way. (As mentioned, the work-around is to copy the file you want to edit to your local workstation, edit and change the file there and send it back to its original location). But, now you can quickly scan the node for any possible problems, browse log files and perform commands that are command-line driven.

However, if you like to have to possiblility to connect to a remote host through sysctl, and still be able to edit files, you can use the following sysctl command:

```
sysctl -h sp21n12 'system "/usr/lpp/X11/bin/aixterm -display sp21cw0:0"'
```

This command executes aixterm on the remote host, displaying it on the Control Workstation (in our case) allowing you to edit files on the remote host, provided the remote host has connection permissions to the local X-server. Building this command into a sysctl procedure would be ideal: it requires, however, dynamic setting of the *display* variable in the sysctl procedure. You can do that with the SCHOST variable, representing the connecting host. This is shown in Figure 22.

```
create proc raix {args} {CLASS:rcmd_auth raix} {
        global SCUSER SCHOST

        # Log, for good measure
        svclog "raix: user=$SCUSER, command=\"aixterm -display&bsl" host=$SCHOST"

        # Assume the remote user's identity
        id user $SCUSER

        # Issue the command and return the exit status
        system "/usr/lpp/X11/bin/aixterm -display $SCHOST:0"

}
#To ensure we can invoke the new command with raix
#and not with rcmds:raix:
#
rename rcmds:raix raix
```

Figure 22. Sample System Command Procedure, Connecting with Aixterm

You can also provide the system command functionality to non-root users by defining a procedure around the system command, as shown in Figure 23 on page 100.

Now, non-root users can invoke the **rlog** command in an interactive sysctl session and experience the look and feel of being logged into a remote host. The "id user $SCUSER" command ensures that the connection is made as the user requesting the connection and not as root. For example:

```
[peter@sp21cw0] /u/peter > sysctl -h sp21n12
Sysctl (Version 1.1) on sp21n12.itsc.pok.ibm.com
sysctl> rlog
$ id
$ uid=7(peter) gid=0(system)
$ exit
0
sysctl> exit
[peter@sp21cw0] /u/peter >
```

```
create proc rlog {args} {CLASS:rcmd_auth rlog} {
        global SCUSER

        # Log, for good measure
        svclog "rlog: user=$SCUSER, command=\"ksh -i""

        # Assume the remote user's identity
        id user $SCUSER

        # Issue the command and return the exit status
        system "ksh -i"

}
#To ensure we can invoke the new command with rlog
#and not with rcmds:rlog:
#
rename rcmds:rlog rlog
```

Figure 23. Sample System Command Procedure

As you may have noticed, the three commands provided in this section as examples for making your own procedures have a certain hierarchy with regard to what you can do on a remote host.

- rexec, execute an AIX command on a remote host and exit
- rlog, create an AIX exit interactively, line mode
- raix, same as rlog but now full-screen through aixterm

Sysctl provides you with functionality that allows you to selectively grant access to one or a set of these commands with the **setauth** command. You could make separate ACL files for the rexec, rlog and raix commands in order to control access to these commands and, more importantly, the access to the remote hosts. Your requirements determine how you implement this.

The configuration used and tested when writing this chapter is shown in Appendix E, "Sysctl Debug Output" on page 261. Included are the /etc/sysctl.conf file, the source file with the rcmd_auth callback, rexec and rlog commands and the log file of the sysctld daemon, with audit and debug mode configured.

Note: The raix command is not a part of that list; it was created after completion of Appendix E, "Sysctl Debug Output" on page 261.

8.3.7 Practical Experiences with Sysctl

From our perspective, sysctl is one of the most powerful distributed-management tools; and it is secure, as well. Sysctl provides granularity to your access hierarchy, which is very difficult to accomplish with other management tools.

Also, when compared to dsh, sysctl performs very well, particularly with a large number of nodes. The dsh command must resolve all environment variables and perform a **cd** to the user's home directory. Sysctl does not do so and is, thus, faster.

Another advantage in comparison with dsh is that sysctl has a more "graceful" way of abnormally ending. With the **-t** flag of the sysctld daemon, you can specify how long the server must wait until a time out must be reported. After the amount of time specified with the -t flag, the sysctld daemon (client side) ignores the timed-out host and continues with the others. Dsh, however, uses rsh, which can take considerable time to time out.

Who has not had an uncomfortable feeling when distributing the root password among a large number of system administrators? With the use of sysctl, almost every system-administrative task can be performed without having to know the root password. So, the root password need only be made known to a limited number of people.

The most difficult part of sysctl is getting started. The concept is new. It took us some time before we saw the light, primarily because of the limited availability of comprehensive documentation. We sincerely hope this section will clear some of the mist.

Finally, with sysctl, as with other tools, you must decide what you want and how you want to do it. Simply implementing sysctl and seeing what you can do with it later will not yield the best from sysctl.

8.4 Distributed System Management Interface Tool (DSMIT)

With regard to distributed or clustered environments, we have covered most of the parallel tools available through the PSSP package. Although it is not a supported product on the SP, we would also like to mention the Distributed System Management Interface Tool (DSMIT) product in this section.

8.4.1 DSMIT Concepts

DSMIT provides an environment where SMIT commands are executed on clients throughout a network. DSMIT works the same as SMIT does for 99% of the operations. DSMIT has a few dialogues that are different from SMIT′s. DSMIT supports different operating systems as clients. All management functions are controlled from a single user interface. In particular, for an SP, DSMIT frees the system administrator from issuing difficult structured **dsh** commands and from building scripts for every complex administrative task (usually built with “cut” and “paste” operations on X-window terminals).

A DSMIT environment contains two members:

- **DSMIT Server**: The DSMIT server builds and distributes the administrative commands. The server also manages the client and group definitions that it serves.
- **DSMIT Client**: The DSMIT client is the receiving end of the DSMIT environment. It executes the commands built by the server. The DSMIT client program can be installed on the following operating systems:
  - AIX 3.2 or higher
  - Sun OS 4.1.3
  - HP-UX 9.0 (700 Series)

A DSMIT server sends SMIT commands to clients on a network. Within DSMIT, you can define groups of clients that you want to serve concurrently. A grouped number of nodes is called a *domain* in DSMIT.

Once a domain in defined, you can specify that domain to DSMIT with the **-W** flag. The nodes defined in a domain served by DSMIT are called a *Working Collective*. (Where have we heard this term before? **dsh**).

DSMIT can operate in two different ways to execute commands:

- Sequential Mode: This mode of operation executes the commands on one client at a time. DSMIT waits for output from the one DSMIT client before sending a command to the next client.
- Concurrent Mode: This mode sends the commands to all clients at the same time. Output will only be shown after completion of all clients.

8.4.2 Setting up DSMIT

As discussed earlier in this section, the DSMIT environment consists of a DSMIT server and DSMIT clients. Following the concepts of DSMIT and the SP, the Control Workstation is most likely to be the DSMIT server and all nodes DSMIT clients.

8.4.2.1 Installing the DSMIT Server

To make the Control Workstation the DSMIT server requires you to install the server portion of DSMIT from tape. The products that must be installed are:

```
1.1.0.0  dsmit_server
1.1.0.0  dsmitmEn_US
```

If you want to include the Control Workstation itself as a client, you must also install the DSMIT client software (1.1.0.0 dsmit_aix) on the Control Workstation.

The installation process downloads the commands and executables from tape and creates the configuration directory /usr/share/DSMIT.

After completion of the installation process, the following steps must be performed in order to make the DSMIT server work. We concentrate on our configuration in the Poughkeepsie Laboratory (refer to Figure 2 on page 7):

1. Create the *domains* file: The domains file contains all domains of the DSMIT server. When used in DSMIT, it corresponds to the Working Collective. Our domains file looks like this:

   ```
   group_d:sp21n07,sp21n08,sp21n09,sp21n10
   group_s:sp21n03,sp21n04,sp21n05,sp21n11
   group_t:sp21n01,sp21n02,sp21n06,sp21n12
   nodes:sp21n01,sp21n02,sp21n03,sp21n04,sp21n05,sp21n06, \
         sp21n07,sp21n08,sp21n09,sp21n10,sp21n11,sp21n12
   all:sp21cw0,sp21n01,sp21n02,sp21n03,sp21n04,sp21n05, \
         sp21n06,sp21n07,sp21n08,sp21n09,sp21n10,sp21n11,sp21n12
   ```

2. Create the *hosts* file: The hosts file contains all clients served by the DSMIT server. The hosts file lists all hosts, each with its hostname and operating-system type. After installation of the server, this file contains only an operating-system stanza: (:AIX_3.2, in our case).

   After installation, our hosts file looked like this:

   ```
   sp21cw0:AIX_3.2:0
   sp21n01:AIX_3.2:0
   sp21n02:AIX_3.2:0
   sp21n03:AIX_3.2:0
   sp21n04:AIX_3.2:0
   sp21n05:AIX_3.2:0
   sp21n06:AIX_3.2:0
   sp21n07:AIX_3.2:0
   sp21n08:AIX_3.2:0
   sp21n09:AIX_3.2:0
   ```

```
sp21n10:AIX_3.2:0
sp21n11:AIX_3.2:0
sp21n12:AIX_3.2:0
```

3. Create the *dsmitos* file: This file could contain any of the possible operating systems (that is, types of clients) served by DSMIT, in the following format:

```
AIX_3.2
HP-UX_9.0
SunOS_4.1.3
```

8.4.2.2 Installing a DSMIT Client

None of the nodes has a physical connection to a tape drive. This means that we must create the DSMIT software image on one of the Control Workstation′s disks in the /usr/sys/inst.images/ssp directory. However, DSMIT cannot be read from tape with **bffcreate** (the command to copy software from tape to disk for future installations). We must fall back on another method. Perform the following steps to get the software images from tape to hard disk:

1. Copy the tape′s table of contents (third image) to disk:
   - Change the tape block_size to zero with: chdev -l rmtX -ablock_size=0.
   - Skip to the third image: tctl -f/dev/rmtX.1 fsf 2.
   - Read the third image: dd if=/dev/rmt0 of=/tmp/toc.
2. Search for all **dsmit**-type products in the table of contents (toc) and register the position of the image on tape. In our case, the toc looked like this:

```
01:810 3 R I }
dsmit_server 01.01.0000.0000 01 N H En_US Distributed SMIT - AIX Server
  ⋮
01:822 3 R I }
dsmitmzh_TW.msg 01.01.0000.0000 01 N U zh_TW Distributed Smit Messages
```

   As you can see, the first DSMIT image started at the 810th position and the last image at position 822.

3. Read the DSMIT images from tape. We created the following script to do this:

```
#!/bin/ksh

chdev -l rmtX -ablock_size=0
tctl -f/dev/rmtX.1 fsf 809

for i in 810 811 812 813 814 815 816 817 818 819 820 821 822
do
   dd if=/dev/rmt0.1 of=/usr/sys/inst.images/ssp/tapefile$i
done
tctl -f/dev/rmt0 rewind
```

4. Create a new **.toc** file in /usr/sys/inst.images/ssp by entering:

```
/usr/sbin/inutoc /usr/sys/inst.images/ssp
```

   .

At this stage, the DSMIT client image is copied to disk and is suitable for installation.

Each node must have the DSMIT client code to run in conjunction with the DSMIT server.

To authorize a DSMIT server to communicate with a DSMIT client, you must configure one of the following files:

- The **/.rhosts** file: DSMIT uses the same level of security as that provided by the TCP/IP rsh commands.
- The **/etc/security/dsmit.hosts** file: If the use of /.rhosts is not desired, the /etc/security/dsmit.hosts file can be used instead. This file contains the hostname of the DSMIT server, in our case, `sp21cw0`.

8.4.3 DSMIT Restrictions on the SP

Although DSMIT uses the same level of security as that provided by the TCP/IP rsh commands, it does not use the rsh command. It uses its own client/server scheme based on the *crexd* and the *srexd* TCP/IP programs, built on top of the rsh command. (They stand for Client-Remote-EXecute-Daemon, and for Server-Remote-EXecute-Daemon, respectively. Refer to the `/etc/inetd.conf` and `/etc/services` files for more details).

This implies the following:

- The `/etc/security/dsmit.hosts` only contains the hostname of the DSMIT server. Therefore, any host can masquerade as the DSMIT server and, thus, violate security.
- We cannot tailor DSMIT to Kerberos. We might have been able to if DSMIT used the standard `/usr/bin/rsh` command. Replacing the TCP/IP rsh command with the Kerberized version would solve the security issue; however, DSMIT does not use the TCP/IP rsh command.

8.4.4 Summary

If you really want to use DSMIT on the SP, you can. We could find no technical restriction for doing so; we encountered no problems with usage nor errors specific to the SP. (Although a different way of installing the DSMIT programs is required). However, DSMIT is not supported. The main reason, at this moment, is the security issue. DSMIT does not (yet) comply with all of the SP security guidelines.

We find the DSMIT programs a very useful addition for systems-management tasks on the SP and regret that it is not officially available.

You can find more information on DSMIT in the following IBM publications:

- *AIX Distributed SMIT/6000 Version 1.1 Guide and Reference*, SC23-2561
- *DSMIT V2.1 for AIX, Guide & Reference*, SC23-2667
- *Experiences Using DSMIT in Heterogenous Environment and Visual System Management Under AIX V3.2.5 and V4.1*, GG24-4380

Part 4. Resource Control

Chapter 9. Data Management on the SP

A general Data Management policy is impossible to define for an SP. Each SP has its own specific data characteristics. Backup and archive policies depend on these characteristics, as well as on available hardware. An SP, as a parallel machine, can also introduce parallel complexity in terms of Data Management. Planning your data strategy before starting to use an SP turns out to be very valuable. As also discussed in Chapter 13, "SP User Management" on page 187, one of the considerations for your data policy is where you will define the SP users and where they can access their data.

We are not able to discuss all possible data-management issues. This chapter covers what is important to know in terms of data characteristics, based on our own experiences.

In addition, we provide more detailed information about an important SP data-management feature, virtual shared disks (in 9.8, "IBM Virtual Shared Disks and Recoverable Virtual Shared Disks" on page 111).

9.1 Centralized or Distributed

The physical distribution of your data will strongly depend on the way the SP is used. When used as a consolidation machine, it is most likely that the data will be stored on one or a limited number of nodes. As in the example we have used throughout this book, you may see a data server for each group. In an SP, and on UNIX clusters in general, NFS is the most commonly-used vehicle for making the data available in the network. On an SP, you have the advantage of using the switch for this purpose. Each connection, from one node to another, benefits from a dedicated link over the switch. When tuned properly, you can reach up to 4MB/sec reads and 1MB/sec writes with NFS over the switch.

When used as a parallel machine, you may want to have direct access to the data on each computing or database node. As with DB2/6000 Parallel Edition, each node contains a part of the database.

9.2 Offline or Online Data

Data can be stored in two different ways, online or offline.

Data is online when a user is able to open a file from the dataset. In the case of the SP, online data will most likely be available through filesystems. These can be both local and remote filesystems.

Data is considered offline when it is not possible to open a file directly from the dataset. An additional step is required to make the data available for usage, for instance by means of ADSTAR Distributed Storage Manager (ADSM/6000). The user must extract the data from the offline dataset using a command like **tar** or an ADSM command.

9.3 Storage Hardware

At this time, four classes of storage hardware are available to the market:

- Hard disks
- Memory
- Optical disks
- Tape media

Each of these classes varies in its price and usage. The speed with which you need to access the data, the price you want to pay for the hardware and the availability of the data will determine the mix of storage hardware for your requirements.

Each customer has some kind of mix, as described above. In most cases, that mix is based on the availability factor and the frequency of usage. Data that is accessed most frequently you want to store on the fastest possible data storage. Data that is not used or is used with very low frequency you want to store on the cheapest or slowest media.

The type of media used will strongly determine the data-management policies. When using a data-storage hierarchy, products like ADSM/6000 can be of great help.

9.4 Storage Software

This section covers the software available in the marketplace for storage solutions. The tables provided below are based on the following scenarios:

- Backup
- Archive
- Space Management
- Offline hierarchy
- Availability

against the following types of files:

- Relational databases
- Large-sized riles (100MB-2GB)
- Medium-sized files (10MB-100MB)
- Small-sized files (0-10MB)

Table 11 (Page 1 of 2). SP Data-Management Solutions

	RDBMS	**Large**	**Medium**	**Small**
Backup	REELibrarian, ADSM, EPOCH, Legato	N/A	ADSM, EPOCH, Legato	ADSM, EPOCH, Legato
Archive	N/A	UniTree, HPSS	ADSM, HPSS, EPOCH	ADSM, EPOCH, Legato
Space MGT	N/A	N/A	FSF/6000, ADSM (1995), EPOCH	FSF/6000, ADSM (1995), EPOCH
Offline Hierarchy	N/A	UniTree, HPSS	ADSM, EPOCH, HPSS	ADSM, EPOCH, Legato

Table 11 (Page 2 of 2). SP Data-Management Solutions

	RDBMS	Large	Medium	Small
Availability	HACMP, RAID, Mirroring, OLTP Rollback	Checkpointing, RAID	HACMP, RAID	HACMP, Mirroring

9.5 System Backups

An SP contains multiple nodes which are either a **/usr** client or which contain a full implementation of AIX. In general, all nodes look the same, each containing a full AIX copy or each served by a **/usr** server. In the case of a partitioned SP (that is, one divided into several server groups), you may see differences among the groups.

At the least, each node has the root volume group: **rootvg**. This volume group is the container of the operating system and must be backed up regularly. Since the nodes all look the same (that is, look the same within the logical group), you need only make one system backup per group. An example of how to make a system backup of an SP node is discussed in 9.6, “AIX Backup Commands.”

9.6 AIX Backup Commands

Apart from the software solutions mentioned in 9.4, “Storage Software” on page 108, you can always use the backup and restore commands provided by AIX. The most familiar are:

- **tar**: This command is probably the most commonly-used backup/restore command. The distinction between backup and restore is made by the flags **-c** for **c**reate and **-x**, for e**x**tract.
- **backup** and **restore**: In addition to making backups and restoring data, these commands allow you to create “levelled” backups. A level “0” backup is a full backup. All other levels are considered incremental backups, which may vary in time frequency. The backup command supports up to nine levels. Refer to InfoExplorer for detailed information.
- **cpio**: Similar to the tar command, it uses the **-i** flag for retrieving files from archive and the **-o** flag for migrating files to an archive.
- **dd**: The dd command is particularly useful for packing and unpacking host files, converting EBCDIC to ASCII and vice versa and manipulating raw devices like the dump device (/dev/hd7). It has also proved to be a very handy tool when there are version differences between the backup/tar command that wrote the archive and the restore/tar command reading the archive. For example, say you receive a tape with a tar image and, for some reason, the tape is unreadable. Try the following example to circumvent:

  ```
  chdev -l rmtX -ablock_size=0

  dd if=/dev/rmtX | tar -xvf -
  ```

  Or, try it with blocksizes:

  ```
  chdev -l rmtX -ablock_size=0

  dd if=/dev/rmtX bs=10240 | tar -xvf -
  ```

- **mksysb**: The mksysb command is built on top of **tar**. The command creates a tape with four images:

  - Boot image
  - Display image
  - Table of contents, dummy in this case
  - The system backup

  System backups are basically only useful when you have a tape drive connected to the system. In the case of an SP, this would require having a tape drive on every node, and that is not very practical. There is, however, a way to share one tape drive on your SP, specifically the one that is attached to the Control Workstation.

  To make a network system backup, perform the following steps:

  1. Log into the system you want to back up.
  2. Determine the hostname of the node/workstation that is connected to a tape drive. In our situation, it is the Control Workstation: (sp21cw0). Set the environment variable to **REMHOST=sp21cw0**.
  3. Issue the following commands:

     a. `export REMHOST`
     b. `cd /tmp`
     c. `mknod /tmp/pipe p`
     d. `mkszfile`
     e. `mksysb /tmp/pipe`

     At this point, you will receive the warning:

     ```
     mksysb: WARNING: /tmp/pipe does not appear to be a tape device and
             will NOT have a bootable image.
     ```

     This is because the first three images of the system backup cannot be made over the network. To do so requires "no rewind" signals, which only work on a local system. After this warning, the backup begins to write output to your terminal. When the write buffer reaches 32K bytes, terminal output will stop. Continue with the following step.

  4. Execute the command:

     ```
     dd if=/tmp/pipe | rsh $REMHOST "dd of=/dev/rmt0 obs=512"
     ```

     It will run until the backup has completed.

     Note: Make sure Kerberos works and generate a ticket for root, or rely on the .rhosts file and use /usr/bin/rsh.

  5. And, finally, enter: `rm /tmp/pipe`

9.7 Related Publications

There are several publications available that may help you to define a data-management policy and implementation. The following list covers most of today's data-management issues on RISC System/6000 and SP:

- *AIX Storage Management*, GG24-4484
- *Getting Started with ADSM/6000*, GG24-4421
- *Using ADSM to Back Up Databases*, GG24-4335
- *ADSTAR Distributed Storage Manager/6000 on 9076 SP2*, GG24-4499

9.8 IBM Virtual Shared Disks and Recoverable Virtual Shared Disks

This section describes the IBM Virtual Shared Disks, the IBM Virtual Shared Disk Data-Striping Devices and also the Recoverable Virtual Shared Disk Software.

You should refer to the *IBM 9076 Scalable POWERparallel Systems SP2 Administration Guide* for details on the virtual shared disk (VSD) and the virtual shared disk data-striping device [also known as hashed shared disk (HSD)]. The HSD is actually a kernel extension that works with the VSD to provide the striping capability. The recoverable virtual shared disk (RVSD) is described in the *IBM Recoverable Virtual Shared Disk User Guide and Reference*.

The following sections describe VSDs, HSDs and RVSDs in turn. An overview of each product is followed by an example of how to set up and manage the product. Finally, the author′s experiences and hints and tips are discussed.

9.8.1 IBM Virtual Shared Disk Overview

The VSD software is used to make AIX logical volumes appear to be located on more than one node. A logical volume that is physically located on a disk attached to a particular node of an SP can also be accessed from other nodes that have been defined to be part of the VSD "cluster." See Figure 24 on page 112 for a look at how VSD can work on SP nodes.

In this figure, three nodes are shown. Nodes 1 and 2 are server nodes and node 3 is a client node. The logical volume LV1 is located on node 1. The logical volume LV2 is located on node 2. Both logical volumes appear as locally available on all three nodes.

Note: Only logical volumes can be accessed using the VSD software. An AIX filesystem cannot be mounted onto this logical volume; therefore, normal AIX files cannot use this mechanism. Logical volumes, accessed through the VSD, are currently used by the Oracle Parallel Server database only.

A logical volume that is to be accessed from other nodes using VSD is physically located on a disk attached to one of the SP nodes. This node is called the server node. The other nodes that access this same logical volume are called client nodes. The client nodes access the logical volume in exactly the same way that they would access local devices. It is transparent to the user or application that the logical volume is actually located on a remote node.

Nodes can be both clients and servers for different VSDs.

The I/O routing is performed by the IBM VSD device-driver layer that works on top of the usual AIX Logical Volume Manager (LVM) and the Internet Protocol (IP).

The network connection that is used for accessing the data between the server and the client nodes can be either a Local Area Network (LAN), Ethernet for example, or it could be the SP High Performance Switch (HPS). If the SP HPS is used for VSD communication between nodes, it must be running the IP network protocol. Currently, the VSD software does not support the *userspace* protocol on the HPS.

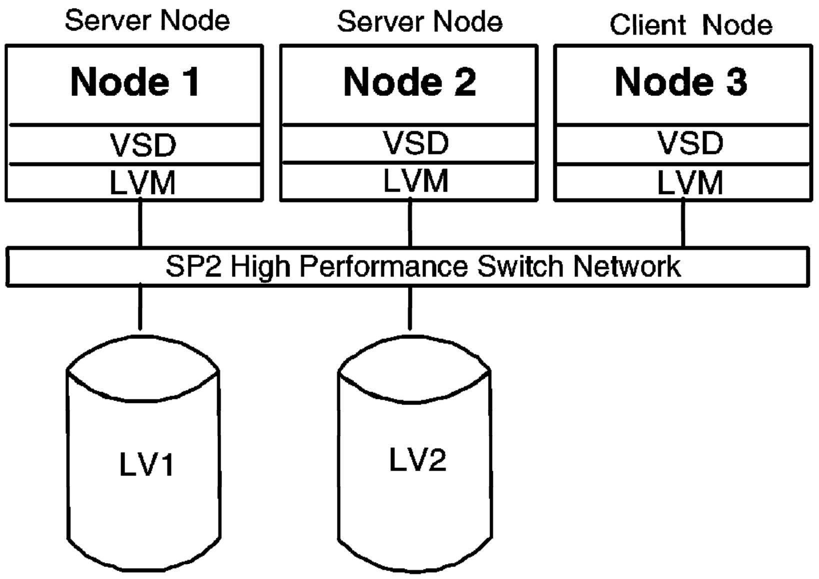


Figure 24. Virtual Shared Disks on Nodes of an SP

For most applications, the increased performance of the HPS would be required since remote I/O across this connection could be a significant factor in application performance. However, the VSD software will function perfectly over a LAN. The latter could be used during development, to set up and test an application or in the event of a failure of the HPS network.

Any application that uses the VSD software must typically provide its own distributed lock management. The VSD software does not provide this. Without a lock manager of some kind, applications or users accessing data from one node could potentially corrupt data that was simultaneously being accessed from another node.

9.8.2 When Can the Virtual Shared Disk Software Be Used?

The VSD software allows AIX logical volumes to be made available transparently across a number of SP nodes. There is no distributed locking or synchronization taking place between these VSDs; some kind of synchronization mechanism would almost always be required.

One commonly-used application that exploits the IBM VSD software is the Oracle Parallel Server database. It uses VSDs to allow Oracle tables to be accessed transparently by all nodes that are being used within the parallel database. A particular Oracle table will actually reside on one specific node, but will appear to be on all nodes. The logical volume, or raw device, containing the database table will be made available to all other nodes that need access, in addition to the server node. The Oracle Parallel Server database software provides its own distributed lock manager to synchronize individual node′s read and write accesses to the VSDs.

As discussed earlier, an AIX journaled file system should not be created on a logical volume that is to be used as a VSD device. However, any application that chooses to use this mechanism can do so. An example might be an application that is used to make logical volumes that are on an SP node without an attached tape drive available to another node that does have a local high-speed tape device. Remote backup using the excellent throughput of the SP HPS could then be achieved. Such an application is likely to be read-only and would be a suitable contender for using VSDs.

9.8.3 How to Manage IBM Virtual Shared Disks

This section describes how to set up and manage VSDs. An example is used to make the process more clear.

As you will have seen from Chapter 1, “System-Management Project Environment” on page 3, the SP nodes 7, 8, 9 and 11 in our environment are in group D, the database group. We will use those nodes for our VSDs in this example. Nodes 7 and 8 will be client nodes and nodes 9 and 11 will be server nodes.

Later on, we will set up a high-availability environment for our VSDs, but for the moment, we will not worry about that.

There are a number of simple steps that we must follow to define our VSDs and get them working. This is not a difficult task when dealing with a small number of VSDs. In practice of course, you may have a large number to define; this would be more time-consuming.

For the most part, the steps required to set up and initialize the VSDs can be performed from the SMIT menus (throughout the discussion, we illustrate our examples using SMIT panels). Equivalent commands can be run on the command line, however. Since all of the steps can be run from the command line, it is possible to create script files that can create and set up large numbers of VSDs with a predefined naming policy.

To create our four VSDs, we follow these steps, each of which is discussed in more detail below:

1. Install necessary VSD software.
2. Determine the location and naming convention for our AIX logical volumes.
3. Define and set up these logical volumes.
4. Determine the naming convention for our VSDs.
5. Define and set up these VSDs on the Control Workstation. These definitions will be stored in the System Data Repository (SDR).
6. Configure the VSDs on all nodes.
7. Start the VSDs on all nodes.

After these steps are completed, our VSDs should be working, and we can then test them. Figure 25 on page 114 shows how the VSDs will be defined on our SP.

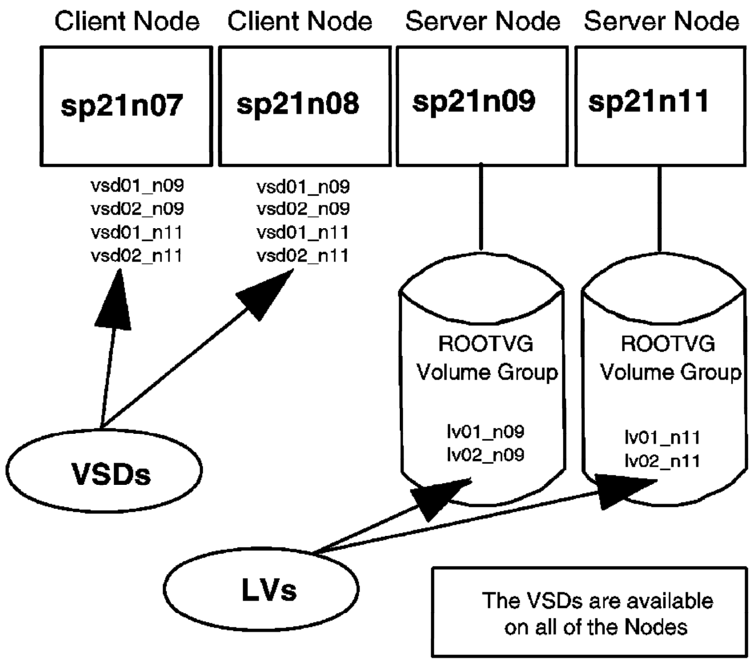


Figure 25. Virtual Shared Disk Layout on Our SP

9.8.3.1 Install VSD Software

In our case, the VSD software that we are using is located in the `/usr/sys/inst.images/ssp` directory on our Control Workstation. Because the Control Workstation will not actually be accessing the VSDs, we must install different software on the Control Workstation than on the SP nodes. The Control Workstation will, however, store all of the definitions in the SDR.

To install the software needed on the Control Workstation, the following SMIT panels are traversed:

```
smitty

└─► Software Installation and Maintenance
    └─► Install / Update Software
        └─► Update Selectable Software
            └─► Install Software Products at Latest Available Level
```

We select the `/usr/sys/inst.images/ssp` directory as our input directory, as this is where our software installable images are stored.

Pressing PF4 lists the installable software in this directory.

To install the required VSD software for the Control Workstation we select:

- 1.2.0.0 csd.vsd
- 1.2.0.0 csd.cmi
- 1.2.0.0 csd.hsd

The csd.vsd software contains the actual VSD software, while the csd.cmi software contains the SMIT menus used on the Control Workstation. At this stage, we do not need to install the csd.hsd software. However, since we will be using it in our discussion of HSDs (see 9.8.5, "IBM Virtual Shared Disk Data-Striping Devices (HSD) Overview" on page 124), we may as well install it now.

When it comes to installing the required software on our four VSD nodes, we get lucky! Since we are on an SP, we can use the **dsh** command to run the same install command on all of the nodes at the same time. To do so, we must:

1. Mount the `/usr/sys/inst.images/ssp` directory containing the software on all four nodes by running the following command:

   ```
   dsh -w sp21n07,sp21n08,sp21n09,sp21n11
   /etc/mount sp21cw0:/usr/sys/inst.images/ssp /mnt
   ```

2. Run the **dsh** command to actually install the required software on the nodes, following the format:

   ```
   dsh -w sp21n07,sp21n08,sp21n09,sp21n11 /etc/installp
   -acXd/mnt/csd.inst.images/ssp/mnt/1.2.0.0 csd.vsd,    \
   1.2.0.0 csd.hsd
   ```

   Note that the above command does not include installation of the csd.cmi software. The Control Workstation is used for running the VSD SMIT panels; the other nodes do not need the code.

All of the software that we need for the moment has now been installed.

9.8.3.2 Logical-Volume Naming Convention

As you probably know, AIX logical volumes actually contain the data. In the case of a database, for example, the database tables reside in these logical volumes. An AIX logical volume can span a number of physical disks as long as the disks are in the same AIX volume group. In other words, each logical volume can be located on only one SP node. In addition, any AIX logical volume can have more than one copy. We can, therefore, provide mirror copies of the data for higher availability. This topic is be discussed in more detail in 9.8.8, "IBM Recoverable Virtual Shared Disk Overview" on page 126.

In the following example, we create four logical volumes, two each on server nodes 9 and 11. Table 12 on page 116 shows our logical-volume naming convention.

For clarity, our naming convention contains the logical volume number and the number of the node on which it resides. We wish to create these logical volumes in the default `rootvg` volume group.

In real life, it would be more common for VSDs to access logical volumes that are not in the rootvg volume group. The rootvg volume group is the default volume group and is usually located on an SP node′s internal disk. It is recommended that customer data be stored on external disks rather than on internal ones. The reason for this is that, in the event of the failure of a node, the data on external disks can potentially still be accessed by making this external volume group available on a node other than the failed one. The external disks have their own power supply and can be be made available independent of the node. This approach can help to provide high availability for SP applications and data and is discussed further when we use the Recoverable VSD software (see 9.8.8, "IBM Recoverable Virtual Shared Disk Overview" on page 126).

Table 12. SP Logical Volumes to Be Used as VSDs

Node Name	Logical Volume Name	Volume Group Name
sp21n09	lv01_n09	rootvg
sp21n09	lv02_n09	rootvg
sp21n11	lv01_n11	rootvg
sp21n11	lv02_n11	rootvg

9.8.3.3 Define and Set Up Logical Volumes

We will now create the first AIX logical volume, on node 9, using the standard method. After logging into node 9 (hostname sp21n09), we traverse the following SMIT panels:

```
smitty

└─► Physical and Logical Storage
    └─► Logical Volume Manager
        └─► Logical Volumes
            └─► Add a Logical Volume
```

We enter information into the SMIT panel, as illustrated in Figure 26 on page 117, selecting rootvg as the volume group in which our logical volume will be created and defining the logical volume name and number of partitions. To get the VSD working initially, we are creating a small logical volume of just two partitions in size. For the moment, we can take the defaults for all other settings.

This process is repeated on node sp21n09 for the other logical volume, lv02_n09, and on server node sp21n11 for the remaining logical volumes, lv01_n11 and lv02_n11.

At this stage, we should be able to list the logical volumes on each of the two server nodes and see ours among them.

9.8.3.4 Naming Convention for Our VSDs

VSDs can be given any name. But, since that name will be seen by all nodes, it must be unique. We have used unique logical volume names in our example as well, to more easily identify where they reside (especially helpful when a large number of VSDs are defined). However, logical volume names need not be unique, as each is in a different volume group on different nodes.

```
                              Add a Logical Volume

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

 TOP                                                    Entry Fields
  Logical volume NAME                                   lv01_n09
* VOLUME GROUP name                                     rootvg
* Number of LOGICAL PARTITIONS                          2
  PHYSICAL VOLUME names
  Logical volume TYPE
  POSITION on physical volume                           midway
  RANGE of physical volumes                             minimum
  MAXIMUM NUMBER of PHYSICAL VOLUMES
      to use for allocation
  Number of COPIES of each logical                      1
      partition
  Mirror Write Consistency?                             yes
  Allocate each logical partition copy                  yes
      on a SEPARATE physical volume?
 MORE...9

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

Figure 26. Adding a Logical Volume

Again, in our example we wish to create four VSDs that will be available on all four SP nodes in our Database Group D. Table 13 shows our VSD naming convention.

Table 13. VSD Naming Convention

Node Name	Logical Volume Name	Volume Group Name	Global Volume Group Name	VSD Name
sp21n09	lv01_n09	rootvg	n09_rootvg	vsd01_n09
sp21n09	lv02_n09	rootvg	n09_rootvg	vsd02_n09
sp21n11	lv01_n11	rootvg	n11_rootvg	vsd01_n11
sp21n11	lv02_n11	rootvg	n11_rootvg	vsd02_n11

We have already defined the logical volume names. You will notice that we have defined a unique name for each volume group called the *Global Volume Group Name*. This name must be unique among the nodes that will be VSD servers or clients. In our case, these are nodes 7, 8, 9 and 11.

We are now ready to input this data into the System Data Repository on the Control Workstation.

9.8.3.5 Define and Set Up VSDs on the Control Workstation

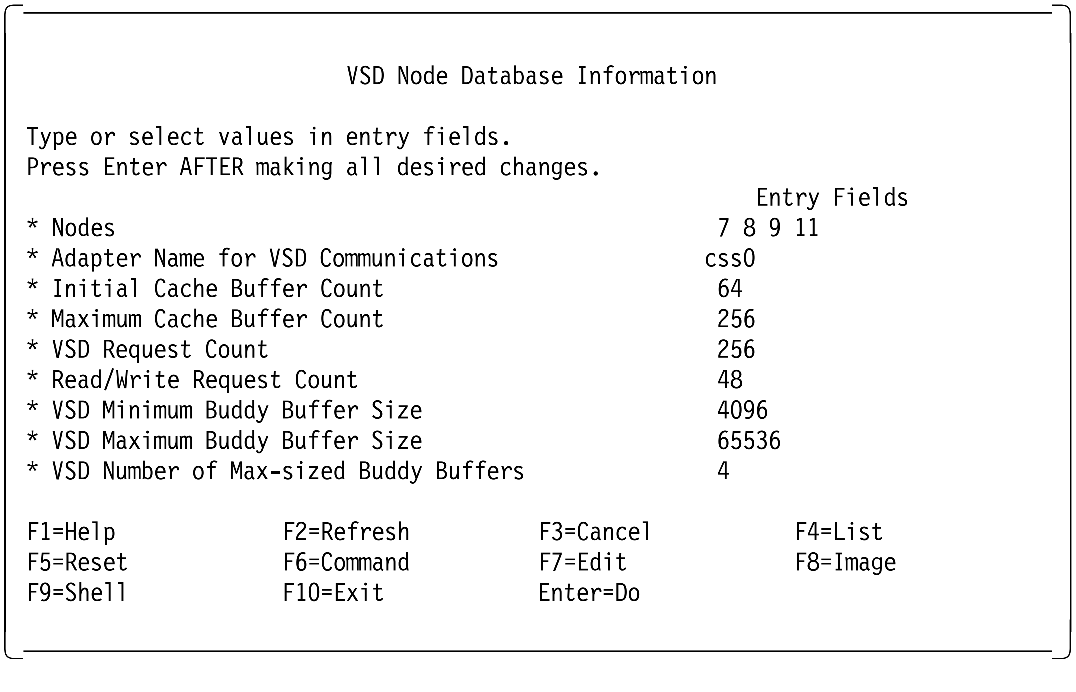
We can now enter the definitions of our VSDs into the Control Workstation, where the information will be stored in the SDR. This information will be transferred to the actual VSD nodes later. Defining VSDs is a three-step process; we must:

1. Input VSD Node Information
2. Input VSD Global Volume Group Information
3. Define Virtual Shared Disks

VSD Node Information To enter the node information, we traverse the following SMIT panels:

```
smitty

└─► 9076 SP System Management
    └─► 9076 SP Configuration Database Management
        └─► Enter Database Information
            └─► VSD Database Information
                └─► VSD Node Information
```

We can now enter our information in the SMIT panel shown in Figure 27.



```
                         VSD Node Database Information

Type or select values in entry fields.
Press Enter AFTER making all desired changes.
                                                         Entry Fields
* Nodes                                               7 8 9 11
* Adapter Name for VSD Communications                css0
* Initial Cache Buffer Count                          64
* Maximum Cache Buffer Count                          256
* VSD Request Count                                   256
* Read/Write Request Count                            48
* VSD Minimum Buddy Buffer Size                       4096
* VSD Maximum Buddy Buffer Size                       65536
* VSD Number of Max-sized Buddy Buffers               4

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

Figure 27. Sample VSD Node Database Information

We can select our four nodes from the list produced when we press PF4. Notice that we have chosen to use the High Performance Switch (css0), but we could equally have selected Ethernet (en0). We can take the defaults for all other values.

We have successfully defined these four nodes in the VSD group of nodes.

VSD Global Volume Group Information: To enter the Volume Group information, we traverse the following SMIT panels:

```
smitty

└─► 9076 SP System Management
    └─► 9076 SP Configuration Database Management
        └─► Enter Database Information
            └─► VSD Database Information
                └─► VSD Global Volume Group Information
```

For our first node, we define the Local Volume Group Name as rootvg and the Primary Node on which the Volume Group resides is sp21n09. (Pressing PF4 produces a list from which we can select). Per our naming convention, its global volume group name is n09_rootvg. We can leave the Secondary Node on which the Volume Group resides as blank for the moment. Figure 28 shows the SMIT panel containing these entries.

```
                       VSD Global Volume Group Information

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

                                                          Entry Fields
* Local Volume Group Name                              rootvg
* Primary Node on which Volume Group is Resident       sp21n09
  Secondary Node on which Volume Group is Resident
  Global Volume Group Name                             n09_rootvg


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

Figure 28. Sample VSD Global Volume Group Information

We must repeat this step one more time as we have two nodes and two volume groups. For our second definition, we use the rootvg volume group on node sp21n11. Its global name is n11_rootvg.

Once we have entered this data, we can move on to the final step in defining the SDR data related to our VSDs.

Define a Virtual Shared Disk: To enter this information, we use the following SMIT panels:

```
smitty

└─► 9076 SP System Management
    └─► 9076 SP Configuration Database Management
        └─► Enter Database Information
```

```
                    └─► VSD Database Information
                        └─► Define a Virtual Shared Disk
```

For our first VSD, we define the Logical Volume Name as lv01_n09, the Global Volume Group Name as n09_rootvg (pressing PF4 produces a list from which we can select) and the Virtual Shared Disk Name as vsd01_n09.

The SMIT panel appears in Figure 29.

```
                          Define a Virtual Shared Disk

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

                                                          Entry Fields
* Logical Volume Name                                  lv01_n09
* Global Volume Group Name                             n09_rootvg
* Virtual Shared Disk Name                             vsd01_n09
  Virtual Shared Disk Option                           nocache


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

Figure 29. Sample VSD Definition Panel

We must repeat this step three more times to define all four of our VSDs.

We have now entered all of the required information into the SDR on the Control Workstation. We can check our definitions for accuracy by listing and viewing them using the SMIT panels, as follows:

```
smitty

└─► 9076 SP System Management
    └─► 9076 SP Configuration Database Management
        └─► List Database Information
            └─► List VSD Database Information
```

Note that a similar SMIT panel allows us to delete VSD definitions in the SDR should we choose to do so.

9.8.3.6 Configuring the VSDs on the SP2 VSD Nodes

In fact, we do not need to use this command directly (although we could). We can use the SMIT *cfgvsd* panel to configure SP nodes with the VSD information. This command, in turn, invokes the dsh command.

To configure all of our VSD nodes, we use the following SMIT panels:

```
smitty

└─► 9076 SP System Management
    └─► 9076 SP Cluster Management
        └─► VSD Management
            └─► Configure a VSD
```

Note that the fast path for this panel is smit configvsd_dialog. We can take the default of All_VSDs here, although, in practice, we may want to specify particular VSDs or nodes by selecting them from the PF4 list. Selecting All_VSDs runs the VSD configuration command on all of the nodes, actually creating a full set of definitions of the VSD data to be stored on each SP node. This information is stored in AIX files in the directory /var/adm/csd/vsdfiles. This links to the /usr/lpp/csd/vsdfiles directory, which contains files describing all of our VSDs. If we change definitions on the Control Workstation later on, we must update these files, as well. Again, the cfgvsd command comes in handy for that. All the changes we made to the VSDs in the SDR will not be effective until we issue cfgvsd.

Now that our VSDs are set up and ready to go, all we need do now is start them!

9.8.3.7 Starting the VSDs

Traversing the following SMIT panels, we can start our VSDs:

```
smitty

└─► 9076 SP System Management
    └─► 9076 SP Cluster Management
        └─► VSD Management
            └─► Start a VSD
```

Once again, we can take the default of All_VSDs here, although, in practice, we may want to specify particular VSDs or nodes by selecting them from the PF4 list.

Alternatively, we can start all nodes from the command line by issuing the command **startvsd -a**. We can start VSDs on particular nodes using the **startvsd** command with that node′s name as the argument.

9.8.3.8 What Can We Do with These VSDs?

Our four VSDs should now be up and running. To verify if they are working, use the following command:

```
·root@sp21n10“ /usr/lpp/csd/bin >lsvsd -l
minor  state server lv_major lv_minor  vsd-name                         option
 1     ACT      9        0        0      vsd01_n09_n10                    nocache
 2     ACT      9        0        0      vsd02_n09_n10                    nocache
 3     ACT     11       25        1      vsd01_n10_n09                    nocache
 4     ACT     11       25        2      vsd02_n10_n09                    nocache
```

The output of the command shows that all the four VSDs are active (state ACT), meaning they can be read and written to. So we accomplished pretty much what we set out to do. But for the sake of completeness, we should also perform some tests on those VSDs.

We can use a test script provided with the VSD software. The test script, vsdvts, can be found in the directory /usr/lpp/csd/bin. Redbook GG24-4344 shows another way of testing the VSDs using directly the dd command.

We must run the command vsdvts on the VSD nodes, preferably on two nodes, a server node and a client node. The command **vsdvts vsd01_n09** produces the output shown in Figure 30.

```
Step 1: Writing the vsdd to the VSD via:
 dd if=/usr/lib/drivers/vsdd of=/dev/rvsd01_n09
Step 1: Success!  Wrote vsdd to /dev/rvsd01_n09   (98).
Step 2: Reading data just written to the VSD via:
  dd of=/tmp/vsdvts1.15588 if=/dev/rvsd01_n09 count=98
Step 2: Success!
Step 3: copying vsdd via:
  dd of=/tmp/vsdvts2.15588 if=/usr/lib/drivers/vsdd count=98
Step 3: Success!
Step 4: Verifying data read from the VSD via:
  diff /tmp/vsdvts1.15588 /tmp/vsdvts2.15588 2>&1 > /dev/null
Step 4: Success!


VSD Verification Test Suite Successful!
```

Figure 30. Testing a VSD Using VSDVTS Script

9.8.3.9 Managing our VSDs

There are a number of commands that we can use to manage our VSDs. These commands can be run on the command line of specific nodes. Or, we can use the SMIT panels on the Control Workstation to *dsh* these commands to some or all nodes.

The commands used to manage VSDs are:

defvsd	Enters VSD information into the SDR. Can be used instead of keying data into a SMIT panel and is particularly helpful when defining large numbers of VSDs.
cfgvsd	Loads the VSD device driver, tells it about the VSDs, and creates the special files in /dev if they are not already there (from a previous configuration). cfgvsd leaves the VSDs in the stopped state.
preparevsd	Changes a VSD from the stopped state to the suspended state. The VSD is not running after command execution, but is available and ready to be started.
resumevsd	Activates an available VSD. The VSD is running after command completion.

startvsd	Equivalent to running a preparevsd followed by a resumevsd.
suspendvsd	Deactivates a VSD, leaving it suspended.
stopvsd	Stops a VSD and leaves it unavailable, but still defined.
ucfgvsd	Unloads the VSD device driver, and removes it from the kernel.
undefvsd	Deletes VSD definitions from the SDR database.

Note that I/O requests to a VSD that is not available will fail. I/O requests to a VSD that is suspended will be queued. Keep this in mind when we review how the Recoverable VSD can cope with node failures (see 9.8.8, "IBM Recoverable Virtual Shared Disk Overview" on page 126).

Figure 31 shows the various VSD commands that we can run and their effect on the status of our VSDs.

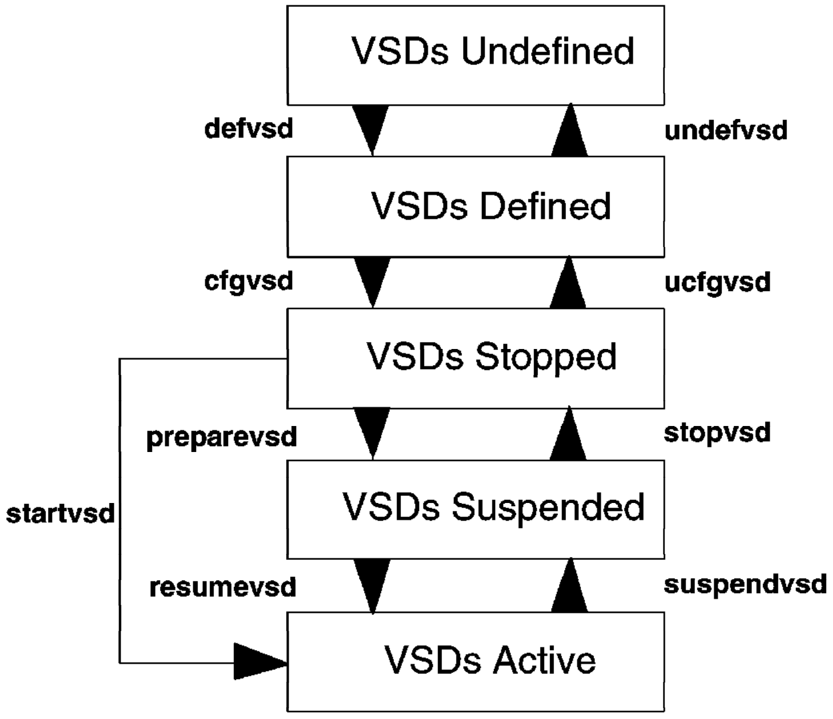


Figure 31. VSD Commands and Their Effect

Note: To do all of the VSD management tasks illustrated in Figure 31, you should be logged in as the root user. In addition, if you wish to *dsh* your commands to other nodes, you will need Kerberos authority to use *rsh*. You will, therefore, need a valid Kerberos ticket. For further details on this subject, see Chapter 4, "Kerberos" on page 39.

9.8.4 IBM Virtual Shared Disks - Practical Experiences

A particular problem that we experienced in our lab, one that was not obvious to us at first, is actually described in the VSD README file. So, in the end, we were forced to read the documentation!

By default, the VSD software uses an `IP_Protocol` value of 63. If the PTF for IX42338 is not installed on your system, this default value causes node-to-node VSD communications to fail. If this PTF is not installed, you must use a different value for the `IP_Protocol`. The README file suggests a value of 22. This must be changed in the `/usr/lpp/csd/bin/cfgvsd` file on every VSD node, as well as the Control Workstation. After making these changes, our VSDs worked fine.

9.8.5 IBM Virtual Shared Disk Data-Striping Devices (HSD) Overview

The IBM Virtual Shared Disk Data Striping Device is an additional software product that can be used in conjunction with the VSD software that has already been described.

It provides the ability to stripe a device (called a hashed shared disk or an HSD) across a number of existing VSD devices. This might be useful in helping to distribute I/O activity across a number of VSD devices. In the case in which a “hot spot” (a large volume of I/O activity to one VSD and, hence, one disk or group of disks) is occurring, it may be possible to distribute the I/O over a larger number of disks by using an HSD.

HSDs are defined and set up after the VSDs are already in place.

The HSD driver is actually a pseudo device-driver layer that is above the VSD layer. The same limitations that apply to VSDs apply to HSDs. In particular, an AIX filesystem cannot be mounted on an HSD.

Figure 32 on page 125 illustrates how an HSD can be defined across a number of existing VSDs.

An HSD is designed to improve performance for I/O in certain situations. It will not improve availability and, in fact, as more disks are accessed, the vulnerability to disk failure is increased. Disk mirroring or RAID disk technology can protect from that kind of failure, of course.

9.8.6 How to Manage IBM Virtual Shared Disk Data Striping Devices

Once the base VSDs are running successfully, defining the HSDs is a simple task.

Remember that we installed the HSD software on the nodes at the same time we installed the VSD software (see 9.8.3.1, “Install VSD Software” on page 114). We can, therefore, go ahead and define our HSDs. In our case, we want to set up one HSD to span our four VSDs. In practice, an exercise would be undertaken to determine any hot spots on our existing VSDs and, using that information, an HSD outline plan could be developed.

Note: We must not use existing VSDs that contain data. If we have VSDs with data, we should define new ones for the HSD and copy the data across.

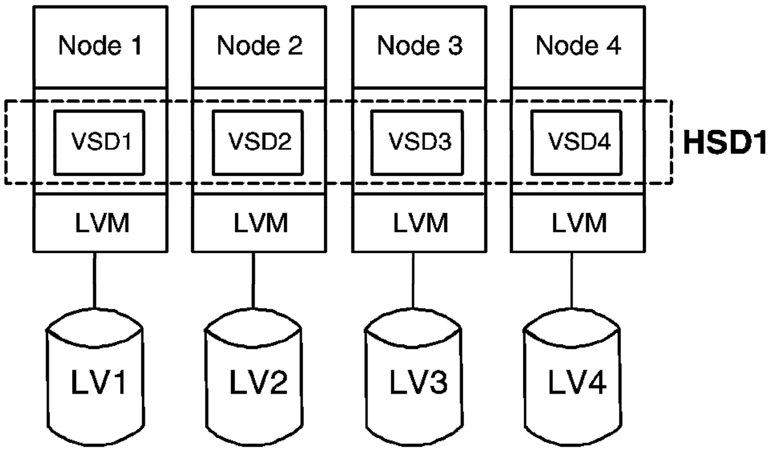


Figure 32. Virtual Shared Disk Data Striping Device (HSD)

The SMIT panels for defining our HSDs on the SP Control Workstation are accessed as follows:

```
smitty

└→ 9076 SP System Management
    └→ 9076 SP Configuration Database Management
        └→ Enter Database Information
            └→ VSD Database Information
                └→ VSD Database Information
                    └→ Define a Hashed Shared Disk
```

At this point, we define:

- An HSD Name of hsd1
- A stripe size of 4096 (bytes)
- The underlying VSDs, in our case, all four of the VSDs

We have now defined our HSD. We can list our HSD through the SMIT panels. We can also delete (or undefine) HSDs using SMIT. All of these tasks can, of course, be achieved through command-line options.

Our underlying VSDs should all be operational before we configure and start our HSD. To get our HSD up and running, we use the cfghsd command, found in the /usr/lpp/csd/bin directory. It must be run on each node, making the cfghsd command a prime candidate for a *dsh* from the Control Workstation.

Once completed, we should have working HSD devices on each VSD node. We can test our HSD devices with a **dd** command similar to that used for VSD testing. If we run the command **hsdvts hsd1** while in the directory `/usr/lpp/csd/bin`, we should see successful output from this test script.

9.8.7 IBM Virtual Shared Disk Data Striping Devices - Practical Experience

We tried running some I/O activity on the HSD devices. In the case of small files, the overhead of "striping" the I/O appeared to result in slower overall times to complete the I/O. In the case of large files that are sequentially read or written (more typical of a technical/scientific application than a commercial one), we would expect to see performance improvements from striping the data using HSDs.

There are many parameters that can be tuned for both VSD and HSD that have a significant impact on performance. Therefore, our simple testing should not be deemed conclusive.

9.8.8 IBM Recoverable Virtual Shared Disk Overview

The IBM Recoverable Virtual Shared Disk (RVSD) software works in conjunction with the standard VSD software to provide high availability for logical volumes and data.

In essence, using RVSD, each VSD server node has a "standby" node that will take over and run when required. Client nodes need not be protected, as another client node can potentially take over from a failed client node, accessing the data from the server node via a VSD. So, only server nodes that are "serving" the VSD logical volumes must be protected. Figure 33 on page 127 shows how two server nodes could be used to provide high availability for logical volumes that are in use as VSDs.

In this case, the node on the left is the primary node and will normally have access to the data in the external volume group. The external volume group (extvg1) will normally be active (varied on) to the primary node. At this time, the secondary node will not, and cannot, have access to the logical volumes that are in this volume group. In the event of the failure of the primary node on the left, the secondary node on the right will take over ownership of this volume group and will access the data, replacing the failed node.

The volume group in this case is known as a *twin-tailed* volume group. The disks contained in this volume group are cabled into both nodes. Although it may be called a shared volume group, it is never accessed from both nodes concurrently; only one node or the other has live access.

Our Recoverable VSD software must be able to do a number of tasks for us:

- Detect that a failure has occurred
- Notify other nodes of the failure
- Take corrective action to allow access to data from another node
- Maintain an updated understanding of the status of VSDs

The RVSD software, working in close cooperation with other standard AIX software, performs these tasks, allowing uninterrupted access to data even during a node failure.

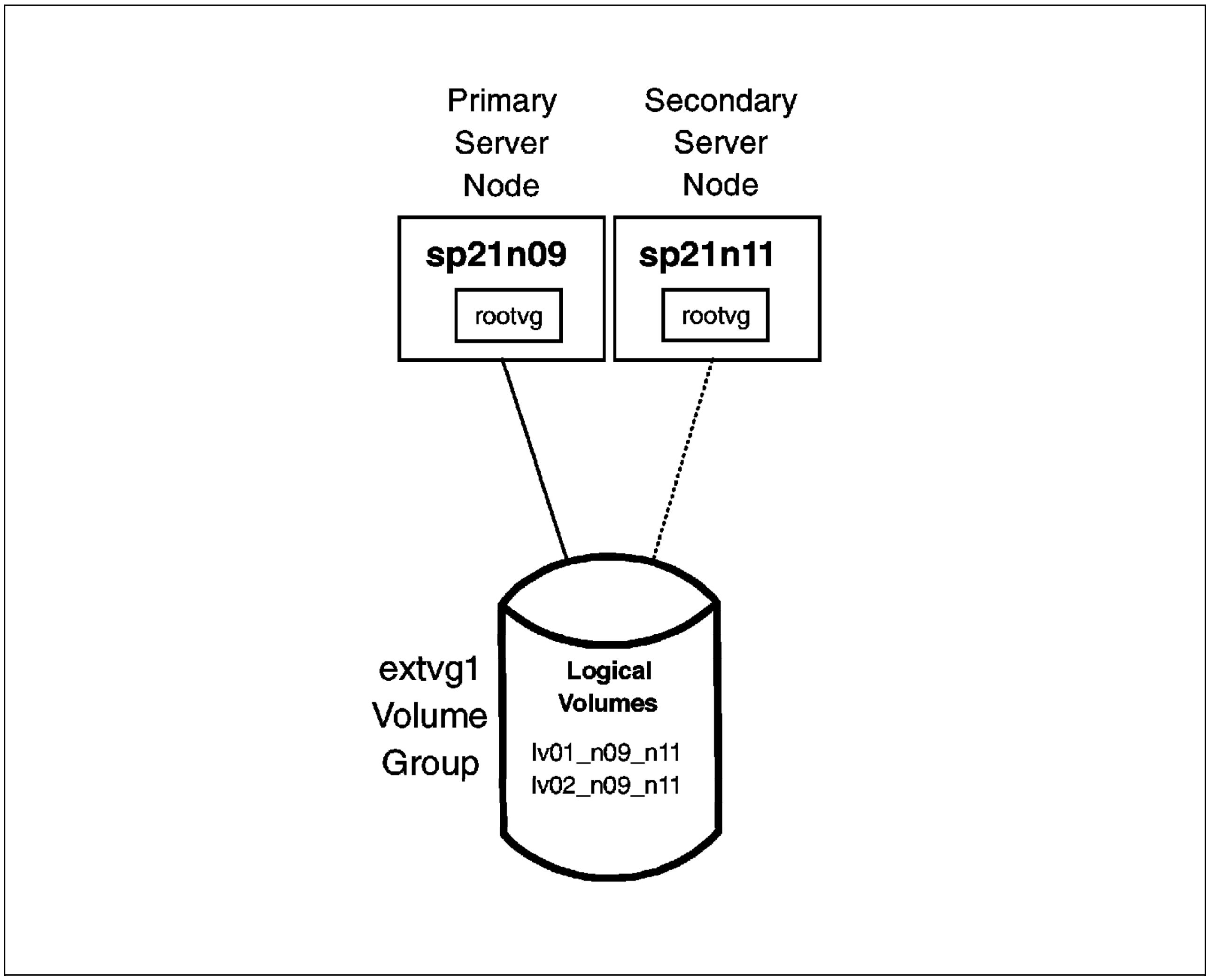


Figure 33. Server and Secondary Nodes for RVSD

Following node failure, the secondary node takes over. It varies on the twin-tailed volume group, making the VSDs available and active once again. No human or manual intervention is required, and node takeover is completely transparent to an application. If a node should fail, any affected VSDs are suspended until takeover has occurred. As I/O is pending while a VSD is suspended, applications should only see a short delay and no I/O or data should be lost.

Some of the function used by the RVSD software is provided by existing PSSP daemons that run on the SP Control Workstation and the SP nodes, including the:

Heartbeat daemon A user-level daemon running on each node of the SP, including the Control Workstation. It exchanges heartbeats with its peers on other nodes over the network. If a node fails to successfully transmit a few heartbeats for any reason, the other heartbeat daemons will declare it dead and the RVSD software will step in.

This daemon is known as the *hb* process. It is reset, quiesced and resumed using *hb* commands. The daemon is started at system boot time on each node (in

/etc/inittab) and will respawn normally if killed. This daemon is unaware of VSDs. It communicates over the Ethernet.

Recovery Daemons Known as the *ha*, the daemons are notified if the *hb* daemon should fail. The *ha* daemon is aware of VSD definitions and can determine what corrective action needs to be taken. It invokes the recovery scripts when a failure occurs. The *hc* daemon notifies the application using the VSD that needs to know about failures. For example, if a system uses Oracle, the **hc** daemon would notify the recoverable distributed lock manager of a failure so that Oracle could take any necessary corrective action.

The components of the RVSD, running on three nodes, are shown in Figure 34.

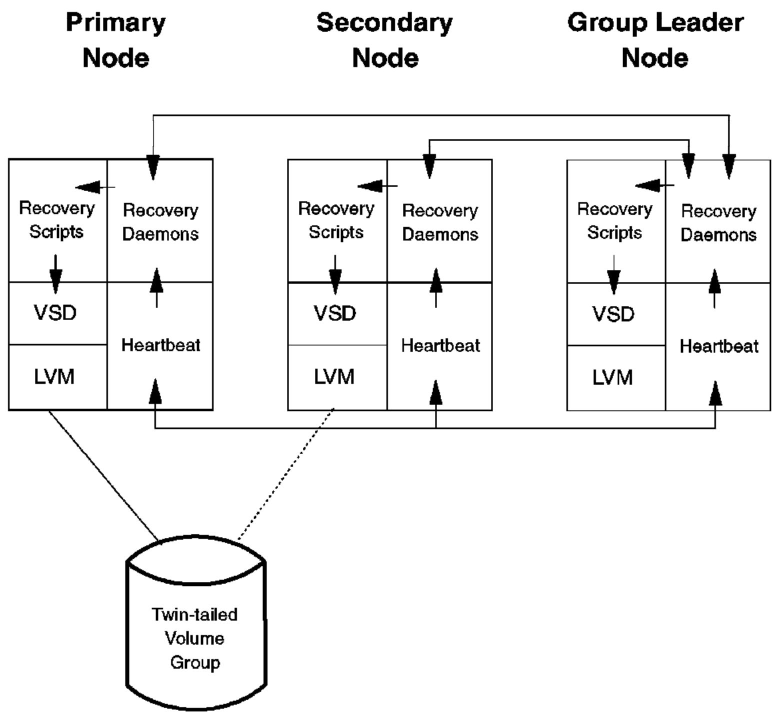


Figure 34. Components of the Recoverable VSD

The Recoverable VSD daemons designate two of the SP nodes as special nodes:

1. Group Leader (GL)
2. Crown Prince (CP)

The group leader is a special node only in the internal design of the hb, ha and hc daemons. For heartbeat (hb), the GL is the node with the highest numerical IP address of the group of nodes. For the ha and hc programs, the GL is the node with the highest node number, as defined in the SDR on the Control Workstation. Note that the ha/hc GL node does not have to be the same as the hb heartbeat GL node.

The heartbeat daemon also has a notion of the crown prince, which is the next-highest-numbered node; it takes over if the group leader fails. If both GL and CP fail at the same time, heartbeat will temporarily suspend, all nodes will perform a self-down, then they will all merge back together to form a group.

So for heartbeat, a failure of the GL node is generally harmless as long as the CP node remains up for the duration of the membership change and propagation of this configuration-change information to the rest of the group. After which, the former CP is designated as the new GL and the next-highest-numbered node becomes the new CP.

Additional failures during this period are not a problem. Only failure of the GL in the middle of the membership-change propagation process will be disruptive and may cause a self-down of nodes.

For ha and hc, failure of the GL node is also likely to be harmless, unless the failure occurs while the GL is actively handling previous membership changes.

Note that multiple simultaneous failures, even those involving the GL, will not cause any problems because heartbeat will report them consistently to all of the nodes. The remaining nodes can detect if they are receiving membership information from heartbeat that they did not get from their GL and will only self-down when they cannot determine what has happened.

While you need something akin to a double failure to be affected by a GL failure, the nature of the second failure is quite different in each of these three programs.

In our case, we used the 9333 serial disks for the RVSD. (SCSI disks could also be used. A RAID disk subsystem could provide disk redundancy to protect the logical volumes and their data). The 9333 disks provide many benefits including their high performance characteristics and their advantage in a high-availability environment. These disk towers or drawers can be concurrently attached to up to eight systems or SP nodes. In addition, the disks themselves are “hot-pluggable.”

9.8.9 How to Manage IBM Recoverable Virtual Shared Disk

We will now set up some Recoverable VSDs on our SP nodes. The simple ones we defined earlier will not be of any use, as they are in the `rootvg` volume group which cannot be successfully twin-tailed.

Assuming that we do not yet have our base VSDs defined, the steps required to get our RVSDs working include:

1. Install the VSD and RVSD software.
2. Determine the location and naming convention for AIX logical volumes that will contain the data.
3. Define and set up these logical volumes on twin-tailed volume groups.
4. Determine the naming convention for the RVSDs.

5. Define and set up these RVSDs on the Control Workstation. These definitions will be stored in the SDR.

6. Configure the RVSDs on all nodes. This creates all of the necessary files on each node.

7. Start the RVSDs on all nodes.

9.8.9.1 Install the VSD and RVSD Software

We looked earlier at how to install the VSD software. We install the RVSD software in a similar manner. In this case, however, the RVSD software is installed on the nodes only, not on the control workstation. The following components of the RVSD licensed program product are installed on all SP VSD nodes (clients and servers):

rcvsd.vsd	Contains the main RVSD software
rcsd.ha	Contains the RVSD ha recovery daemon software
rcsd.hc	Contains the hc daemon

In place of the above components, we install the following on the Control Workstation:

rcsd.tool	Contains the RVSD graphical monitoring tool that enables monitoring of the status of nodes and RVSDs. This component should be installed on the Control Workstation only.
rcsd.docs	Contains the RVSD documentation. It could be installed on any node. In our case, we have chosen the Control Workstation.

9.8.9.2 Determine the Location and Naming Convention for Logical Volumes

We have chosen to place two external volume groups on 9333 disk towers. Our first server node (sp21n09) will be primary for two logical volumes in the extvg1 volume group. The second server node (sp21n11), will be primary for another two logical volumes in a second external volume group, extvg2.

We are going to add one more complication into the equation. We want to mirror our logical volumes as added prevention in case of disk failure. This could be done when we define our logical volume in the SMIT panel, but we want to mirror to a disk that is housed in a separate 9333 disk tower (cabinet). This would allow for a power failure to a disk tower also. This is discussed in more detail in 9.8.10, “IBM Recoverable Virtual Shared Disk - Practical Experiences” on page 134.

Our logical volume layout is shown in Figure 35 on page 131.

9.8.9.3 Define and Set Up Logical Volumes

To set this up, we first define the extvg1 volume group on node sp21n09. We define the disks that we wish to use, taking careful note of the volume layout in Figure 35 on page 131.

We start with the extvg1 volume group. In this volume group, we can define both of our logical volumes. When we define and create these through the SMIT panel to add a logical volume, we specify that we want two copies of the logical volume, and each copy should be on a separate disk. We make these logical volumes 128 partitions in size (that is, 500MB). We must ensure that the two disks with the copies are in separate 9333 disk towers.

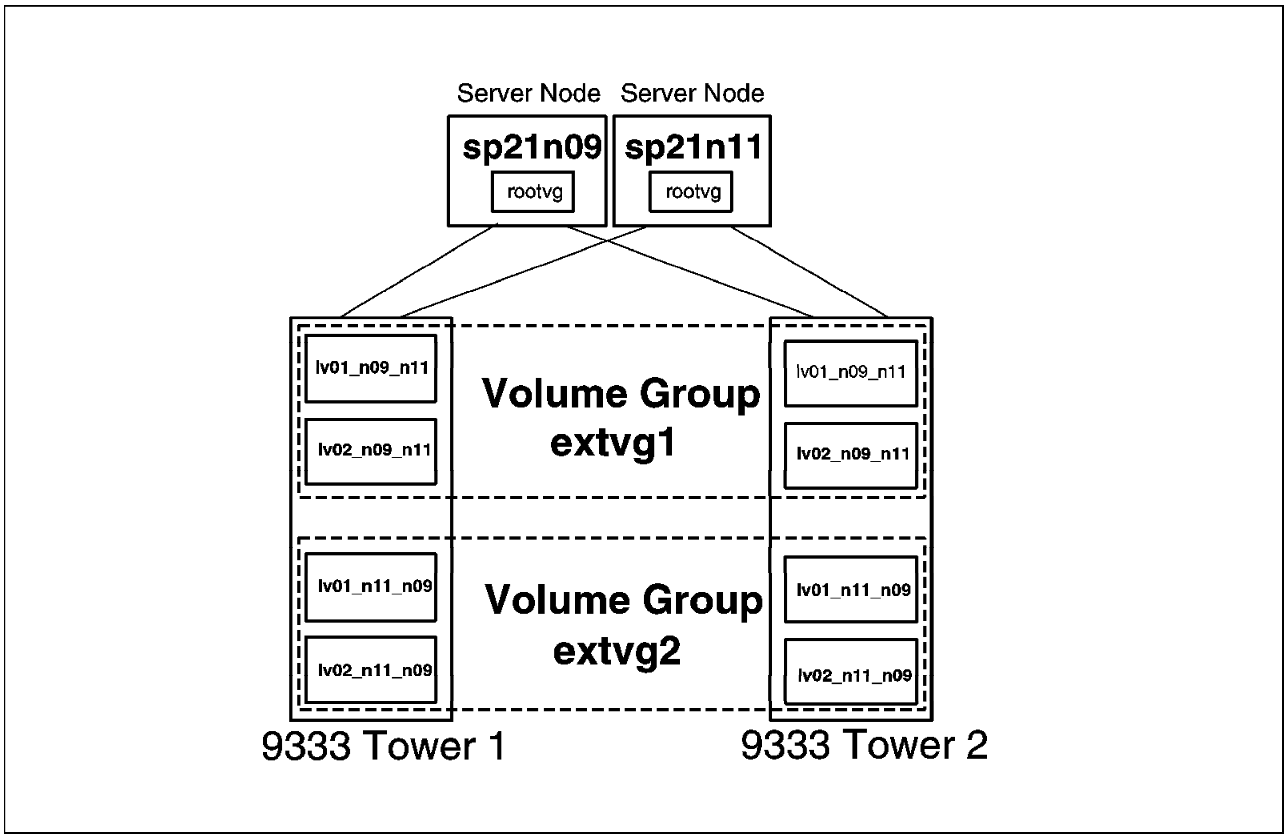


Figure 35. Logical Volume Layout for our RVSDs

The SMIT panel that allows us to define our logical volume appears in Figure 36 on page 132.

Once both of the logical volumes in this volume group are defined, we set the automatic activation of the volume group to “no.” This will prevent the volume group from starting by itself at boot time. We want the RVSD software to control this process for us.

Once work on this first volume group has been completed, we must vary off the volume group and export it. This can be achieved by running **varyoffvg extvg1**. We must also export all of the volume group definitions. This can be done by running **exportvg extvg1**.

We are now able to import this volume group into the secondary node. Its volume group name is still extvg1. It has exactly the same characteristics as on the primary node. Again, we set the automatic activation of this volume group to “no” in the relevant SMIT panel.

We can import the extvg1 volume group into the secondary node (sp21n11) with the command **importvg hdisk4** (where hdisk4 is one of the disks on node sp21n11). Note that the disk numbering on the two nodes may not match, depending on the number of internal disks that would typically be numbered first.

We can vary on the volume group, check all of the definitions and change the automatic activation setting.

```
                       Add a Logical Volume

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

 TOP                                                    Entry Fields
  Logical volume NAME                                   lv01_n09_n11
* VOLUME GROUP name                                     extvg1
* Number of LOGICAL PARTITIONS                          128
  PHYSICAL VOLUME names
  Logical volume TYPE
  POSITION on physical volume                           midway
  RANGE of physical volumes                             minimum
  MAXIMUM NUMBER of PHYSICAL VOLUMES
      to use for allocation
  Number of COPIES of each logical                      2
      partition
  Mirror Write Consistency?                             yes
  Allocate each logical partition copy                  yes
      on a SEPARATE physical volume?
 MORE...9

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

Figure 36. Adding a Logical Volume (RVSD)

This entire process is repeated for the two logical volumes in the other twin-tailed volume group. In this case, our primary node is node 11, and the volume group is extvg2.

9.8.9.4 Naming Convention for our VSDs

We define base VSDs as shown in Table 14.

Table 14. VSD Naming Convention for RVSDs

Node Name	Logical Volume Name	Volume Group Name	Global Volume Group Name	VSD Name
sp21n09	lv01_n09_n11	extvg1	extvg1	vsd01_n09_n11
sp21n09	lv02_n09_n11	extvg1	extvg1	vsd02_n09_n11
sp21n11	lv01_n11_n09	extvg2	extvg2	vsd01_n11_n09
sp21n11	lv02_n11_n09	extvg2	extvg2	vsd02_n11_n09

Our naming convention shows the number of the VSD, the primary node and the secondary node, in that order.

9.8.9.5 Define and Set Up VSDs on the Control Workstation

The process for setting up RVSDs is almost exactly the same as when defining base VSDs. We use the same SMIT panel, but must now also define a primary and secondary node for each Global Volume Group, as shown in Figure 37.

```
                      VSD Global Volume Group Information

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

                                                          Entry Fields
* Local Volume Group Name                              extvg1
* Primary Node on which Volume Group is Resident      sp21n09
  Secondary Node on which Volume Group is Resident    sp21n11
  Global Volume Group Name                             extvg1


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

Figure 37. Sample VSD Global Volume Group Information (*RVSD*)

As you may have noticed, we have chosen a global volume group name that is the same as the local volume group name. This is fine as long as the name is unique.

9.8.9.6 Starting the VSDs

We are now ready to start the RVSDs. We must run the **hb** command with flags to reset the RVSD nodes. Alternatively, we can reboot the nodes. If we are not going to reboot, we should vary off the external twin-tailed volume groups on all nodes. The RVSD software will take care of starting the volume groups for us (on the primary nodes only).

If we are not going to reboot, we should also run the **hb reset** command. This resets the daemon, reestablishing communication with the other RVSD daemons.

9.8.9.7 Managing Our Recoverable VSDs

In most cases, the RVSD manages the VSDs for us. We can manage individual VSDs on specific nodes in the same way as base VSDs or, for multiple nodes, use the dsh command. However, be careful not to interfere with RVSD, especially when a node fails or boots. RVSD will start running its scripts; so, it is best not to touch the VSDs at those times. (Refer to 9.8.3, “How to Manage IBM Virtual Shared Disks” on page 113).

The commands discussed earlier that relate to managing VSDs apply equally here, with a few important exceptions. Do *not* use the following commands:

- **preparevsd**
- **resumevsd**
- **suspendvsd**

There are three flags that we can use with the **hb** command that we may find helpful:

hb reset	“Restarts” the daemon, reestablishing communication with other RVSD daemons.
hb quiesce	Stops the daemon. Normally, the daemon is set to respawn; this command prevents that from happening. Also simulates a node failure as far as RVSD is concerned (we used this for testing node failures without having to power off the nodes). Any failure other than the **hb** daemon is not detected by RVSD.
hb resume	Returns the **hb** to its normal respawning state after **hb quiesce** is run.

RVSD provides a very nice monitor with which to show the status of those nodes with VSDs. This monitor can only be run on the Control Workstation.

The executable to run the monitor can be found in the directory `/usr/lpp/csd/tool`. It is started by issuing the **monvsd** command. A small window is opened, displaying our VSDs. The window cannot be resized.

The **monvsd** window shows us an icon (oval shape) for every node in the SP; VSD nodes are indicated by a blue triangle. If a node dies, the icon turns from green to red. If a node is powered off, the icon turns black. If the oval shape has a segment missing, that indicates the node is not running on the SP Switch.

Mouse buttons allow us to list VSDs on nodes, open rlogin sessions to selected nodes and perform other useful functions.

9.8.10 IBM Recoverable Virtual Shared Disk - Practical Experiences

In our experience, the RVSD software is very powerful and, when used in conjunction with protection of the disks and their data, provides a very useful way of achieving high availability for logical volumes.

Remember, of course, that RVSD only protects logical volumes and not filesystems. However, for a database that is storing its data on logical volumes, RVSD can be very useful. A distributed lock manager that understands what the RVSD is doing is also required. At the time of writing, Oracle is working on a recoverable distributed lock manager for the Oracle Parallel Server Option. Figure 38 on page 135 shows how the Oracle components interface with the RVSD components. Also shown is the relationship between the various IBM components. Those on the right of the dashed line in the diagram are recoverable components, those on the left are base VSD components.

We came across one or two limitations when working with the Recoverable VSD software on our system in the lab. In particular, as described earlier, the complete definition (including the logical volume definitions) of a twin-tailed volume group must be identical on both nodes to which the volume group is attached.

When setting up for the first time, this is not a problem. You can vary off the volume groups at will. However, when the system is live, if you do something such as change the size of a logical volume on the “live” primary node, this change will not be reflected on the secondary node until the volume group on the primary node is varied off and the secondary node is varied on. This would require you to stop any applications that are currently accessing this data on the primary node, a potential problem on a live system. You could create additional new volume groups for changes, but this would not be ideal.

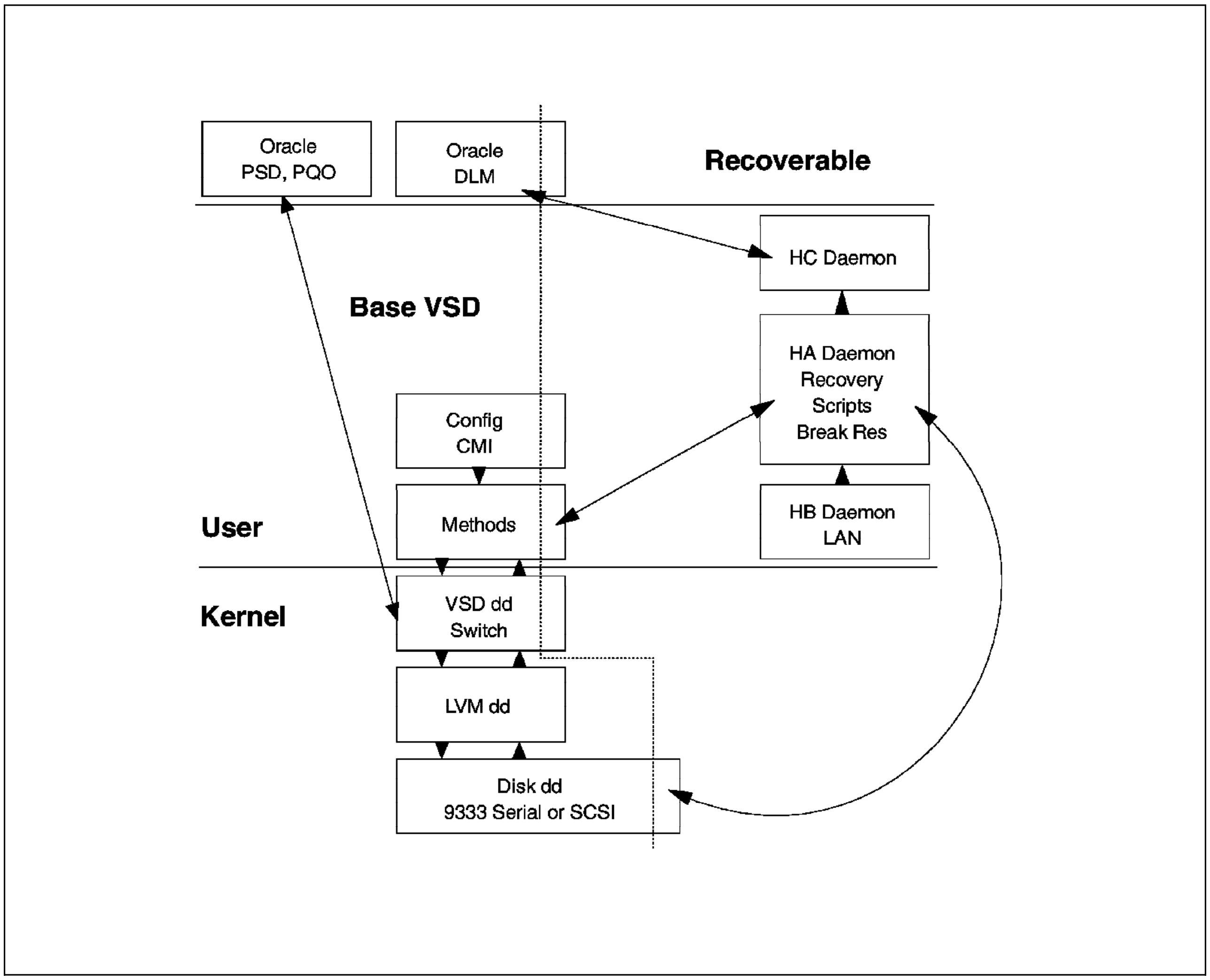


Figure 38. Oracle Working with Recoverable VSDs

The following tasks require careful planning:

- Changing the characteristics of logical volumes
  - Changing the size of a logical volume
  - Adding a logical volume
  - Adding an extra mirror copy to a logical volume
- Changing the characteristics of the shared volume groups
  - Adding an additional disk into a shared volume group
  - Removing a disk from a volume group

An RVSD is dependent on the hb daemon to understand whether a secondary node should take over from the primary. This is not foolproof in that it cannot cater to all instances. For example, if a disk controller should fail, the hb daemon will not detect a failure and automatic takeover would not occur. User scripts could be developed that monitor particular processes, initiating takeover by quiescing the hb daemon as required.

A complete node failure will result in takeover, but partial failures may or may not result in a takeover. For critical applications, a careful study of potential failures and resulting actions would need to be made.

The last minor problem that we encountered is that the monvsd window that displays the status of nodes and VSDs does not display VSDs that are defined to run over Ethernet. The tcl script only contains a search for VSDs connected by SP High Performance Switch (css0). These Switch VSDs are displayed as blue triangles on each SP node where VSDs are defined. The triangles will not be displayed on nodes that use VSDs over the Ethernet.

These are minor problems. In practice, we found the RVSD software easy to install, set up and manage.

A few other remarks that we record here relate to disk mirroring (which we have set up), not specifically to RVSDs. However, they may be useful reminders to anyone who sets up this kind of environment.

First, we added an additional disk into each of our volume groups. Our logical volume layout, as shown in Figure 35 on page 131, does not tell the whole story. We show it in this way for the sake of simplicity. In fact, as it stands, if we were to lose one of our 9333 disk towers, we would not continue to have access to our data. The AIX quorum for this volume group would be lost, and the volume group would no longer be available.

To counteract this problem, we put an additional disk into each of the volume groups. This disk is known as a *quorum buster* disk. If an entire disk tower fails, we still have more than 50% of the disks on line; we do not lose quorum. Alternatively, we could switch the quorum off for the logical volume, but there are limitations to this approach in the event of a disk failure. Recovery is more complex.

In our environment, recovery is very straight-forward. We run a **syncvg**, specify our logical volumes or disks, and AIX resynchronizes our copies after a disk failure. We perform the synchronization after we have replaced the failed disk. We are able to lose a disk, or an entire disk tower, with no impact on our performance or functionality. And, the loss is completely transparent.

Node takeover with RVSD takes a short while and is dependent on the actual configuration. In particular, the size of the volume group, or the number of disks, is an important factor. Since there are no filesystems to mount (only logical volumes), the need for a **fsck** is eliminated, speeding up the process considerably. In our environment, takeover takes approximately one minute.

The RVSD software does not start the Switch after a node comes back on line. But, the volume group is taken back by the primary node if it rejoins the SP set of VSD nodes after failure.

Manual procedures may be required to ensure that any necessary tasks are performed at recovery time. In particular, an **Estart** command may need to be run to start the Switch.

9.9 Backup/Restore

In the following, we describe different ways of doing backups, both of user and system data. The backup of user data using ADSM was presented in the redbook *ADSM on the SP*. However, ADSM does not currently back up logical volumes. Since the Oracle Parallel Server, one of the main applications running on the SP, uses raw devices as the base of the physical database, we spend most of this chapter showing how logical volumes can be backed up.

9.9.1 Attaching a 3490E to an SP Node

The device used to perform backups in our example is a fast tape drive, an IBM 3490E Model C11. The 3490E can be attached to an SP node through the High Performance I/O controller F/C 2420. The 2420 card is installed within the node. The tape drive comes with a device-driver microcode diskette. To install the microcode, copy the image of the diskette to a file (for example, /tmp/mc3490E) in the Control Workstation using *smit bffcreate*. Then, mount the file to the node where the 3490E is attached. The tape device driver is installed with SMIT screens in the node. Or, use sysctl from the Control Workstation, as follows:

```
sysctl -h sp2nodex exec /etc/installp -acd/tmp/mc3490E  Atape.driver
```

To configure the 3490E, enter **cfgmgr** followed by **smit tapes**. Perform the following:

1. Select **Add a Tape Drive**.
2. Select **A3490E SCSI SSD 3490E Tape Device**.
3. Select the appropriate SCSI card.

On the menu that appears, set:

- The *CONNECTION address* to MN, where M is the SCSI target address of the tape drive (in the back of the drive, normally set to 0) and N is the *SCSI LUN* address (0 for the left drive and 1 for the right drive when facing the unit). If there is only one drive, N is 0.
- The *Block Size* to an appropriate value (for example, 1024).
- The *Use Hardware Compression on Tape* to yes if data compression is desired. A compressed tape can hold 2GB of data.
- *Use DEVICE BUFFERS during write* to yes for performance.
- *Use Autoloading Feature of ACL* to yes if the tape stacker is to act as one large virtual tape.
- *Allow Device to Dominate SCSI Bus Cycles* to yes if the tape drive should be given priority on the SCSI bus.

In our study, we attached the 3490E to a server, node sp21n11, in group S. Other nodes access this tape drive remotely. Local use of the tape is trivial and is not presented here.

9.9.2 9076 SP Backup Strategy

We strongly recommend the separation of system and user data. When possible, a user volume group should be defined to hold user data. This allows you to define different schedules for the system and user data backups. For example, if the system data is more stable than the user data, back up the system data only when it has been changed.

9.9.3 Remote Backup/Restore with the dd Command

In the following, we show how to archive regular files to remote tape drives for occasional backup of data using tar piped to the rsh command. For example:

```
tar -cvf - file1.x file2.x | rsh sp21sw11 dd of=/dev/rmt0 obs=1024
```

This command archives the files file1.x and file2.x on a tape in a drive attached to node sp21n11 through the HPS interface. On the HPS, the mtu is 65520. This can be verified using the following command:

```
netstat -nI css0

Name  Mtu   Network     Address         Ipkts    Ierrs Opkts    Oerrs Coll
css0* 65520 <Link>                          1845     0     1835     9     0
css0* 65520 9.12.16.32  9.12.16.41          1845     0     1835     9     0
```

Be careful about such parameters as the block size of the remote tape drive and the mtu of the network. As shown in the example in Figure 39, the mtu should always be greater than the input block size because of the way the dd command works.

```
[root@sp21n04] / >netstat -nI en0
Name  Mtu   Network     Address         Ipkts    Ierrs Opkts    Oerrs Coll
en0   1500  <Link>                        773549     0   725272     0     0
en0   1500  9.12.30     9.12.30.54        773549     0   725272     0     0

[root@sp21n04] / >tar -cf - file1.out | rsh sp21n11 dd of=/dev/rmt0 obs=1024
[root@sp21n04] / >tar -cf - file1.out | rsh sp21n11 dd of=/dev/rmt0 bs=1024
48
dd: 23+2 records in.
dd: 6+0 records out.

4
dd: 12+0 records in.
dd: 12+0 records out.

[root@sp21n04] / >tar -cf - file1.out | rsh sp21n11 dd of=/dev/rmt0 bs=2048
8
dd: 0511-053 The write failed.
: A system call received a parameter that is not valid.
dd: 3+1 records in.
dd: 3+1 records out.
```

Figure 39. dd Command Failure When Block Size Greater Than MTU

Alternatively, the cat command could also be used:

```
[root@sp21n04] / >tar -cf - file.out | rsh sp21n11 "cat > /dev/rmt0"
```

Be aware that the dd command, in any form, runs very slowly over the network.

Note: The redbook *Oracle 7.13 Parallel Server and Query Option on 9076 SP2* presents a way to remotely back up a logical volume.

9.9.4 SYSBACK/6000

SYSBACK/6000 provides an easy-to-use way to back up and restore Oracle datafiles. You can use SMIT screens to do both local and remote backups of logical volumes.

9.9.4.1 SYSBACK/6000 Installation

In addition to the node to which the tape drive is attached, SYSBACK/6000 must be installed on all SP nodes where backups are to be performed. The image of SYSBACK/6000 should be copied to a file in the Control Workstation using `smit bffcreate` and mounted to all nodes targeted for installation. Using **sysctl** and **installp** from the Control Workstation, the installation can be done in parallel on all of the nodes, as follows:

```
sysctl -h sp2nodea sp2nodeb sp2nodec exec /etc/installp -acd/tmp/sysback.bin all
```

9.9.4.2 Configuring a Backup Server

The backup server must have at least one local tape drive, in our case, the 3490E. To configure a tape drive for use with SYSBACK, use the following SMIT screens:

```
  smit

  ╘═► AIX System Backup & Recovery/6000
      ╘═► Tape Drives
          ╘═► Configure a Defined Tape Drive
```

Select a tape drive from the list of defined tape drives. Figure 40 shows a tape drive configured and available for SYSBACK/6000.

```
                                  COMMAND STATUS

Command: OK            stdout: yes           stderr: no

Before command completion, additional instructions may appear below.

name status    location    description

rmt0 Available 00-06-00-00 IBM SSD 3490E tape device
```

Figure 40. Listing Tape Drives Known to Sysback

Next, configure this node as the backup server for the client′s SP nodes by traversing the following SMIT panels:

```
  smit

  ╘═► AIX System Backup & Recovery/6000
      ╘═► Remote Backup Services
          ╘═► Configure Remote Backup Services
```

Figure 41 on page 140 shows the resultant SMIT screen that allows the configuration of Remote Backup Services.

Perform the selection as indicated and press Enter to do remote configuration. Next, you must add a local tape access to remote client nodes. This is done through SMIT as shown in the following:

```
  smit

  ╘═► AIX System Backup & Recovery/6000
      ╘═► Remote Backup Services
          ╘═► Add Local Tape/Disk Access to Remote Hosts
```

```
                              Remote Backup Services

Move cursor to desired item and press Enter.

  Configure Remote Backup Services

    SERVER OPTIONS
  Add Local Tape/Disk Access to Remote Hosts
  List Remote Hosts with Local Tape/Disk Access
  Remove Local Tape/Disk Access from Remote Hosts

    CLIENT OPTIONS
  Define a Remote Backup Device for Use by this System
  List Defined Remote Backup Devices
  Undefine a Remote Backup Device
```

Figure 41. Configuring Remote Backup Services

Figure 42 shows sp21sw03 is defined to have access to the local tape drive on sp21sw11. Note that the HPS interface name is used, allowing backups to use the High Performance Switch. Repeat this step for each client node that is to use a remote tape server for backups.

```
                    Add Local Tape/Disk Access to Remote Hosts
Type or select values in entry fields.
Press Enter AFTER making all desired changes.
                                                       [Entry Fields]
* HOST NAME of Client System                        [sp21sw03]
  ALLOW this host to write to local disk?            yes
```

Figure 42. Adding Local Tape Access to Remote Hosts

9.9.4.3 Configuring a Backup Client

All client nodes can be configured in parallel using the following **sysctl** command:

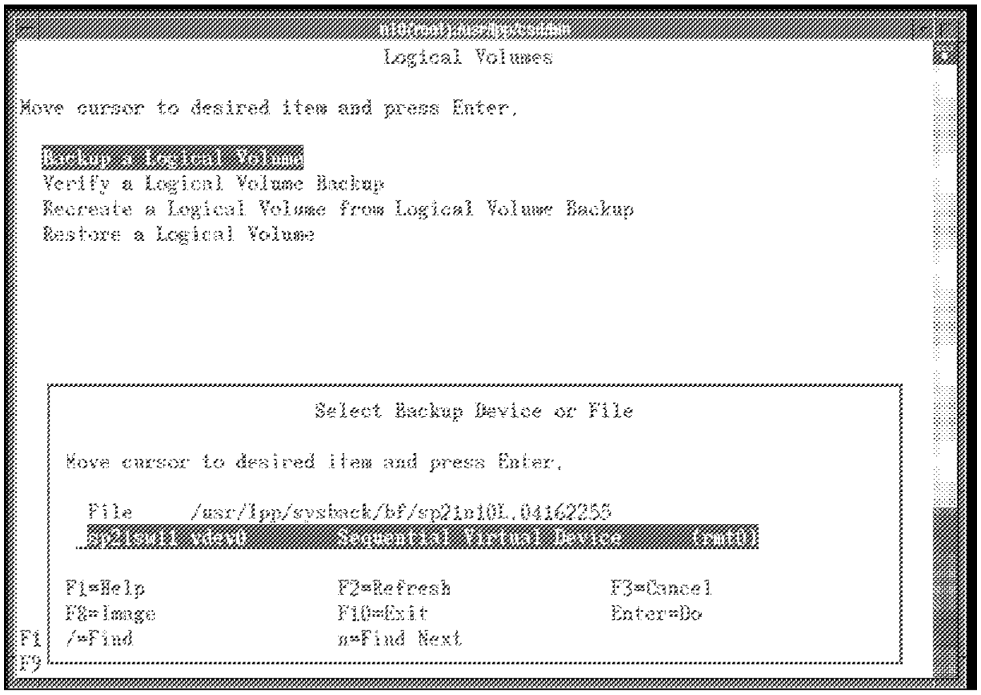
```
sysctl -h sp21n07 sp21n08 sp21n09 sp21n10 exec /usr/bin/cfgremtape -a
-h′sp21s11′ -f′vdev0′
```

9.9.4.4 Backing Up a Logical Volume

A client node can remotely back up a logical volume where a vsd is defined using SYSBACK/6000. To set this up, access the following SMIT panels:

```
smit

└→ AIX System Backup & Recovery/6000
    └→ Backup/List/Restore Options
        └→ Logical Volumes
            └→ Backup a Logical Volume
```

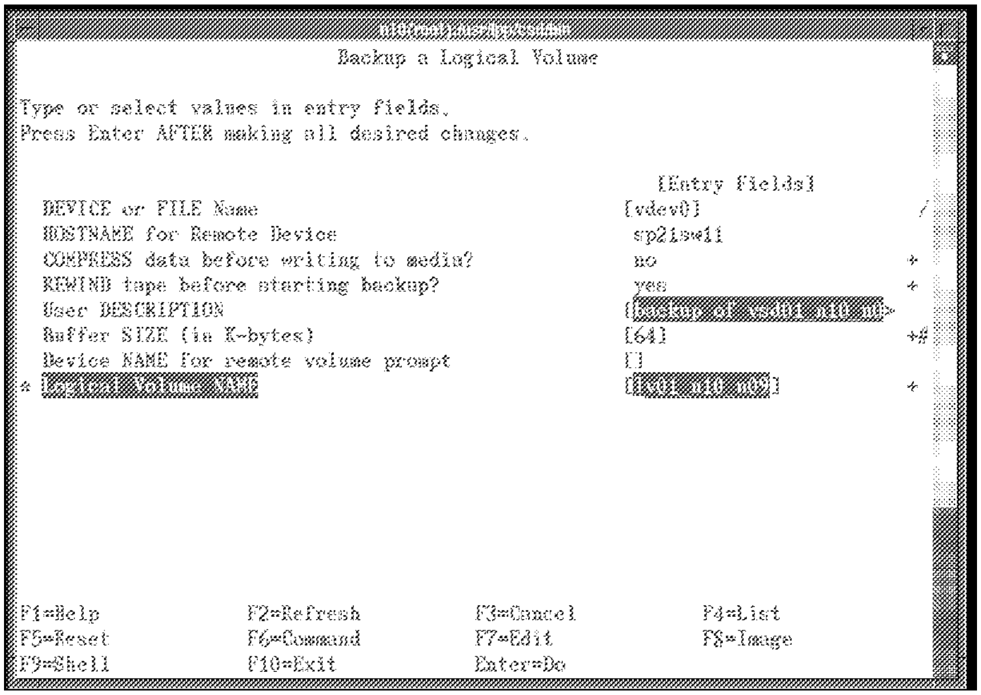
Figure 43 on page 141 shows how to select a remote tape device on the server sp21sw11 for backing up a logical volume.



```
                          Logical Volumes

Move cursor to desired item and press Enter,

  Backup a Logical Volume
  Verify a Logical Volume Backup
  Recreate a Logical Volume from Logical Volume Backup
  Restore a Logical Volume

                     Select Backup Device or File

    Move cursor to desired item and press Enter,

      File     /usr/lpp/sysback/bf/sp21n10L.04162255
      sp21sw11 vdev0        Sequential Virtual Device      (rmt0)

    F1=Help                F2=Refresh              F3=Cancel
    F8=Image               F10=Exit                Enter=Do
F1  /=Find                 n=Find Next
F9
```

Figure 43. Select a Virtual Tape Device to Back Up a Logical Volume

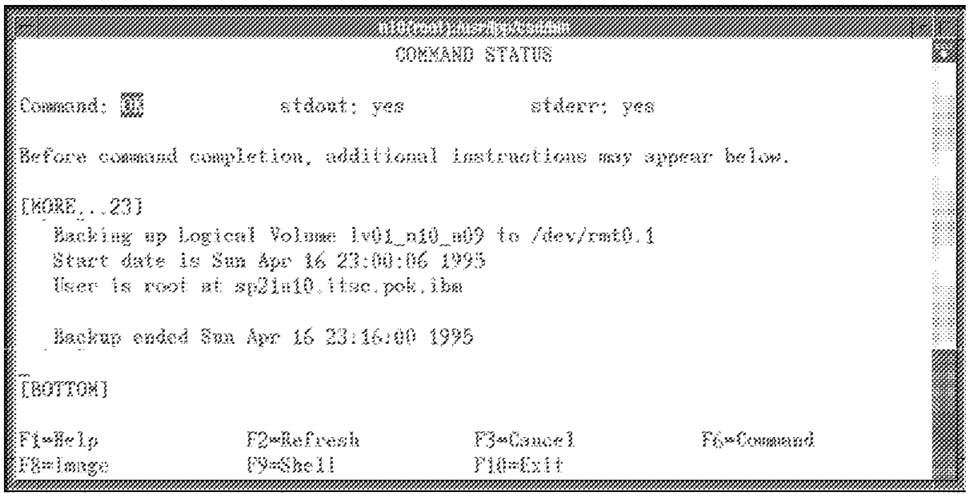
Figure 44 shows the SMIT screen used to back up logical volume lv01_n10_n09.



```
                        Backup a Logical Volume

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

                                                    [Entry Fields]
  DEVICE or FILE Name                              [vdev0]               /
  HOSTNAME for Remote Device                        sp21sw11
  COMPRESS data before writing to media?            no                   +
  REWIND tape before starting backup?               yes                  +
  User DESCRIPTION                                 [backup of vsd01 n10 n0>
  Buffer SIZE (in K-bytes)                         [64]                  +#
  Device NAME for remote volume prompt             []
* Logical Volume NAME                              [lv01_n10_n09]        +

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

Figure 44. Backing Up a Logical Volume with SYSBACK/6000

When the backup is complete, the backup time is indicated, as shown in Figure 45 on page 142.



```
                         COMMAND STATUS

Command: OK            stdout: yes           stderr: yes

Before command completion, additional instructions may appear below.

[MORE...23]
   Backing up Logical Volume lv01_n10_n09 to /dev/rmt0.1
   Start date is Sun Apr 16 23:00:06 1995
   User is root at sp21n10.itsc.pok.ibm

   Backup ended Sun Apr 16 23:16:00 1995

[BOTTOM]

F1=Help             F2=Refresh          F3=Cancel           F6=Command
F8=Image            F9=Shell            F10=Exit
```

Figure 45. Successful Completion of a Logical Volume Backup with SYSBACK/6000

Note: After the datafile is backed up, the tape is not rewound. This is to allow *stacked backup* of many logical volumes on the same tape.

9.9.4.5 Restoring a Logical Volume

Restoring a logical volume is also done remotely and consists of the two following steps:

- Re-create the backed-up logical volume

  This step only creates the logical volume. No data is loaded. Access the following SMIT panels:

  ```
  smit

  └→ AIX System Backup & Recovery/6000
      └→ Backup/List/Restore Options
          └→ Logical Volumes
              └→ Recreate a Logical Volume from a Logical Volume Backup
  ```

  Note: The name of the logical volume you create must be the same as that of the one backed up on the media. Inconsistencies between the two logical characteristics result in messages being displayed; appropriate actions should be taken to correct these inconsistencies. Also, you must enter the correct image number of the logical volume on the tape if it contains stacked backups.

- Restore the backed-up logical volume

  This step actually loads the data on the backup media to the logical volume re-created in the previous step. Through SMIT, access the following:

  ```
  smit

  └→ AIX System Backup & Recovery/6000
      └→ Backup/List/Restore Options
          └→ Logical Volumes
              └→ Restore a Logical Volume
  ```

  Note: The system searches for the correct image of the logical volume on the tape and loads the data.

Chapter 10. Job Management (Batch and Interactive)

This chapter describes tools and programs that can help system administrators to implement job management. In particular, we discuss the Resource Manager and Interactive Session Support. For information on another key resource-control tool, LoadLeveler, refer to the Redbook *IBM LoadLeveler Technical Presentation Update* (GG24-4511, scheduled for publication in August, 1995).

10.1 The Resource Manager

The Resource Manager on the SP is responsible for allocating SP resources to parallel jobs. When parallel jobs are running in User-Space mode using the Message Passing Protocol instead of TCP/IP, dedicated use of the switch adapter is required. Only one User-Space job can run at a time over the switch. Other types of jobs may run currently. Also, with a finely-tuned parallel job, you may not want to allow any other user-related interference on the nodes running the parallel job. So, you want to block the parallel nodes from being able to log in. Resource Manager provides the tools and commands to enable these requirements. Refer to *IBM 9076 Scalable POWERparallel Systems SP2 Administration Guide* for detailed information on how to install and configure the Resource Manager.

10.1.1 Access Control of Resources

When a user submits a parallel job that uses one of the Parallel Program Application Interfaces, the application passes a message to the Resource Manager for dedicated use of the switch. The Resource Manager checks whether the switch adapter is in use. If the switch is free, the Resource Manager submits the job, denying subsequent requests for use of the switch until the original job completes.

10.1.2 Access Control of Users

When you want no interference by interactive users on parallel nodes, you can define access control in the Resource Manager configuration. In principle, you disable login and rlogin for users on parallel nodes, preventing the users from logging in. Control of user logins is initially set by the **spacs_cntrl** command, but thereafter is controlled by the Resource Manager. When a user submits a job, the Resource Manager changes the login status of the user from “blocked” to “allow.” This changes the login attributes in the /etc/security/user file to “true” and updates the **spacs-control** files in /var/spacs. During the execution of the parallel job, the user is able to login interactively. To prevent this, configure the /etc/hosts.equiv file such that parallel job users cannot login with **rsh** or **rlogin** and comment out the **telnet** entry in the /etc/inetd.conf file. Interactive use of the spacs_cntrl command in conjunction with the Resource Manager may cause loss of user login status and, therefore, is not recommended. Refer to the *IBM 9076 Scalable POWERparallel Systems SP2 Administration Guide* for more details.

10.2 SP Nameservers and Interactive Session Support

This section discusses some of the options for setting up network nameserving on the SP, allowing you to use the SP as a single system while helping you to distribute workload among the nodes. The following topics are covered:

- TCP/IP name resolution
- Advantages of Interactive Session Support
- Interactive Session Support options
- Setting up Interactive Session Support
- Practical experiences using Interactive Session Support

10.2.1 TCP/IP Name Resolution

When a host computer wants to communicate with another over a TCP/IP network, it must use a TCP/IP network address to do so. The TCP/IP address is a number consisting of four components, 9.180.234.40, for example. Part of the address defines the network address and part defines the host address. Which part is the network component and which the host component can be determined by the *class* of the TCP/IP address. There are three classes of address, A, B and C. For example, the address 9.180.234.40 is a class-A address. This is determined by the first of the four numbers, which in this case is 9. A class-A address, by default, uses the first number to identify the network address and the remaining three components to identify the host address within that network. So, in this case, the network address is 9 and the host address is 180.234.40. By using a network mask, you can modify the network/host component split.

Using the netmask for splitting a network in logical parts is called subnetting. It is a very commonly-used method. Subnet addressing allows an autonomous system made up of multiple networks to share the same Internet address. The subnetwork capability of TCP/IP also makes it possible to divide a single network into multiple logical networks (subnets). For example, an organization can have a single Internet network address that is known to users outside the organization, yet configure its network internally into departmental subnets. In either case, fewer Internet network addresses are required while local routing capabilities are enhanced. Refer to InfoExplorer for a more detailed explanation of subnetting.

The addressing structure of TCP/IP allows you to set up a network. As long as you use the correct TCP/IP address, a message destined for a particular host on the network can find its way to that host through various routing protocols run by TCP/IP. This is true even in the most complex of networks, as long as the network addressing and routing have been defined correctly.

So, as a user, you can simply communicate with another system by specifying a command (such as a remote login or **rlogin**) and a TCP/IP address.

In real life, it would be pretty painful if you had to communicate with other TCP/IP hosts on the network using these TCP/IP addresses. If you are unfamiliar with the workings of TCP/IP, these addresses may not be easy to recognize or remember. Most of us would prefer to identify another machine on the network using a name for that machine, which TCP/IP refers to as a *hostname*.

To use the simple hostname, a mechanism is required that automatically translates the hostname into the TCP/IP address that the TCP/IP network must use. This mechanism is called TCP/IP *name resolution* and can be performed in a number of ways.

10.2.1.1 Name Resolution Using /etc/hosts File

The simplest way to perform name resolution employs a table on a user′s system. This table matches the hostnames that he chooses to use with the TCP/IP addresses. The user′s system could be a workstation, PC, RISC System/6000, other UNIX system or any system that is running the TCP/IP network protocols and software. In the case of a RISC System/6000, the table is stored in a file in the /etc directory of the system and is named /etc/hosts. An example of a simple /etc/hosts file is shown in Figure 46 on page 146.

Those lines that begin with a hash mark (#) are comments only and are not interpreted. The entries in the file for the hosts themselves contain a TCP/IP address followed by the hostname. In addition, another shorter name or alias can be used, if required; this is listed after the hostname. In our example, some of the hostnames are followed by the full hostname containing the TCP/IP full domain name (for a definition of the term domain, see 10.2.1.4, "TCP/IP Domains" on page 148). This is followed, in turn, by the alias or shorter name that you would normally use.

If name resolution by /etc/hosts is used, each and every one of the users′ workstations must have and maintain such a file. In addition, the file must contain a complete list of the hosts with which the user wishes to communicate.

When considering a large network of systems, changes to the network, such as adding a new system into the network, result in the need to change the /etc/hosts file on every system on the network if all of the information is to remain correct. For any reasonable-sized network, this would be an extremely tedious, if not impossible, task; therefore, another way of performing name resolution is required.

However, for the SP there is also another way of managing hostnames and IP-addressed across the system. Adding the /etc/hosts file to a File Collection would be a perfect way of keeping the /etc/hosts the same. This would be a reasonable strategy for those sites that are new to using TCP/IP and do not yet have a name resolution strategy. Refer to 13.4, "SP File Collections" on page 206 for details on File Collections.

10.2.1.2 TCP/IP Nameserver

A better method of name resolution for a large network or, indeed, any network with more than a handful of systems, is to set up one machine as a server for all of this information. This server machine is called a *nameserver*.

Instead of using the /etc/hosts file on each system, each host or client on the network points to the nameserver. The nameserver is the only one to keep an up-to-date table of all of the names and addresses on the network. In this way, any changes to the network need only be reflected on the TCP/IP nameserver; maintenance becomes very straightforward. The /etc/hosts file contains very minimal information in this case and does not need to be updated when the network changes.

```
# COMPONENT_NAME: TCPIP hosts
#
# FUNCTIONS: loopback
#
# ORIGINS: 26  27
#
# (C) COPYRIGHT International Business Machines Corp. 1985, 1989
# All Rights Reserved
# Licensed Materials - Property of IBM
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
#  /etc/hosts
#
# This file contains the hostnames and their address for hosts in the
# network.  This file is used to resolve a hostname into an Internet
# address.
#
# At minimum, this file must contain the name and address for each
# device defined for TCP in your /etc/net file.  It may also contain
# entries for well-known (reserved) names such as timeserver
# and printserver as well as any other host name and address.
#
# The format of this file is:
# Internet Address Hostname # Comments
# Items are separated by any number of blanks and/or tabs.  A ′#′
# indicates the beginning of a comment; characters up to the end of the
# line are not interpreted by routines which search this file.  Blank
# lines are allowed.

# Internet Address Hostname           # Comments
# 192.9.200.1   net0sample            # Ethernet name/address
# 128.100.0.1   token0sample          # token ring name/address
# 10.2.0.2      x25sample             # x.25 name/address

127.0.0.1       loopback  localhost   # loopback (lo0) name/address

9.12.0.36       risc0036.itsc.pok.ibm.com   risc0036

9.180.234.40    olive
9.12.0.137      sp21cw0
9.12.30.61      sp21n11
9.12.30.63      sp21n13.itsc.pok.ibm.com   sp21n13
```

Figure 46. A Simple /etc/hosts File

TCP/IP nameserving is very common and is used for most TCP/IP networks. (Excellent documentation on TCP/IP nameserving can be found in the AIX InfoExplorer softcopy documentation for the RISC System/6000 and SP. Also refer to “DNS and BIND,” a publication of O′Reilly and Associates).

The nameserver does not use the standard /etc/hosts file to store information on hosts and their TCP/IP addresses. Rather, this information is stored in three configuration files.

1. **named.data**

   This file contains the complete list of host names and addresses in tabular format. The information is very similar to that stored in the `/etc/hosts` file, but it is formatted in a specific way. In addition, it contains extra information used by the nameserver processes.

   An example of a `named.data` file can be found in Figure 51 on page 157.

2. **named.rev**

   This file has a similar purpose to that of the `named.data` file, but is used for reverse name resolution, (meaning the ability to determine a hostname if a TCP/IP address is specified). This is required for a TCP/IP nameserver to be fully functional.

   The information in this file corresponds to that in the `named.data` file; that is, there is an entry in this file for every host on your network. The `named.rev` file also has specific configuration information related to how this nameserver operates.

   An example of a `named.rev` file can be found in Figure 52 on page 158.

3. **named.boot**

   This file defines a number of the ways in which this nameserver could operate.

   For example, there are a number of different types of nameservers. A *primary* nameserver resolves all of the TCP/IP requests for a given domain and contains all of the host and address information in its own configuration files.

   Alternatively, a *secondary* nameserver can be configured to get its information from another primary nameserver at predefined intervals. A secondary nameserver can run independently of a primary nameserver once its information has been downloaded.

   Using various types of nameservers, it is possible to set up a network with alternative nameservers that can be utilized in the event of a failure of one of the nameservers

   (A full discussion of nameservers is beyond the scope of this book. The reader who has a requirement for further information should refer to InfoExplorer).

   By default, the `named.boot` file that determines the type of a nameserver is located in the `/etc` directory. The `named.boot` file can be placed anywhere, but, if it is not in its default place, its location must be specified when the nameserver process is started.

   The `named.boot` file also specifies the full path name and filenames for the `named.data` and `named.rev` files. These can also be located anywhere.

   A nameserver should really be called a TCP/IP domain nameserver as it resolves TCP/IP names and addresses for a specific TCP/IP domain. However, a nameserver can be serve multiple domains. If this is the case, each of these will be listed in the `named.boot` file.

   Finally, a nameserver can operate as a different type of nameserver for different domains at the same time. For example, a single nameserver could operate as a primary nameserver for one domain and as a secondary nameserver for a different domain, concurrently. All of this would be specified in the `named.boot` file.

The exact format of these files must be carefully reproduced. A file must have spaces where specified and each and every period (full stop) in the file can be critical. A single missing period can cause the nameserver to fail!

Luckily, there are script files that can be used to help create the named.data and the named.rev files. These scripts take input from the standard /etc/hosts file and output the named.data and named.rev files, which can then be edited as necessary. The scripts can be found in the /usr/lpp/tcpip/samples directory. The **hosts.awk** script creates the named.data file and the **addrs.awk** script creates the named.rev file. Sample files for named.data, named.rev and named.boot can also be found in this directory, along with full instructions on how to run the awk script files.

To start a nameserver that is already configured, you must start the **named** (name daemon) process. It can be started from the command line by issuing the command:

```
startsrc -s named
```

Or, use the SMIT panel to start subsystems.

By issuing this command without flags, the default /etc/named.boot configuration file is used to define how this nameserver will run.

10.2.1.3 Client Systems

Each client system that wishes to use a nameserver must specifically point to that nameserver. In the case of a RISC System/6000 or other UNIX system, this information is stored in a file called /etc/resolv.conf. Other workstations, such as PCs, store this information in a similar manner.

In addition, there could be multiple nameservers on the network that are each responsible for different segments or domains within the network. If this is the case, the /etc/resolv.conf file can have multiple entries.

10.2.1.4 TCP/IP Domains

It is possible to group together a number of hosts in a TCP/IP network into a collection of systems that is known as a TCP/IP domain. The name for such a domain should follow standard TCP/IP naming conventions and should indicate the department or organization that uses this domain. For example, here in Poughkeepsie, the domain name for the systems on the network is itsc.pok.ibm.com. This domain name indicates that the department (International Technical Support Center) is located in Poughkeepsie (New York), that the department is part of IBM, which, in turn, is a commercial organization.

A single nameserver can handle name resolution for a number of TCP/IP domains or, alternatively, multiple nameservers could be used for resolving the names and addresses for separate domains.

10.2.1.5 Nameserving with NIS

It is also possible to use Network Information System (NIS) for resolving TCP/IP hostnames in a fashion similar to TCP/IP domain nameserving. This is described further in 13.5, “Network Information Service” on page 223.

10.2.2 Advantages of Interactive Session Support (ISS)

In the case of the SP, there is an additional nameserving requirement. The objective in all aspects of user management on the SP is to make the fact that the SP has multiple nodes completely transparent to the users. That being the case, when a user communicates with the SP, you may wish to use a single hostname that is a generic name for a number of SP nodes.

We have partitioned our SP according to application requirements. As you will recall, Group D, for example, actually contains four nodes, nodes 7, 8, 9 and 11. As a user, you may wish to log into the Group D set of nodes without actually being aware that more than one node is involved. You, therefore, wish to communicate with a hostname called “Group D” that actually consists of four TCP/IP hostnames: sp21n07, sp21n08, sp21n09 and sp21n11.

This kind of function can be provided by the IBM utility Interactive Session Support (ISS), a component of the IBM LoadLeveler product.

In fact, ISS provides additional function other than the grouping of TCP/IP hostnames into a generic group name. ISS is capable of balancing the logins and workload between each of these nodes or systems in a way that you can define. This can be very useful in helping to dynamically make the most effective use of the available hardware and nodes within the SP system, while, at the same time, making the system appear very simple to the end users.

ISS requires the use of a TCP/IP nameserver in order for it to operate.

ISS, once started, runs processes that query the groups of nodes at defined intervals to establish their status. The configuration file for ISS specifies commands that will be run on each of the nodes to determine which node is the most suitable candidate for login sessions. The criteria used are user-defined and can be very sophisticated, if required.

The ISS server requires **rsh** access to each of the nodes in the group to run these commands. Kerberos authority must be in place for ISS to issue **rsh** commands from the ISS nameserver to each of the nodes in the groups of nodes.

10.2.3 Interactive Session Support Options

Setting up ISS is similar to setting up a normal TCP/IP nameserver, but there are some minor differences. In addition, there are a number of choices open to you in this environment. These choices are related to the type of TCP/IP nameserver you already have in place and whether or not you wish to use that server for managing ISS. Different users have different integration requirements.

You can choose to run an SP node as a nameserver for either basic or ISS nameserving. Equally, you could select an RS/6000 or other system external to the SP frame for this task.

There are three questions that you must answer when setting up an ISS environment:

1. Where will the nameserver management software run?
2. Where will the ISS software run?
3. Where will the clients′ “first point of call” for name resolution be?

There are essentially four cases that should be considered when answering these questions. Each of the four cases has a corresponding approach as to how you configure ISS.

1. Utilize an existing nameserver if one exists, and add SP nodes to the existing domain.
2. Utilize an existing nameserver if one exists, but add a new subdomain for the SP nodes.
3. Add a new nameserver of your own, pointing it to an existing nameserver if one exists.
4. Add a new nameserver, but allow clients to initially go to an existing nameserver for name resolution.

Some of these approaches may appear very similar, but there are subtle differences in how they work. We will discuss each of the approaches in turn, then look in detail at how to implement one of them. We will implement option 3, as this exercises all of the skills needed to implement any of the other approaches.

10.2.3.1 Use Existing Nameserver and Existing Domain

When reviewing each of the options for setting up an ISS nameserver, let us consider the case where you wish to remotely log into the Group D set of nodes. The name of the group is defined as `groupd_lite.sp2`. The idea is that when you log into this generic hostname, you end up on the node with the lightest load within the Group D set of nodes. As a user at a workstation, you would issue the command **rlogin groupd_lite.sp2**. The name `groupd_lite.sp2` must be resolved. We will look at how this name resolution would take place in each of the four cases.

In the first option, an existing nameserver is already in place. You wish to run ISS on this nameserver to support your groups of SP nodes. This approach requires changes to the existing nameserver.

Figure 47 on page 151 shows this scenario.

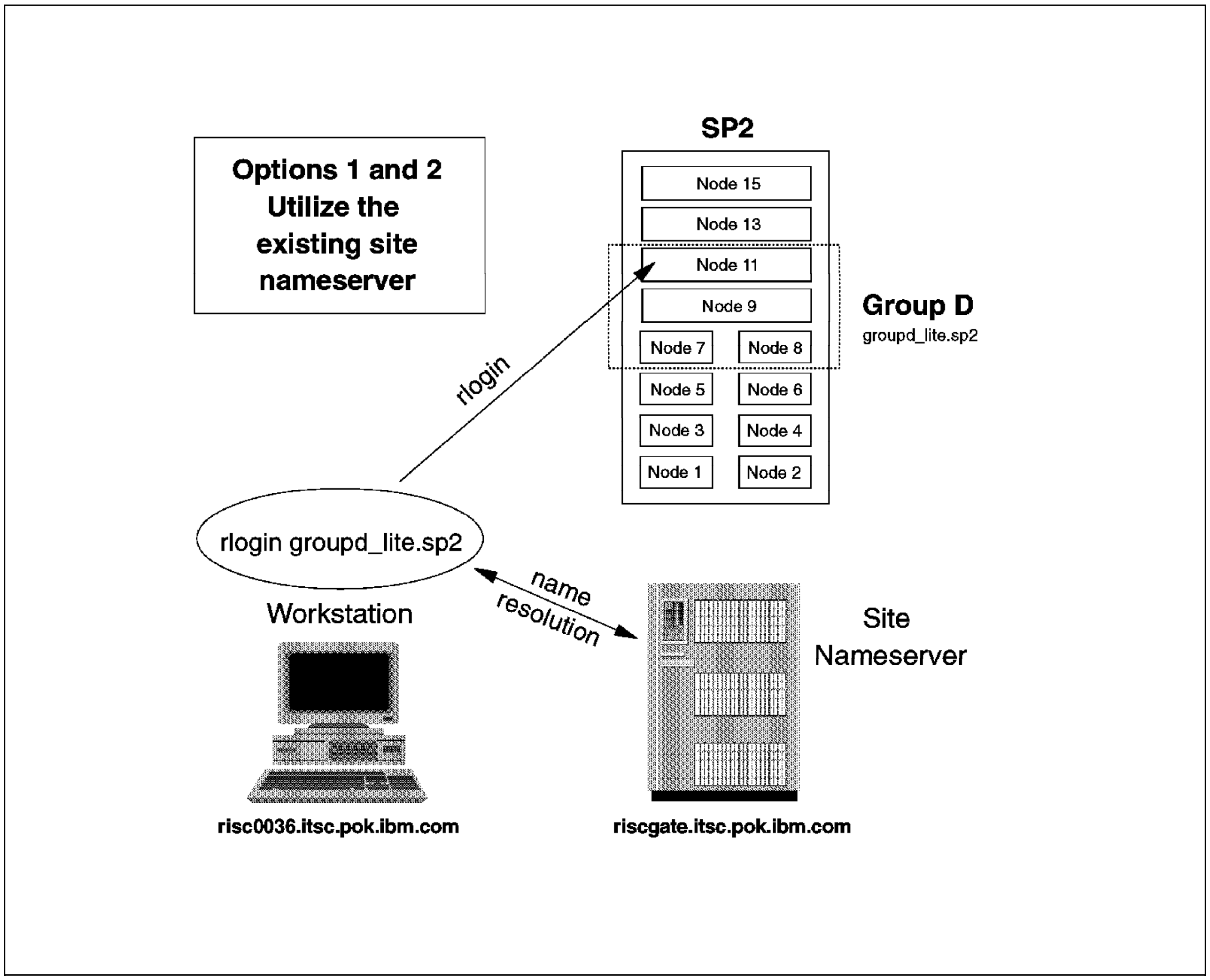


Figure 47. Options 1 and 2 - Utilize Existing Nameserver

In this scenario, an existing nameserver performs name resolution for all of the hosts on the network. Hostnames and addresses for each of the SP nodes must be added. This is business as usual so far, allowing TCP/IP access to each of the SP nodes individually

However, since you wish to run ISS, you must make some minor changes to this nameserver.

First, you must add definitions for your SP node groups into the `named.data` file. In the diagram, you will notice that the name of the group is `groupd_lite.sp2`. The name is meant to indicate that the group of nodes is the Group D set of nodes, that the selection process will pick the node with the lightest load at that time and that the group is on the SP. You could select any name you like here. (In our case, we have a number of logical node groups. We have named them in a way that should help us to identify the group). In real life, of course, a user can be shielded from any such complex-looking name; the user can start an application that communicates with his or her group of nodes transparently.

The `named.data` file is standard in most respects, but contains the additional entries for the SP groups.

In this scenario (using the existing nameserver), you would expect the clients to be pointing to this nameserver already. In a large network environment, you might potentially see hundreds of PCs, workstations and other clients. One of the real benefits of using this option when setting up ISS is that you do not have to change anything on any client. The only machine that you must change is the nameserver.

In this scenario, all name-resolution requests are routed directly to the existing or site nameserver. (This nameserver also runs the ISS-specific software). A response is returned to the client identifying one of the allowable SP nodes within the Group D collection of nodes. The remote login targets that node directly.

10.2.3.2 Use Existing Nameserver with New SP Subdomain

With this option, you decide to use an existing nameserver once again, but to define a TCP/IP subdomain for the SP nodes that will be serviced by ISS.

The process is very similar to the last case, but you subdivide your domain and define a specific SP subdomain for your collection of nodes.

Why would you do this? Well, at frequent intervals, ISS updates the information that the nameserver maintains. It, in fact, predefines the answer to the “lightest node” question and stores this response until it is replaced after the next update or polling period.

Each time this occurs, the nameserver reloads all of its information on behalf of ISS. For a large network, this could result in a significant overhead on the nameserver.

To prevent this problem from occurring, you can define a subdomain for just the SP nodes that are polled in your groups. Should a reload occur now, it would only affect the information in this subdomain, not likely to require much overhead.

For a large network, the additional step of defining a new SP subdomain would certainly be worthwhile. Once again, you do not need to change the software or configuration of any machine other than that of the existing nameserver. The clients continue to point to the existing nameserver and the SP nodes are set up as before. You do not, for example, have to configure the SP nodes in your groups differently. The SP subdomain is a logical domain that exists on the nameserver only for the purposes of ISS.

Figure 47 on page 151 again shows how this would operate.

10.2.3.3 Implement New Nameserver with New SP Subdomain

The next option involves adding an extra nameserver purely to look after the ISS nameserving. No changes to the existing nameserver are required if one exists.

Figure 48 on page 153 shows how this would operate.

One of the nodes is chosen to operate as the ISS nameserver. This node runs the usual nameserver process (named process) plus the ISS software. The diagram shows an example in which node 5 is the ISS nameserver.

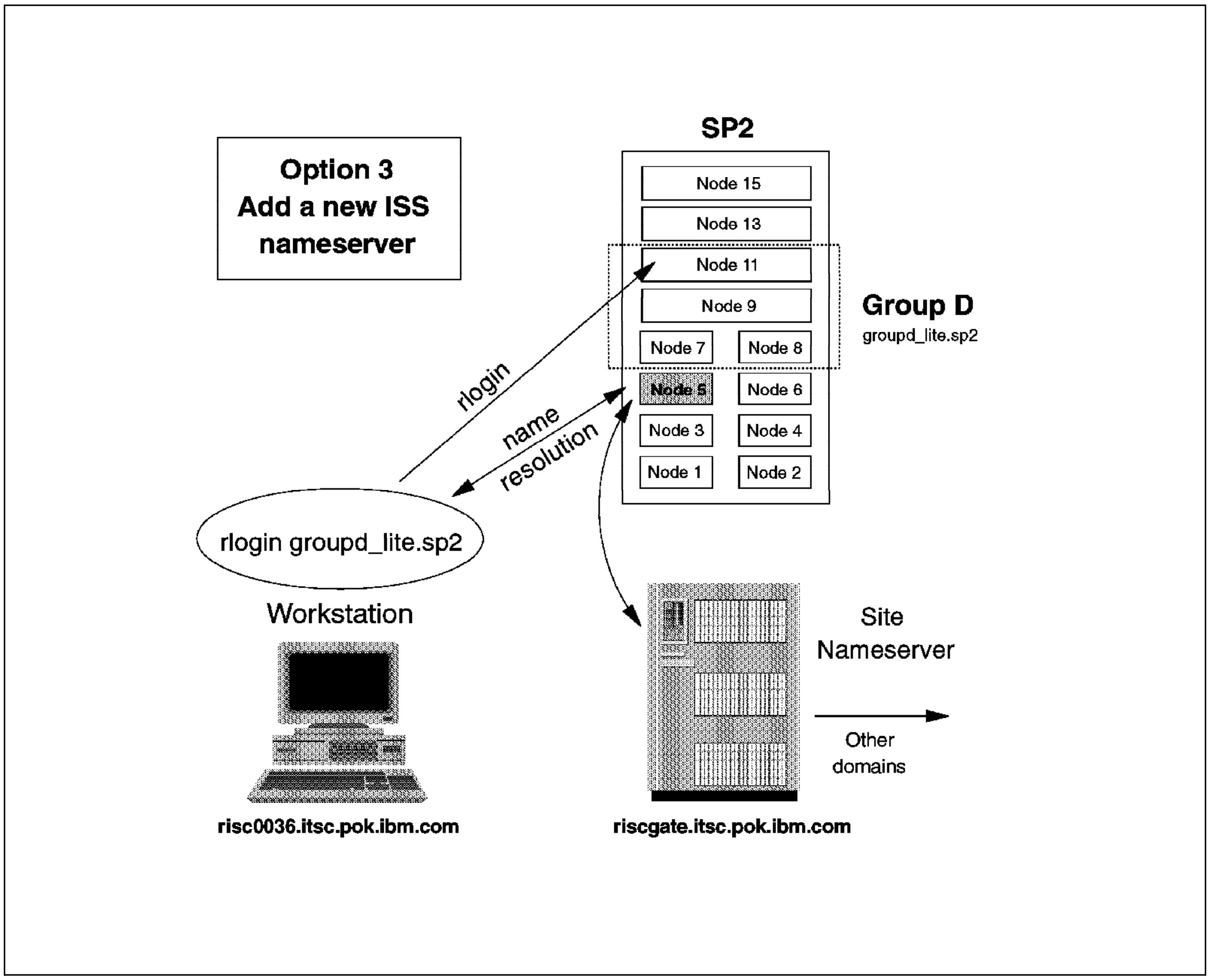


Figure 48. Option 3 - Utilize Existing Nameserver with Subdomain

The ISS nameserver can be within the group of SP nodes if required. Alternatively, the ISS nameserver can be running on a separate SP node or even, as is shown in this case, an external RISC System/6000 or UNIX system.

The installation and setup of this option are described in 10.2.4, "Setting Up Interactive Session Support" on page 155.

The initial request for name resolution is passed to the ISS nameserver. If an existing nameserver is already in place, it need not be changed at all. But, the clients' /etc/resolv.conf files must be altered to reflect the fact that each is using the new ISS nameserver rather than the existing site nameserver.

It would be an administrative headache if you had to maintain correct data and information on both the original site nameserver and the new ISS nameserver. Rather than do that, it would be better once again to define a new SP subdomain. You can specify that name resolution of any hosts in this SP subdomain occur on the ISS nameserver, whereas resolution of any other host in the original site domain be automatically passed to the site nameserver for it to handle. This, then, only requires you to keep the SP subdomain information up to date on the ISS nameserver. There will be no duplication of information across the two nameservers.

10.2.3.4 Implement New Nameserver with Clients Using Existing Nameserver

With the last option, we again add a new nameserver, but clients initially attempt to resolve their names from an existing nameserver. This nameserver passes on the unresolved queries related to the SP ISS nodes to the ISS nameserver.

Figure 49 shows how this could be achieved.

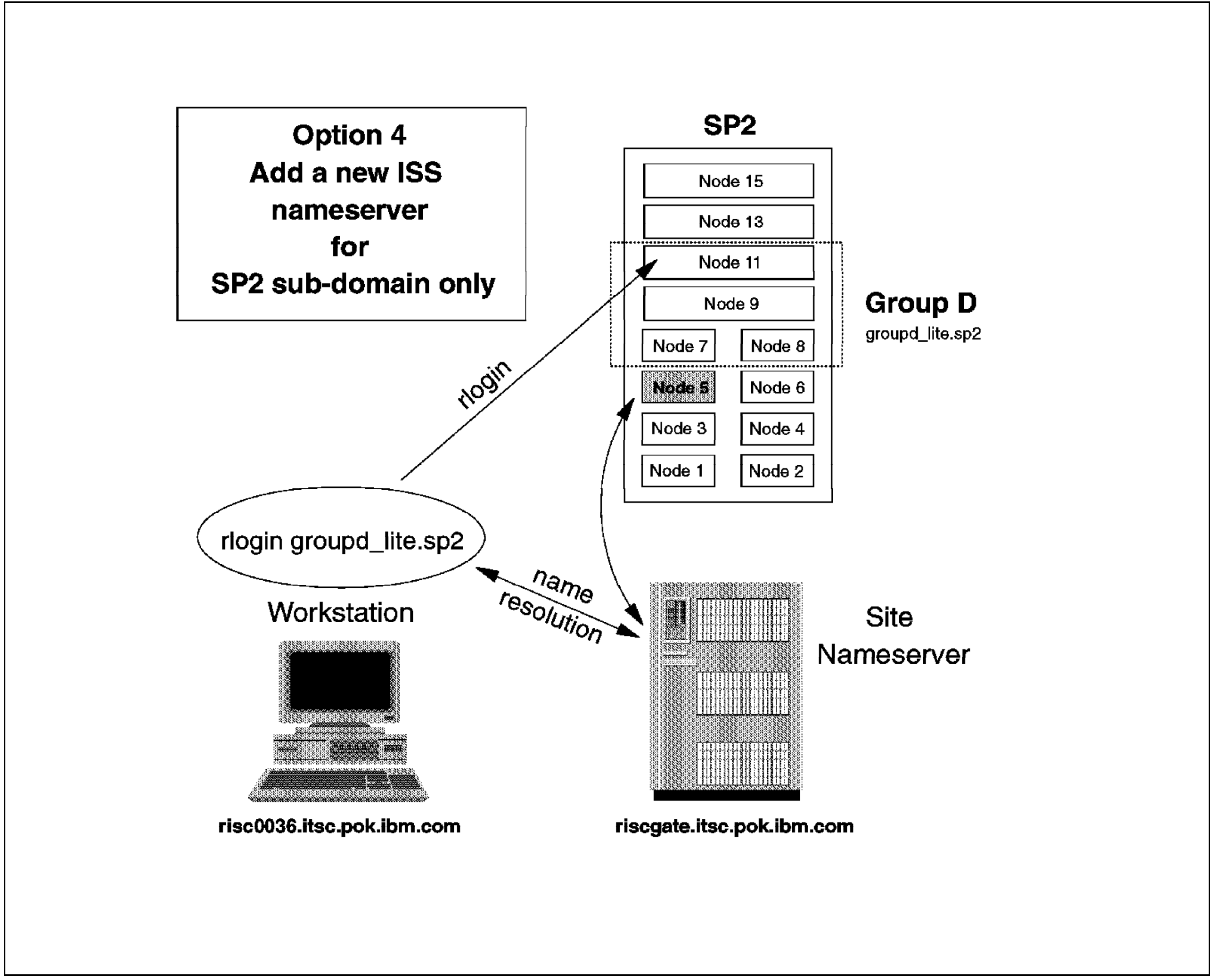


Figure 49. Option 4 - Utilize New SP Nameserver with Subdomain

The existing site nameserver must be modified to understand the existence of this new SP subdomain. It can then pass on the resolution for these nodes to the ISS nameserver (shown running on node 5 of the SP).

As with all cases, you do not duplicate information on the two nameservers. Each of them is responsible for name resolution for its own domain or subdomain, and maintenance of the nameservers is minimized when network changes are made.

10.2.4 Setting Up Interactive Session Support

Let us now go through the steps required to set up Option 3 for ISS, as shown in Figure 48 on page 153.

The steps in outline are as follows:

1. Install the required software
2. Edit the configuration files
3. Start the nameserver and ISS software processes

We are assuming that there is an existing nameserver operational for this site. In our case, `riscgate` is the hostname of the nameserver for the site domain, `itsc.pok.ibm.com`. So, the full domain name of this nameserver is `riscgate.itsc.pok.ibm.com`. We will not be making any changes to this nameserver, but will use it for resolving all hostnames outside of the SP nodes in our ISS pool.

The SP nodes are already set up and installed. The nodes in Group D are nodes 7, 8, 9 and 11 with hostnames of sp21n07, sp21n08, sp21n09 and sp21n11.

The full domain names for these nodes are:

- `sp21n07.itsc.pok.ibm.com`
- `sp21n08.itsc.pok.ibm.com`
- `sp21n09.itsc.pok.ibm.com`
- `sp21n11.itsc.pok.ibm.com`

These nodes need no changes either.

We must, however, install software on node 5 (our future ISS nameserver) and configure it.

10.2.4.1 Installing the Required Software

We will now install the required software on node 5.

The *named* daemon software is part of standard TCP/IP and should already be installed. We can, however, edit the `/etc/rc.tcpip` file to specify that the *named* process be automatically started whenever we start TCP/IP. The line in the file that starts the *named* process is commented out. Remove the hash mark (#) to make this operational.

The *named* process will now start when next we initialize TCP/IP. We could also start it from the command line, but will not do so until we have edited the configuration files.

The ISS software is part of the LoadLeveler program product. If you just intend to use ISS, you do not need to install or use any of the other parts of LoadLeveler.

In our case, the LoadLeveler install image is stored in the `/usr/sys/inst.images/ssp` directory on the SP Control Workstation. Using NFS, we can mount that directory onto node 5 with the command:

```
mount sp21cw0:/usr/sys/inst.images/ssp /mnt
```

This command should be run on the command line of node 5.

Prior to installing LoadLeveler, including ISS, we must define a user with the name of loadl; this user should be in the loadl group. The install process is carried out by the root user.

We can now install the ISS software using the standard SMIT panels. We select the LoadL.iss.obj 01.02.00.01. portion of LoadLeveler.

All required software should now be installed.

10.2.4.2 Edit the Configuration Files

We must customize the nameserver files and the ISS configuration file.

Nameserver Configuration: As you already know, there are three files that should be configured for node 5 to operate as our ISS namesever. We will configure and test this before we start ISS.

The named.boot file specifies how this nameserver will operate. Figure 50 shows our named.boot file.

```
domain          sp2.itsc.pok.ibm.com
primary         sp2.itsc.pok.ibm.com          /usr/lpp/LoadL/iss/named.data
primary         in-addr.arpa                  /usr/lpp/LoadL/iss/named.rev
```

Figure 50. Our named.boot File

As you can see from the figure, we are defining a subdomain here called sp2.itsc.pok.ibm.com. As shown in this file, the ISS nameserver will resolve TCP/IP network information for this subdomain only; it will be a primary nameserver and the two important data files that the nameserver software needs, named.data and named.rev, are located in the /usr/lpp/LoadL/iss directory.

The sample file found in the /usr/lpp/LoadL/iss directory can be copied and used as a starting point for creating the named.boot file.

When we have finished editing the named.boot file, we can copy or move it to the /etc directory, where the nameserver processes expect it to be.

The next file that we should edit is the named.data file. Again, the sample one in the /usr/lpp/LoadL/iss directory can be copied and used as a starting point. (If you do this, make sure that you keep a copy of the original sample for reference purposes).

Figure 51 on page 157 shows our named.data file.

On the first line, we define the full domain name of our ISS nameserver (that is, node 5). Note that we do not configure node 5 with this domain name; this is a logical subdomain name only. The actual full domain name for node 5, as configured in its TCP/IP definitions, is sp21n05.itsc.pok.ibm.com, just like all of the other nodes.

Due to the limited width of this page, the whole of the first line is not shown. We use a continuation symbol (), which does not actually exist in the file; rather, the first two lines in our example appear as one long line in the file.

Most of the rest of this file is standard, as for any TCP/IP nameserver. There are, however, a few minor differences.

```
@               9999999 IN      SOA     sp21n05.sp2.itsc.pok.ibm.com. \
                                        root.sp21n05.sp2.itsc.pok.ibm.co

                                        80719           ; Serial
                                        3600            ; Refresh
                                        300             ; Retry
                                        3600000         ; Expire
                                        86400 )         ; Minimum
                9999999 IN      NS      sp21n05
loopback        9999999 IN      A       127.0.0.1       ; loopback (lo0) name/address
localhost       9999999 IN      CNAME   loopback

itsc.pok.ibm.com.               9999999       IN      NS      riscgate.itsc.pok.ibm.com.
riscgate.itsc.pok.ibm.com.      9999999       IN      A       9.12.0.32

groupd_lite     9999999 IN      A       9.12.30.58
groupd_next     9999999 IN      A       9.12.30.58
groups          9999999 IN      A       9.12.30.61
groupt          9999999 IN      A       9.12.30.56
```

Figure 51. Our named.data File

You will notice that we have defined another nameserver for the `itsc.pok.ibm.com` domain. This is the existing `riscgate` nameserver. We point to that nameserver so that it can resolve addresses outside of our logical SP subdomain.

Finally, you will see the names of the groups of nodes that we have chosen to use with ISS:

groupd_lite.sp2	This is the database group of nodes (Group D) on our SP, composed of nodes 7, 8, 9 and 11. The node that is selected will be the one with the lightest load.
groupd_next.sp2	The same group of nodes, but, this time, each login will be directed to the next node in sequence, cycling around the four nodes.
groups.sp2	This is the group of SP server nodes as defined on this system, nodes 3, 4, 5 and 13, in our case. The node with the lightest load will again be selected.
groupt.sp2	This group is the development and test group, composed of nodes 1, 2, 6 and 15. The node with the lightest load will be selected.

At this stage, the address listed for each group is the address of any of the hosts in that group. It will be replaced with the appropriate address when we start ISS. ISS will update this file at predefined intervals.

The last file that we must edit is the `named.rev` file.

Figure 52 on page 158 shows our `named.rev` file.

```
@               9999999 IN      SOA     sp21n05.sp2.itsc.pok.ibm.com. \
                                        root.sp21n05.sp2.itsc.pok.ibm.co

                                        1.1             ; Serial
                                        3600            ; Refresh
                                        300             ; Retry
                                        3600000         ; Expire
                                        86400 )         ; Minimum
                9999999 IN NS  sp21n05
1.0.0.127               IN PTR loopback.itsc.pok.ibm.com.
12.9            9999999 IN NS  riscgate.itsc.pok.ibm.com.
```

Figure 52. Our named.rev File

The information in this file is similar to that in the `named.data` file, but this information is used for reverse resolution (determining the hostname if the address is known).

Once again, the first line was too long for the page and has been continued here. This is not the case in the real file.

Figure 53 shows the `/etc/resolv.conf` file that we will use on this nameserver. This points to our own nameserver on node 5.

```
nameserver 9.12.30.55
```

Figure 53. Our /etc/resolv.conf File on the ISS Nameserver

We are now ready to start the nameserver software. We can do so by issuing the command **startsrc -s named** from the command line or using the SMIT panel.

Alternatively, while we are testing this, it can be useful to run the following command from the command line:

```
startsrc -s named -a "-d"
```

This runs the nameserver in debug mode. It writes messages about the status of the nameserver to a file in `/var/tmp`.

We can check whether our nameserver is working properly by issuing a **nslookup** command. For example, to check that the nameserver can resolve the hostname `sp21n11`, we can type **nslookup sp21n11**. The TCP/IP address should be displayed. We should check that the SP nodes (used by ISS and in the SP subdomain), along with other hostnames, are all resolved. In other words, we need to check not only that our nameserver is working properly, but also that requests for other domains are passed on to the site nameserver.

To fully test our nameserver, we must test name resolution from a client. In our case, our client is a RISC System/6000 in the same domain with a hostname of `risc0036`.

The client should point to the new ISS nameserver with its `/etc/resolv.conf` file. Figure 54 on page 159 shows the `/etc/resolv.conf` file that we will use on this client. This points to the ISS nameserver on node 5.

```
domain     itsc.pok.ibm.com
nameserver 9.12.30.55
```

Figure 54. Our /etc/resolv.conf File on the Client Workstation

If this all works, we are ready to configure ISS fully. Do not move on to ISS until nameserving is working properly.

ISS Configuration File: There is only one file that we must configure to use ISS. This file describes our policies for selecting nodes within a group at login time. This ISS configuration file can be called anything and is identified at startup time for ISS. In our case, we have left the file in the same directory as the sample files (that is, the `/usr/lpp/LoadL/iss` directory). Our configuration file is called `Iss_config`.

Figure 55 on page 160 shows our `Iss_config` file.

An entry exists for each of the groups that we listed in `named.data`.

For example, the entry for `groupd_lite` specifies the NAME for the group and the polling period, five seconds. It also specifies the way in which we determine the selected node for the next login. In our case, we run a **vmstat** command remotely on each node in the group, cutting out the fourteenth column (that is, the value for CPU User activity). This METRIC determines a value for each of our nodes. The entry also specifies the TCP/IP addresses of all four nodes. Finally, we specify that of all of the nodes, the one with the MIN value as measured above will be selected.

We have similar entries for all of our groups. You will notice that the `groupd_next.sp2` group uses the ROUNDROBIN metric, telling ISS to cycle around the nodes in that group.

ISS Options: In our case, we have only installed ISS and not the full LoadLeveler product. LoadLeveler can be used for managing batch jobs in a very powerful manner.

If we had been using LoadLeveler, we would not have to define parameters that had already been set up for LoadLeveler when setting up ISS. We can specify that we wish to use this predefined LoadLeveler information in the ISS configuration file.

There are a number of keywords that can be included in the ISS configuration file:

NAME This is the name of the group of nodes that you wish to treat as one entity. In our case, our pools of nodes are:

- groupd_lite.sp2
- groupd_next.sp2
- groups.sp2
- groupt.sp2

```
#
# Iss_config
#
# Sample configuration file for Interactive Session Support
#

# Uncomment the next line to turn the ISS trace facility on
#ISS_TRACE

# Uncomment the next line to provide a non-default named bootfile
# inserting the fully pathed filename of your bootfile in place
# of <filename>
#NAMED_BOOTFILE <filename>

# Group D lightest loaded node entry
#
NAME            groupd_lite
POLL            5
METRIC          CUSTOM 'vmstat 1 2 | tail -1' | awk '{print $14}'
SERVERS         9.12.30.57 9.12.30.58 9.12.30.59 9.12.30.60
POLICY          MIN

NAME            groupd_next
POLL            5
METRIC          ROUNDROBIN
SERVERS         9.12.30.57 9.12.30.58 9.12.30.59 9.12.30.60

NAME            groupt
POLL            5
METRIC          CUSTOM 'vmstat 1 2 | tail -1' | awk '{print $14}'
SERVERS         9.12.30.51 9.12.30.52 9.12.30.56 9.12.30.62
POLICY          MIN

NAME            groups
POLL            5
METRIC          CUSTOM 'vmstat 1 2 | tail -1' | awk '{print $14}'
SERVERS         9.12.30.53 9.12.30.54 9.12.30.55 9.12.30.61
POLICY          MIN
```

Figure 55. Our Iss_config Configuration File

POLL This is the polling period that ISS will use for this group. In our case, our polling period is five seconds. ISS will check each node at five-second intervals and update its opinion on which is the node with the lightest load. It will then plug this node's TCP/IP address into the `named.data` file. If the polling period is too short, you can put an added load onto the SP nodes and the network. If the period is too long, ISS's opinion about which is the node with the lightest load will be way out of date.

METRIC This defines one of three ways in which ISS will operate:

1. **LOADLEVELER**

   If this is specified, all of the LoadLeveler configuration information (for batch-job management) will also apply to

ISS. This saves having to define pools of nodes all over again if they have already been defined for LoadLeveler.

2. **CUSTOM**

   Here you can specify some command or user-defined entry that will be run on each of the nodes in the SP pool using the remote **rsh** command.

   An integer must be returned so that ISS can decide which node is the best candidate using its MIN or MAX choice.

   It is useful to test your **rsh** commands on the command line before you feed them into ISS. You can check that, first, they work and, second, they give the answer that you expect.

   As shown in Figure 56, **rsh** commands must be run on the remote nodes.

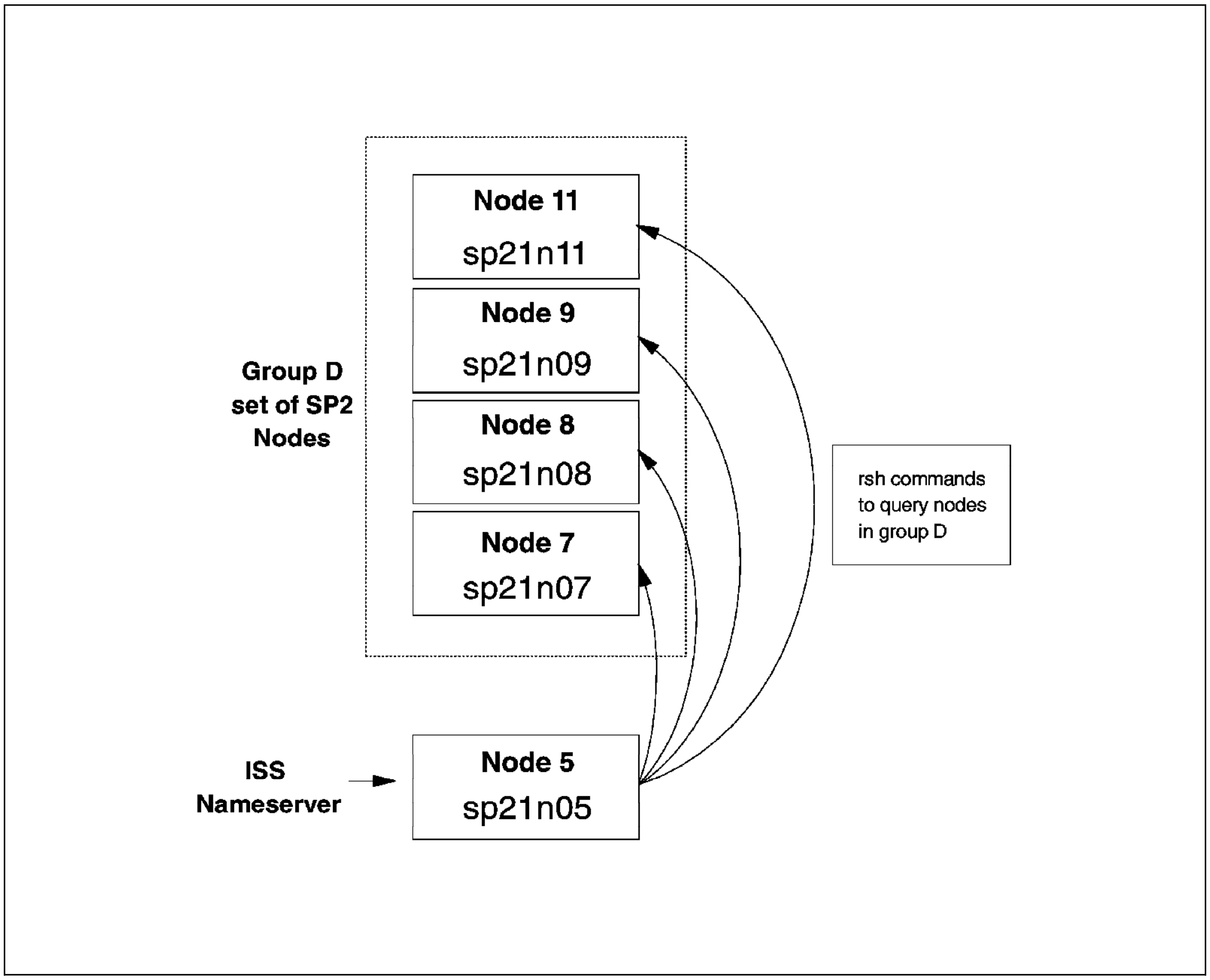


Figure 56. Running Remote Commands on ISS Nodes

3. **ROUNDROBIN**

   ISS cycles round each of the nodes in turn, changing the node according to the polling period. It does not matter, in this case, how busy the nodes are.

SERVERS Here you can list each of the TCP/IP addresses for your nodes in this pool. We can specify hostname or TCP/IP address.

POLICY This can be set to MIN or MAX as described above. The values returned from the CUSTOM METRIC are checked and either the minimum or maximum value selected, as required.

CLASS This allows you to restrict logins to nodes that are configured to run particular classes of LoadLeveler jobs. It only applies if you are already using LoadLeveler.

REQUIREMENTS You can specify LoadLeveler requirements that must be met before logins into specific nodes are allowed. This, again, only applies if you are using LoadLeveler.

PREFERENCES This also is used only with LoadLeveler, allowing you to favor nodes with particular LoadLeveler characteristics.

Table 15 shows which keywords are mandatory and which ones apply when using LoadLeveler only.

Table 15. Parameters to be Defined in the ISS Configuration File

KEYWORD	LOADLEVELER METRIC	CUSTOM METRIC	ROUNDROBIN METRIC
NAME	mandatory	mandatory	mandatory
SERVERS	ignored	mandatory	mandatory
POLL	mandatory	mandatory	mandatory
POLICY	ignored	mandatory	ignored
CLASS	mandatory	ignored	ignored
REQUIREMENTS	optional	ignored	ignored
PREFERENCES	optional	ignored	ignored

Once we have configured ISS, we can start it. The **iss** command can be found in the `/usr/lpp/LoadL/iss` directory.

In our case, we start ISS with the command **iss Iss_config &**.

If ISS is working properly, our client workstation can perform a remote login to the Group D group and we will end up on an appropriate node. For example, running the command **rlogin groupd_lite.sp2** will result in us logging into the node with the lightest load.

10.2.5 Practical Experiences Using Interactive Session Support

We note here a few things that we came across that may be useful to others who are setting up ISS. Some of these may be obvious, but they caught us when we were not careful.

The *named* daemon must be started *before* the ISS processes are started or you may get unpredictable results.

Unpredictable results, in our case, meant the `named.data` file being overwritten as a zero-length file. Making a backup copy of all of your configuration files makes good sense.

ISS must be able to perform **rsh** commands on the remote nodes to gather its information. This requires Kerberos access and a valid **rcmd** ticket. The problem we experienced in the current PSSP release was that the ticket lifetime is too short. Tickets expire after 21.25 hours. In Appendix B, "New Features For ssp.clients" on page 237 an APAR description is provided that discusses the latest enhancements to ssp.clients. One of the enhancements is a longer ticket lifetime.

Logging into the nameserver node and running the **kinit** command is not practical; this would have to be performed every day! We put an entry in cron to run the **rcmdtgt** command and gather a valid Kerberos (remote command) ticket every 12 hours. Although not covered in the product documentation, an alternative would be to use the standard TCP/IP **rsh** command for ISS.

At one stage, we were also running NIS for name resolution on our nameserver node. This interfered with our nameserving as we had not fully configured NIS at that stage. As TCP/IP Domain Nameserving and NIS Nameserving were not both required, we stopped NIS from running until it had been set up correctly. It is perfectly possible to run both together, but this would need to be carefully managed. Make sure that if you use NIS, you have defined all of the network information or it will not function correctly. If you run into problems, check whether the *ypbind* process (NIS) is running.

Finally, we hit some problems with the POLICY parameter when set to MAX in specific cases. We found instances where it did not calculate the correct node information. Our work-around was to use MIN, instead. For example, instead of measuring the idle time on nodes and selecting the MAX value, we measured the busy time and selected the MIN value. This had the same effect and worked correctly.

Part 5. System Monitoring and Control

Chapter 11. NetView for AIX on the SP

Version 3 of IBM NetView for AIX is a comprehensive network management tool for heterogeneous, multi-vendor devices and open networks. It supports network management in a multi-vendor TCP/IP network, provides management of TCP/IP devices that include Simple Network Management Protocol (SNMP) agents and monitors all IP-addressable devices.

NetView for AIX can manage a range of devices on the network including:

- Bridges
- Hubs
- Mainframes
- PCs
- Routers
- SP Nodes
- Workstations

NetView for AIX is capable of monitoring and controlling an SP system in a number of ways. In most cases, the functionality is very much the same as if managing a network of RISC System/6000s or other systems.

In the case of an SP system, a number of different configurations can be adopted. We look first at the three most commonly-used products in the areas of SP network and problem management, namely:

1. NetView for AIX Version 3
2. Systems Monitor for AIX Version 2
3. Trouble Ticket for AIX Version 3 Release 2

Let us have a look at how these can be implemented on the SP.

11.1 Overview of NetView for AIX

NetView for AIX is a licensed program product that is now fully supported on the SP.

NetView for AIX is an easy-to-use graphical tool that can perform many management tasks. It consists of a manager component that is typically installed on one system or node only. The agent software that runs on other devices on the network can be as basic as the Simple Network Management Protocol (SNMP) software, a standard part of the TCP/IP software in AIX. Alternatively, additional management information can be collected and processed by NetView for AIX if further agent software is used.

Systems Monitor for AIX (not to be confused with the SP Systems Monitor, or **spmon**) and Trouble Ticket for AIX are applications that can be used in conjunction with NetView for AIX.

NetView for AIX performs three major functions:

1. Configuration Management

   NetView for AIX provides automatic "discovery" of the network. NetView creates a graphical display or map for you when the application is initialized. You do not have to feed information about the network in beforehand. You can use the Graphical User Interface (GUI) in NetView to view, tailor and maintain these topological maps.

The network can be displayed as a physical, logical or geographic map using the very sophisticated GUI.

2. Performance Management

   NetView for AIX monitors network statistics and displays critical network resource status and statistical summaries for analysis and corrective action.

   Historical data can be accumulated for enhanced decision making.

3. Fault Management

   NetView for AIX can be used to verify the integrity of the network. It can utilize thresholding and filtering algorithms for easier alert notification. Where a known event is detected, it is easy to define and implement corrective action to SNMP traps that may be received.

NetView for AIX Version 3, in particular, provides some enhancements over the previous releases of the product. These include:

- Additional relational-database product support for Internet Protocol (IP) topology data, SNMP collected data and logged trap data, while taking advantage of the client/server capability of the individual relational database products. The databases supported in NetView for AIX are:

  – IBM DB2/6000 Version 2
  – INFORMIX Relational Database
  – Ingres Relational Database
  – Oracle Relational Database
  – Sybase Relational Database

  You have the option of selecting the database support at installation. NetView for AIX does not have to work with a relational database if one is not required.

- Enables you to define a backup NetView for AIX, which assumes management responsibility for any user-defined subset of network devices.

- Provides a Systems Performance Monitoring tool for monitoring the NetView workstation resource status and provides automated commands to correct performance problems.

- A number of enhancements related to the user interface, automatic discovery of the network and overall performance.

- Enables the customer to access the General Topology Manager (GTM) data with a GTM Application Program Interface (API).

- Provides full-function management of the local-area-network (LAN) resources with the IBM LAN Network Manager for AIX.

- Supports the Trouble Ticket for AIX product, which provides enhanced trouble ticketing, system inventory and notification.

- Provides extended systems management with the new IBM Systems Monitor for AIX V2 Mid-Level Manager (MLM) and System Information Agent (SIA) features.

11.2 Overview of Systems Monitor for AIX

Systems Monitor for AIX is one of the many applications that can be used with NetView for AIX. It is just as useful in an SP environment as when used in a networked RISC System/6000 environment.

Systems Monitor for AIX can perform two quite separate functions when used in conjunction with NetView for AIX.

1. It can act as an intermediate manager in conjunction with NetView for AIX Version 3 allowing for the distribution of systems and network-management tasks including discovery, status checking and threshold polling into the network. This frees up the network-management platform so that it can manage larger, more complex networks.
2. It provides detailed system-level information on RISC System/6000 workstations or SP nodes, allowing you to keep a close watch on them to ensure that they operate smoothly.

The product is split into functions that can be ordered separately; so, you only pay for the function you use.

- The Configuration Application, which is included in the basic license, is used to configure all of the Systems Monitors in your network.
- The Systems Information Agent (SIA) provides the detailed systems management and is priced such that it is practical to install on every workstation or SP node.
- The Mid-Level Manager (MLM) provides the distributed management functions and, therefore, would be installed on a subset of systems or nodes.

Systems Monitor for AIX is closely aligned with NetView for AIX Version 3. When NetView for AIX discovers a Mid-Level Manager on the network, it automatically assigns to the MLM the responsibility for managing and monitoring the segment on which the MLM is located. In addition, more management functions, including discovery and status checking, can be offloaded from NetView for AIX Version 3 to the Mid-Level Manager. The discovery function has been added to the MLM, allowing it to discover the local subnet. This eliminates the need for the NetView for AIX platform to poll across a wide-area network (WAN) and also allows faster discovery of new nodes. Status checking is also added to the distributed functions that it performs.

11.3 Overview of Trouble Ticket for AIX

Trouble Ticket for AIX Version 3 can be used as a stand-alone product. Typically, however, it is used in conjunction with NetView for AIX.

Trouble Ticket provides an integrated set of applications that includes functions such as trouble ticketing, system inventory and notifications. This unified set of applications enables pro-active management of the day-to-day problems encountered in the network environment.

Through Trouble Ticket and Systems Monitor, you can easily manage problems from initial discovery to problem closure. Diverse analysis tools and reports help circumvent problems before they occur. The system-inventory function provides the ability to keep track of detailed information on network devices, software applications and external services that support network operations.

Trouble Ticket for AIX tracks the problem history of each device connected to the network and each service used by the network, giving you immediate access to information that can help the network run more efficiently. It provides reporting that allows you to easily identify network trouble spots, chronic hardware and software failures, vendor hardware history and the status of all open trouble tickets.

Trouble Ticket allows easy storage and retrieval of problem- management information. It also has a database facility.

11.4 Options for Implementing NetView for AIX on the SP

In an SP environment, there are a number of options you can choose to run the software products that we have just discussed.

You must, first, make two decisions about how you want to manage your network:

1. Which system or node will run the NetView for AIX management software?
2. Which systems or nodes will run the client or agent software?

It is usual to select just one system to be the main focal point for network management. In the case of the SP, this focal point could be any SP node or, alternatively, any other RISC System/6000 on the network.

Note: It is not feasible to run the NetView for AIX management software on the SP Control Workstation. NetView for AIX assumes that it is the only application that runs on a system and can consume a lot of the resources of that system.

NetView for AIX performs polling of the network. The SP monitor that runs on the Control Workstation also polls the SP nodes. As a result, running both functions on the Control Workstation would cause excessive network traffic. The problem would be particularly prevalent when nodes were rebooting, with SP nodes sending out *bootp* packets and NetView reporting and querying nodes that were down.

As a result, you should not install the NetView for AIX management software on the Control Workstation. It is possible, however, to install the agent software on the Control Workstation and to manage it as any other device.

Figure 57 on page 171 shows a typical configuration for running NetView for AIX in an SP environment.

The management software running on an SP node (or another RISC System/6000 workstation if used for the focal point) is typically composed of:

- NetView for AIX V3
- Systems Monitor for AIX V2 (Mid-Level Manager)
- Trouble Ticket for AIX Server
- NetView for AIX trapgend subagent
- Systems Monitor SIA subagent
- Trouble Ticket Client

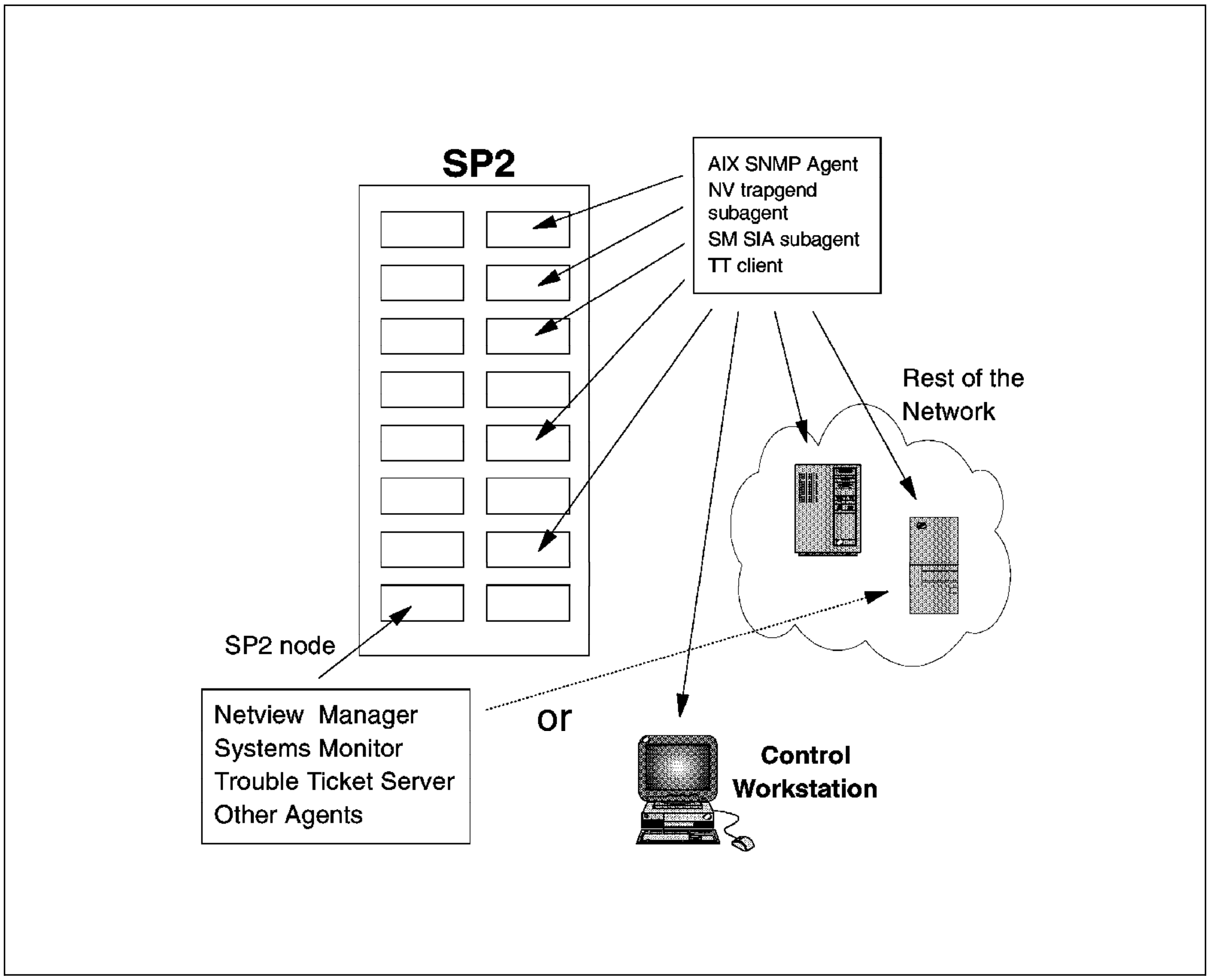


Figure 57. NetView for AIX on the SP

In addition, it is perfectly possible to install the SP client software on this node, allowing the *spmon* monitor to run on the node. This allows a single point of control for these tools. For example, it would be typical for the spmon monitor to be launched from a NetView screen, in the event of certain conditions, so that the network manager could monitor or change SP characteristics. Having the SP client software and spmon installed allows you to do this locally.

The clients or other systems on the network may include:

1. SP nodes
2. SP Control Workstation
3. Other workstations or systems

These clients would typically have the following agent software installed to provide data to the NetView manager node:

- AIX SNMP Agent

  This is the standard SNMP software provided with TCP/IP. It provides basic SNMP MIB (Management Information Base) data to NetView for AIX.

- NetView trapgend subagent

  Further, more-detailed information can be provided if this software is installed on SP nodes or other systems on the network. It can very easily be installed remotely from the NetView menus. Remote hosts to be installed are selected, the passwords entered and the install happens automatically.

- Systems Monitor SIA subagent

  This client software allows detailed monitoring of specific events on this node. For example, if this application should fail, particular critical processes could be monitored and corrective action initiated.

  This can provide added functionality over using just NetView for AIX, but it can also offload work from the central manager machine. In a large network, this can be important.

- Trouble Ticket client

  This client software allows users to open and manage problems within the Trouble Ticket problem-management structure, but with local access. The problem ticket details and the data are, of course, stored on the server.

  SNMP traps can also easily be configured to automatically open additional incidents.

It can be very sensible, if required, to distribute some of the software functions amongst SP nodes. For instance, you might choose to run the Trouble Ticket server or a relational database for storing information on separate nodes. It is recommended that only NetView for AIX and its associated management applications run on the selected node. It is not, for example, sensible to run other business applications on the management node. This is due to the performance characteristics of NetView. This advice would apply wherever NetView for AIX is running.

11.5 Experiences with NetView for AIX on the SP

We had a limited amount of time to test these products on the SP. However, the process to install and manage the products is almost identical to that on a standard network.

Only the products described in this chapter are supported on the SP. Earlier NetView products, such as NetView/6000, Systems Monitor/6000 and Trouble Ticket/6000 require Motif 1.1.4. The SP software requires Motif 1.2; therefore, the earlier NetView products are not supported in an SP environment.

NetView for AIX requires AIX Version 3.2.5. In addition, the following licensed program products and fixes are required:

- X11fnt.ibm850.pc.fnt V1.2 or later
- X11fnt.coreX.fnt V1.2 or later
- X11rte.motif1.2.obj U433184, U428196, U432350
- X11rte.obj U428198, U428199, U432909, U431144
- bosnet.snmpd.obj U428290

At this point in time, NetView for AIX does not fully exploit the features of the SP. The nodes are seen very much like a network of systems rather than one entity.

SNMP polling running over an SP network can impact parallel jobs depending on the nature of the job. SNMP polls in a “roundrobin” fashion; this may interrupt parallel jobs causing them to run for longer than if no SNMP polling was taking place.

In order to manage the SP switch network, the network-management system must have a path to the local switch network. If the manager is in the same sub-network as the SP, then static TCP/IP routes can be added to the manager system, allowing it to report the correct status of the SP switch. If the manager is not in the same sub-network, then a route must be supplied by an intermediate router.

Some specific details and information about the High-Speed Switch network are not fully understood by NetView for AIX. NetView can monitor the status of the Switch network, but it cannot report specific, detailed information about the adapter.

NetView for AIX or any other IP-based management tool will not be able to manage the SP High Performance Switch adapters on individual nodes if they are using the user-space protocol.

These limitations not withstanding, NetView for AIX is still a very useful and useable tool.

11.6 IBM Performance Toolbox/6000

Performance Toolbox/6000 is a performance-monitoring tool from IBM that can display graphical information about systems in real time and allows users to easily monitor performance.

The menu-driven interface allows users to easily use default monitors, create customized monitors, record and playback information and perform a range of sophisticated performance-management tasks. Performance Toolbox/6000 has two components:

1. PTX/6000

   This is the manager part of Performance Toolbox/6000. It is normally only run on one system in the network, often an SP node. It could equally be any other RISC System/6000 system on the network.

   The Control Workstation could potentially be used for running this software. but caution must be exercised. A user of PTX/6000 may open up a large number of windows and can display in real time a lot of graphical information. This workload can impact the machine on which it is running depending on its configuration.

   In particular, PTX/6000 uses X Windows which can consume a great deal of memory resource.

   If the SP Control Workstation is to be used for running PTX/6000 as well as its other responsibilities, you must make sure that the configuration of the Control Workstation is suitable. Additionally, a faster LAN than Ethernet may be required to attach the Control Workstation to the SP nodes where PTX/6000 agents are running.

2. PAIDE/6000

   Performance Aide/6000 (PAIDE/6000) is the agent component of Performance Toolbox/6000. It is required on each node or system that is to be monitored by Performance Toolbox/6000. It should also be installed and running on the PTX/6000 node.

PAIDE/6000 works in a very efficient manner to collect data about performance and feeds it back to PTX/6000. It typically produces overhead of only a few percent in terms of CPU workload. By default, if after five minutes it is not asked for information, PAIDE/6000 will stop collecting information.

PAIDE/6000 has an open interface allowing other application data to be fed into Performance Toolbox so that applications can be monitored from a performance point of view. Performance Toolbox/6000 can also output data to SNMP so that NetView for AIX can collect the performance information.

Figure 58 shows how Performance Toolbox could be used on the SP.

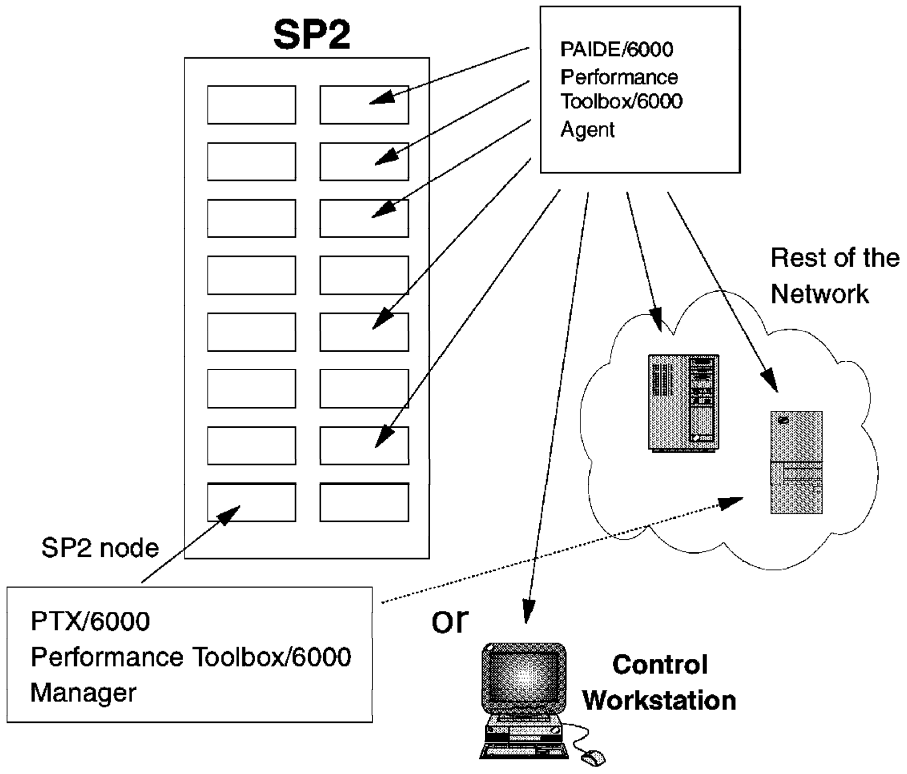


Figure 58. Using Performance Toolbox/6000 on the SP

Part 6. System Administration and Operations

Chapter 12. Print Management

This chapter describes printing of large amounts of data on a high-speed printer that is channel-attached to the 9076 SP using Print Services Facility/6000 (PSF/6000).

Additional information is provided in the document *IBM AIX Print Services Facility/6000: Printing with the IBM 9076 Scalable POWERparallel Systems*, available from your IBM Representative (on MKTTOOLS as SP2PSF TERS3820).

12.1 9076 SP Printing Concepts

Originally, due to the limited configurability of the 9076 SP1 (precursor to the 9076 SP), users had to rely on an outside server to satisfy their need for producing hard copies. Although the SP family has evolved considerably since its inception, the support for printing has not changed. An SP node is designed to use a unique remote host, PRINT_HOST, as a print server. The AIX operating-system printing environment is disabled if a response other than "false" is typed in as an answer to the *print-management configuration* question in the *site environment file* at installation time. All of the AIX print commands are renamed to commandname **.AIX**. Symbolic links are created to point the users to an SP print-management switch, *pmswitch*, that invokes the equivalent AIX command just issued. For example, the AIX command **enq** is a symbolic link to **/usr/lpp/ssp/bin/pmswitch**.

Why do all of this instead of using remote AIX print facilities? The answer to this question again lies back in time. All of the SP1 nodes were diskless clients of some kind of file server. Whatever remote printing environment was defined by this server was the environment that diskless clients would get if they were to use AIX printing commands. To provide some flexibility and independence, the print-management system of the SP1 allowed users to move data to the remote host without going through a queueing system and to run their print command on the remote host as if it were a local command. To allow the execution of remote shell commands from the SP nodes requires only the setup of an /etc/host.equivalent at the remote host. Not all of the AIX printing-subsystem commands and options are supported. The intent here is clearly to provide a simple, basic and temporary solution while AIX improves its current print-management system.

Control and data flows of SP printing combined with AIX and PSF/6000 at the remote PRINT_HOST are shown in Figure 59 on page 178 and Figure 60 on page 179.

12.2 Installation and Customization of a Channel-Attached 3825 with PSF/6000

In our SP configuration (see Chapter 1, "System-Management Project Environment" on page 3), the print server is installed, as expected, in group S, on node sp2sw11. It is assumed to serve the entire community of developers in groups D and T.

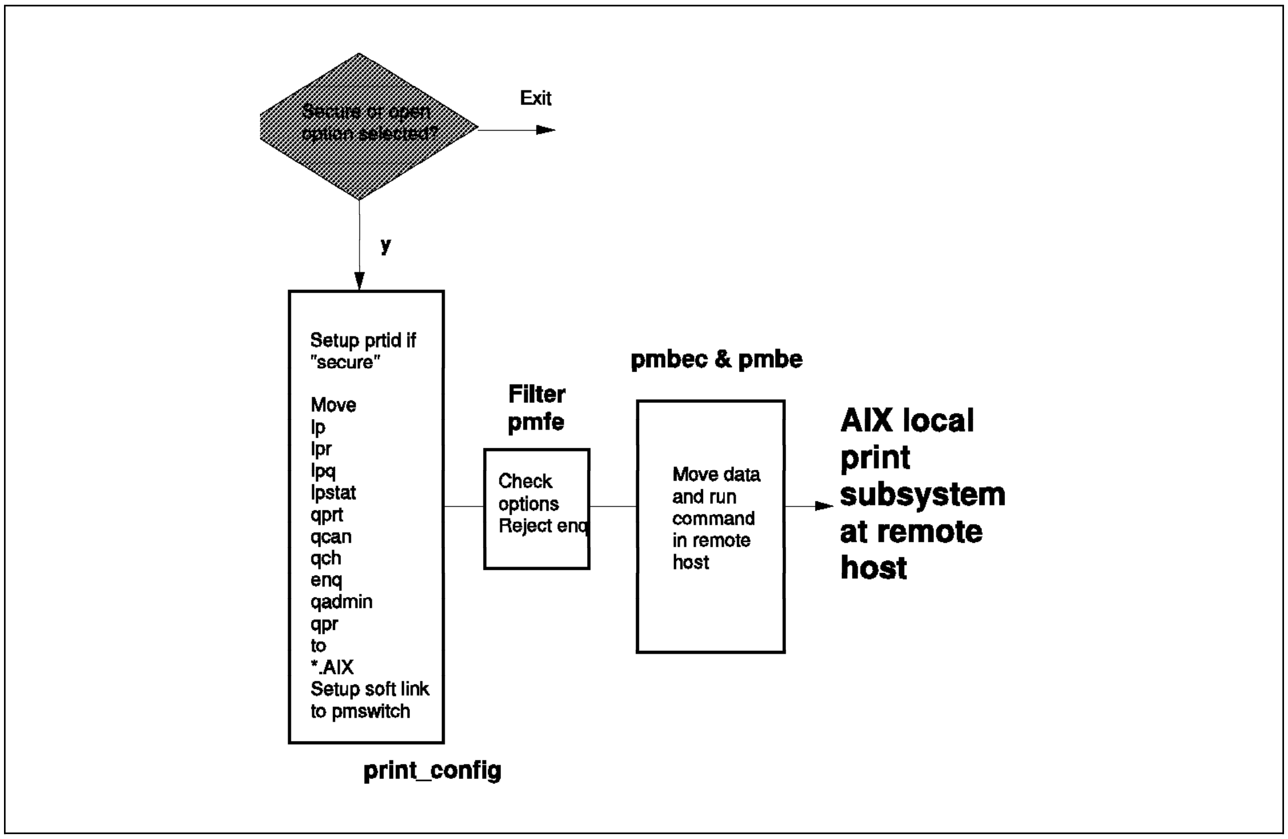

Secure or open option selected?
Exit
y
Setup prtid if "secure"
Move
lp
lpr
lpq
lpstat
qprt
qcan
qch
enq
qadmin
qpr
to
*.AIX
Setup soft link to pmswitch
print_config
Filter
pmfe
Check options Reject enq
pmbec & pmbe
Move data and run command in remote host
AIX local print subsystem at remote host

Figure 59. SP Print-Management System Configuration

12.2.1 Hardware Requirements

Aside from the printer itself, a 3825, the following hardware components are needed to set up the connection with the printer:

- A S/370 channel emulator RPQ 8K1922. See Appendix G, "S/370 Channel Emulator (RPQ 8K1922) Hardware Characteristics" on page 375 for hardware ordering characteristics.
- Channel cables terminated by black/grey endings, as shown in Figure 61 on page 180. Cables and emulator are part of the same RPQ.

Optionally, if you have a 3814 switch-management system, you may connect the 3825 to the 3814 and attach the S/370 channel emulator to the 3814. If an extension is needed, remember to connect the white endings of the cables to the black endings of the cables attached to the emulator.

12.2.2 Software Requirements

The channel-emulator driver code comes with PSF/6000. So, the only required software is really PSF/6000 version 1.2 or higher.

12.2.3 Installation

The S/370 emulator occupies one micro-channel slot. The entire set, emulator and channel cables, is shown in Figure 61 on page 180.

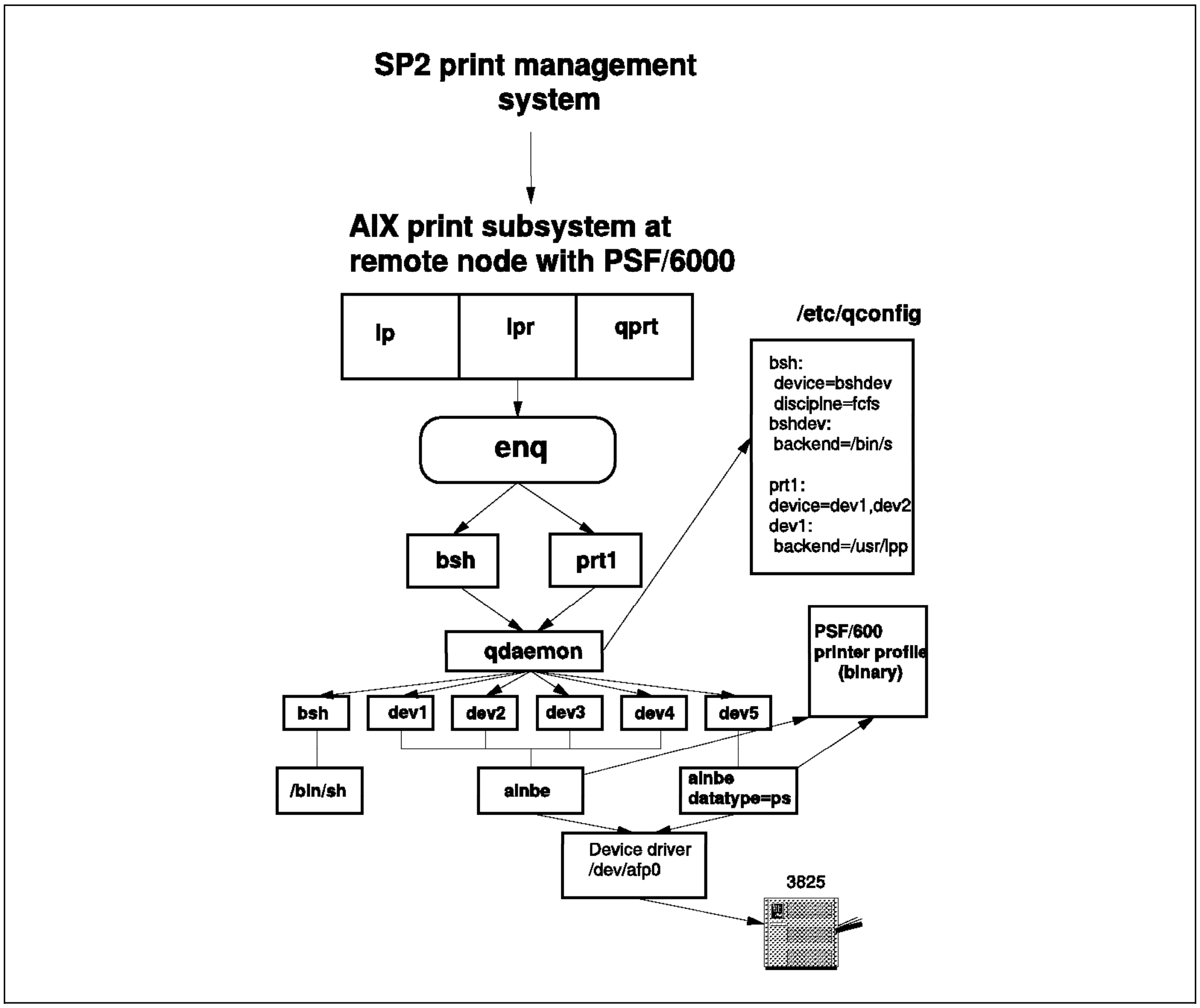


Figure 60. AIX Print-Management System and PSF/6000

Once the emulator is inserted into node sp21sw11 and connected to the cables, no special microcode needs to be loaded. As mentioned in the previous section, the driver code comes with PSF/6000. Also, there is no need to configure the emulator. This will be done by PSF/6000 at installation time. So the emulator will not show if you do **lsdev**, even when you run **cfgmgr**. But do not worry; after the installation of PSF/6000, it will.

The final hardware setup is shown in Figure 63 on page 181. Node sp2sw11 of group S is the print server. All of the software customization is done on that node.

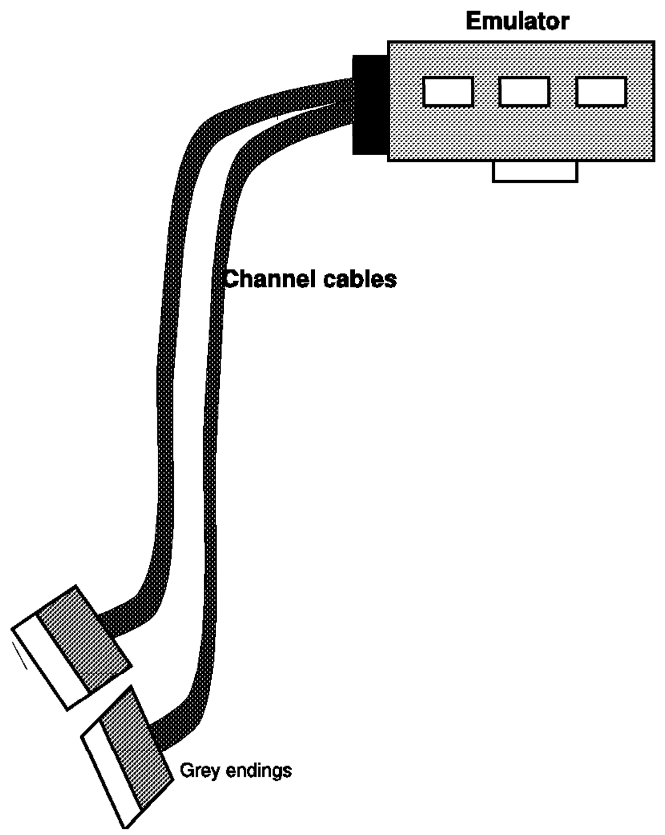


Figure 61. S/370 Channel Emulator and Channel Cables

12.2.4 PSF/6000 Installation and Customization

The installation of PSF/6000 is done through **installp** with SMIT. When it ends successfully, the emulator is also configured; its status should be "available," as shown in Figure 62.

```
[root@sp21n11] / >lsdev -C -l chna0
chna0 Available 00-04 IBM S/370 Channel Emulator/A Adapter
[root@sp21n11] / >lsdev -C -l afp0
afp0 Available 00-04-00 IBM S/370 Channel Emulator/A Printer Driver
[root@sp21n11] / >_
```

Figure 62. Checking the Status of the Channel Emulator

Figure 62 also shows that the drive code, afp0, is loaded. If it is not loaded, SMIT can be used to load it again. The system is now ready for printer and device configuration.

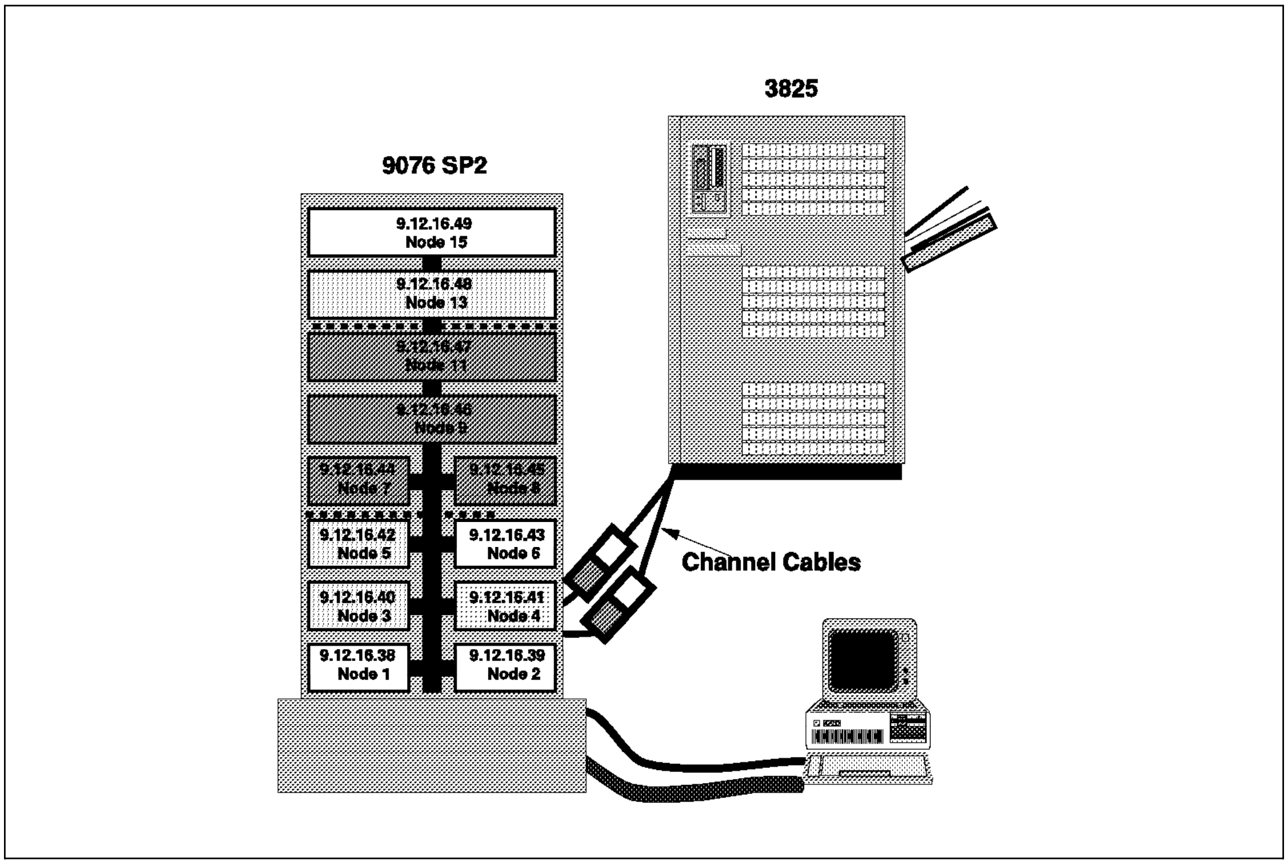


Figure 63. RS/6000 SP and Channel-Attached 3825 Configuration

12.3 Configuring a Channel-Attached High-Speed Printer

The printer and access path must be made known to PSF/6000. To do this, invoke SMIT using the following fastpath:

```
smit psfcfg
```

Then select:

```
 └─► Add a S/370 Channel-Attached Printer
```

Figure 64 on page 182 shows an example of the data that should be entered in the SMIT panel.

The slot number in Figure 62 on page 180 is given by the command **lsdev**. Although, the slot number is designated by lsdev as 00-04, in our case, it is entered as a single digit (4) in SMIT. The control unit address is the address given to the control unit of the printer by the customer engineer upon installation of the printer. In our case, it is **b0**. The number of default queues is set to four, which seems reasonable. The maximum number of queues allowed is nine. Those queues can process any type of data using the **-o datatype** option of the enq command. Unfortunately, the enq command is not supported by the SP print-management system. Nor is the **-o datatype** option supported with **lpr** or **qprt**. There is, however, a way to print PostScript files on the SP by creating special queues, as we will see in 12.4.1, “Setting Up a PostScript Print Queue” on page 182.

```
                    Add a S/370 Channel-Attached Printer

Type or select values in entry fields.
Press Enter AFTER making all desired changes
                                                         [Entry Fields]
* Data stream type                                     IPDS
* Printer NAME                                        [prt1]
* SLOT number for S/370 Channel Emulator/A Adapter     4
* Control unit ADDRESS                                [b0]
* Number of QUEUE DEVICES                             [4]
  Description                                         [Channel Attached]
  To specify other characteristics after you add
    a printer, select Change Characteristics of
    a Printer from the
    PSF/6000 Printer Definition panel.
```

Figure 64. Adding a Channel-Attached Printer Using SMIT

12.4 Setting Up the Print Server

As noted in the previous sections, the SP print-management system must be disabled on the print server (only if it was enabled). To do this, enter:

```
export PRINT_CONFIG="false"
/usr/lpp/ssp/install/bin/print_config
```

All AIX print commands are restored and will be used to support local, as well as remote, SP and AIX printing. For remote AIX printing, remote print queues must be set up, as usual. For remote SP printing, nothing special needs to be done at the print server nor with the /etc/host.equivalent. Printing on the server can begin as soon as SP print management is disabled and the printer is connected and configured. This node will now support **enq** and all of the options of the print commands. After the ps2afpd daemon is started, it can print PostScript files using the **-o datatype** option of the enq command. However, for SP client nodes, a special PostScript queue must be set up.

12.4.1 Setting Up a PostScript Print Queue

A special PostScript queue can be added using the following SMIT fastpath:

```
smit mklque
```

SMIT displays the Add a Local Queue panel, as shown in Figure 65 on page 183. From this panel, change the BACKEND program to:

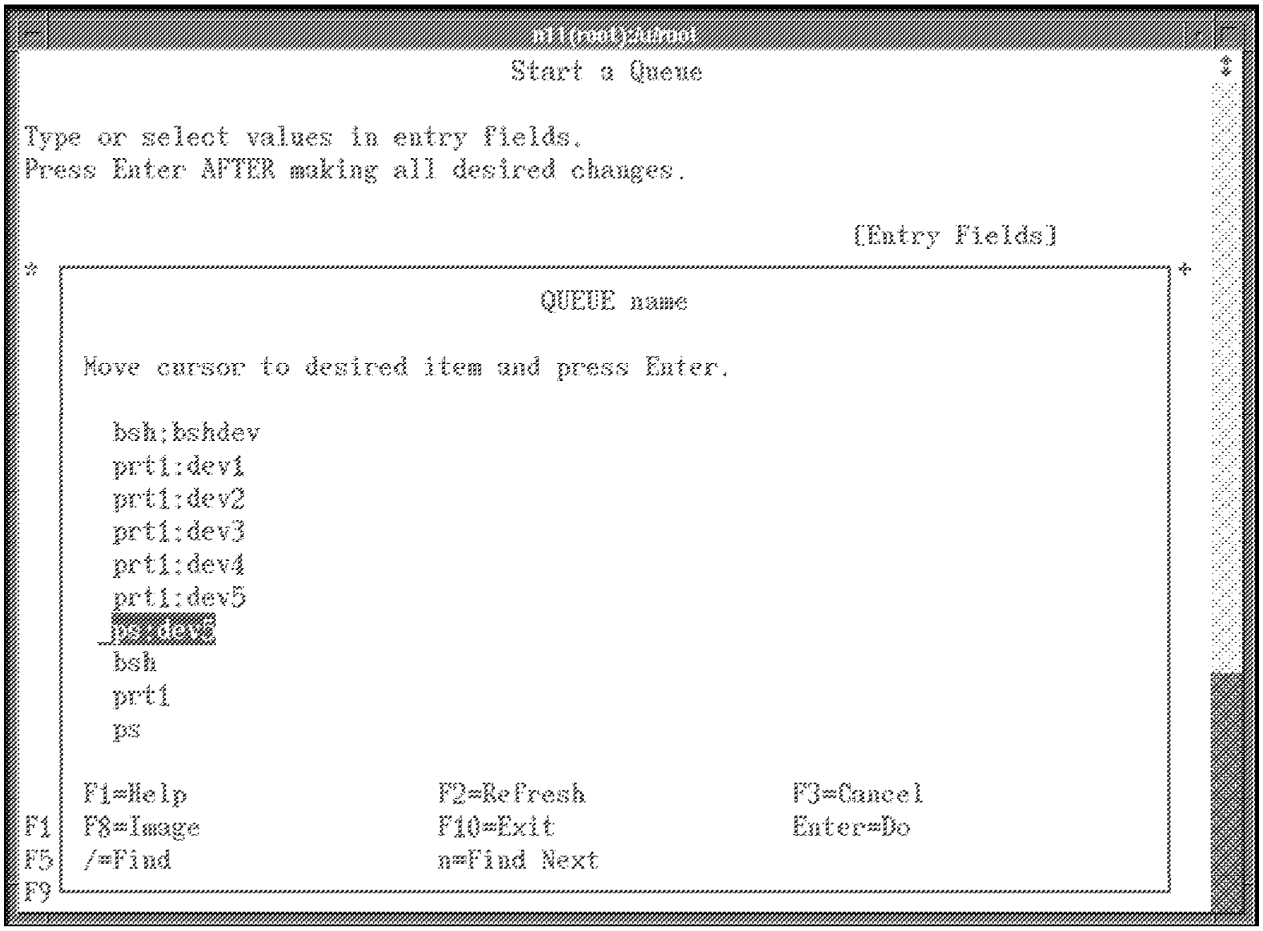
```
/usr/lpp/psf/bin/ainbe prt1 datatype=ps
```

where prt1 is the name of the printer defined in Add a S/370 Channel-Attached Printer panel shown in Figure 64.

```
                         Add a Local Queue

Type or select values in entry fields.
Press Enter AFTER making all desired changes
                                                       [Entry Fields]
* NAME of queue to add                              [ps]
  ACTIVATE the queue?                                yes
  Will this become the DEFAULT queue?                no
  Queueing DISCIPLINE                                first come first s
  ACCOUNTING FILE pathname                          []
* NAME of device to add                             []
  BACKEND OUTPUT FILE pathname                      []
  ACCESS MODE of backend output file                 write only
* BACKEND PROGRAM pathname                          [/usr/lpp/psf/bin/ainbe]
  Number of FORM FEEDS prior to printing            [0]
  Print HEADER pages?                                always
  Print TRAILER pages?                               always
  ALIGN pages between files within jobs?             yes
```

Figure 65. Setting Up a PostScript Queue for SP Remote Client Nodes

After the queue is defined, it can be verified by starting it. To do this, use SMIT and move to the Start a Queue panel, as shown in Figure 66. The newly-created PostScript queue, ps, should show here. Press Enter to start it.



```
n11(root):/u/root
                          Start a Queue

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

                                                  [Entry Fields]
*
                          QUEUE name

   Move cursor to desired item and press Enter.

     bsh:bshdev
     prt1:dev1
     prt1:dev2
     prt1:dev3
     prt1:dev4
     prt1:dev5
     ps:dev6
     bsh
     prt1
     ps

   F1=Help              F2=Refresh           F3=Cancel
F1 F8=Image             F10=Exit             Enter=Do
F5 /=Find               n=Find Next
F9
```

Figure 66. Starting a Print Queue

Other special queues, like a DBCS ASCII print queue, can also be created in a similar manner.

12.4.2 Disk Space

One of the reasons we chose node sp21sw11 as the print server within group S is because it has plenty of disk space (4GB). When printing PostScript, the conversion daemon, ps2afpd, needs quite a large amount of space in /var. If several sp2 users are printing PostScript files concurrently, the required amount of space could be huge. In general, count on 2MB of disk space per node. Change the size of /var accordingly.

12.5 Setting Up SP Printing on Client Nodes

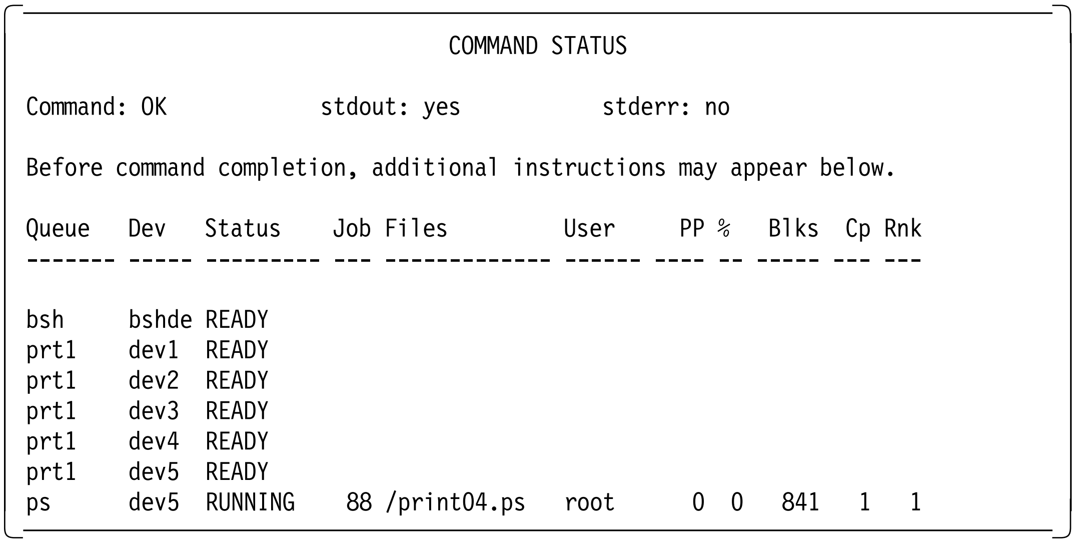
For remote SP printing, the kerberized version of rsh must be set up on remote nodes. To do this, ensure that the directory /usr/lpp/ssp/rcmd/bin is searched before /usr/bin. The variable PRINT_HOST must be set to the print server, directing rsh to the server; for example:

```
export PRINT_HOST=sp21n11.itsc.pok.ibm.com
```

To print from node sp21n04 after PRINT_HOST has been set to sp21n11.itsc.pok.ibm.com, we use the following command:

```
lpr -P ps /tmp/print04.ps
```

With rsh, the command is executed immediately on sp21n11, and the data is queued on device ps, as shown in Figure 67.



```
                                COMMAND STATUS

Command: OK           stdout: yes          stderr: no

Before command completion, additional instructions may appear below.

Queue   Dev   Status    Job Files         User     PP %   Blks  Cp Rnk
------- ----- --------- --- ------------- ------ ---- -- ----- --- ---

bsh     bshde READY
prt1    dev1  READY
prt1    dev2  READY
prt1    dev3  READY
prt1    dev4  READY
prt1    dev5  READY
ps      dev5  RUNNING    88 /print04.ps   root      0  0   841   1   1
```

Figure 67. Data Queueing on Server Node

12.6 Hints and Tips

After the installation of PSF/6000, check the connection with the printer. With slot number 4 housing the emulator and address b0 assigned to the control unit, the command to check for the connection is shown in Figure 68 on page 185. As you can see, the response to that command was negative (printer not defined); however, printing on the 3825 was performed with no problem. So, if you see that message, you can ignore it. However, when we tested by disconnecting the channel cables, we got the following message:

```
printer not connected at address 4b0
```

So, if you receive a message indicating that your printer is not connected, you had better check your cables. (You should also see your queues going down one after another).

```
[root@sp21n11] / >sense 4b0
printer not defined at address 4b0
[root@sp21n11] / >_
```

Figure 68. Verifying Printer Connection

If you use a channel-cable extension, make sure that the ends that connect to the cables attached to the emulators are off-white in color, since the ends of the emulator cables are black or grey. You do not have to use a metal plate to connect the cables; you may connect them directly.

If you print a PostScript file and nothing happens, check if the daemon *ps2afpd* was started. If it was not, start it by entering:

```
/usr/lpp/psf/bin/ps2afpd
```

If the daemon was started and the file still does not print, check if /var is full. If so, delete files or increase the size of /var.

Chapter 13. SP User Management

This chapter discusses how an SP System Administrator can manage users and user directories. SP systems can be categorized into two major groups:

- Parallel compute or database servers
- LAN-consolidation machines

In the first case, you would like the SP to appear as one machine. In other words, it is of no importance on which node a user logs in or from which node a user submits a parallel job. Therefore, a user must be able to treat the SP as one logical machine, requiring that certain system configuration definitions comply with that need. Refer to 13.3, "The Berkeley Automounter (Amd)" on page 194 for more information on this matter.

In the second case, the SP is probably configured as a partitioned system, where several groups of nodes serve different functions. One or more applications are grouped together in logical partitions throughout the SP, for instance, as we use in this book, in a Server group, a Development group and a Database group. Preferably, you would like to limit access of certain users to certain groups, for instance, an application user (Server group) must not be able to log into the Database group of nodes. This subject is addressed further in 13.4, "SP File Collections" on page 206 and in 10.2, "SP Nameservers and Interactive Session Support" on page 144. Those topics that are addressed specifically in this chapter are:

- Managing SP users
- Using Amd
- Using File Collections
- Using NIS

13.1 Managing SP Users

This section describes how you can manage SP users. The PSSP package provides a couple of new commands specifically for the purpose of managing users in a cluster. They are discussed later in this section.

User Management on AIX and on clusters of AIX machines can be viewed in the following way:

- The existence of a user: To AIX, a user is defined by a user entry in the /etc/passwd file. This means that, if you want to have user IDs available on all nodes in the network, you must define entries for those user IDs in the /etc/passwd file. You can add users to all nodes by using the standard AIX user-management tools on each node separately, but you had better use replication mechanisms. File-Collection Management, one mechanism that can be used for this purpose, is discussed in detail in 13.4, "SP File Collections" on page 206. Another tool that can be used is Network Information Service (NIS), which simulates the existence of a user in the /etc/passwd file. See 13.5, "Network Information Service" on page 223 for further discussion.
- Access permission: When using replication mechanisms for user IDs and user credentials, you must consider whether certain users are allowed to

login everywhere in the cluster or network. There are several ways to control remote access:

- Disabling or enabling remote services per node using TCP/IP: TCP/IP provides several services that are maintained in the /etc/inetd.conf file. You can configure each individual service on each node to be available or not. Also, you can control remote AIX access services (rsh, rexec and rlogin) per host, per user or per group of users in the /etc/hosts.equiv file. SP remote access is controlled using the .klogin file.
- Disabling or enabling individual access using the login mechanism: If all remote network services must be available, for instance, for consistency of node configuration, you can also control individual user access by using the /etc/security/user file. In this file, you can specify administrative attributes for each user (in this case, whether or not a user can login or login remotely). The command provided with the SP is **spacs_cntrl**, discussed in 13.2, “Access Control” on page 193.
- The home directory: Each user has must have access to a *home directory*. This home directory usually resides on the node you are working on, most likely on your own workstation. To enable transparent access of the user′s home directory in a network, you can create home directories on each individual node in the network. You had better make use of tools that provide transparent access to your “original” home directory throughout the network. Network File System (NFS) provides such a service. Specifically for the SP, you can make use of the Automounter (Amd) facilities. Automounter is a utility built on top of NFS, enabling transparent management of both NFS and mounts. Amd is discussed in 13.3, “The Berkeley Automounter (Amd)” on page 194.

So, when adding new users to your SP system, your checklist of user characteristics could look like this:

- When using your SP as a parallel machine, where would you like to have the physical location of the users′ home directories? On the Control Workstation, which is the default, or on one or more of the SP nodes?
- How much data do you anticipate having? If a lot, consider balancing the physical location of the data on several nodes in your parallel environment.
- When using your SP as a LAN-consolidation machine, define user home directory servers within the predefined grouped nodes (as in our example, group Server, group Development or group Database)
- Are users restricted to log into certain groups of nodes only?
- Are users going to share their home directories throughout the SP (that is, using NFS to distribute the home directory)? In that case, consider using Amd.
- Define whether to use NIS for users′ credential distribution or File-Collection Management.

Once you have decided on these points, you can work out an implementation strategy. For each of the mentioned points, System-Management procedures must be worked out.

13.1.1 Adding SP Users

As discussed earlier, you can use your SP as a parallel machine or as a LAN-consolidation machine. Here, we discuss two ways to add users. The first just uses the "standard" user-management tools and commands provided by AIX. The second involves the user-management tools provided with PSSP.

13.1.1.1 Using AIX User-Management Tools

If you want to use only the standard AIX user-management tools, specifying "false" in the appropriate field of the Site Environment Information panel of SMIT will not add the User Administration Interface to SMIT. Note that this only removes the interface, not the underlying commands, so you can reset it at any time. But, should you configure the system to use the User-Management Interface, you can still use the standard AIX tools, since the SP User-Management tools are built on top of them. So, adding a user on a node would require the following steps:

- As root, log into the node on which you wish to add the user.
- Execute **mkuser** `<user-name>` with the appropriate user attributes.
- Define the initial user's password.

You will have created a user equivalent to a normal RISC System/6000 AIX user; nothing is shared. This is, of course, one extreme.

13.1.1.2 Using the PSSP User-Management Tools

PSSP provides the tools necessary to add a typical SP user. Say, for example, you define the following characteristics for your new SP user:

1. Home directory is on node 13, `sp21n13`.
2. Uses an NFS-shared home directory.
3. May only log into group Server; cannot log into group Database or group Development.
4. Credentials must be known throughout the SP.

Based on these user characteristics, the commands and actions that must be taken, in order, are:

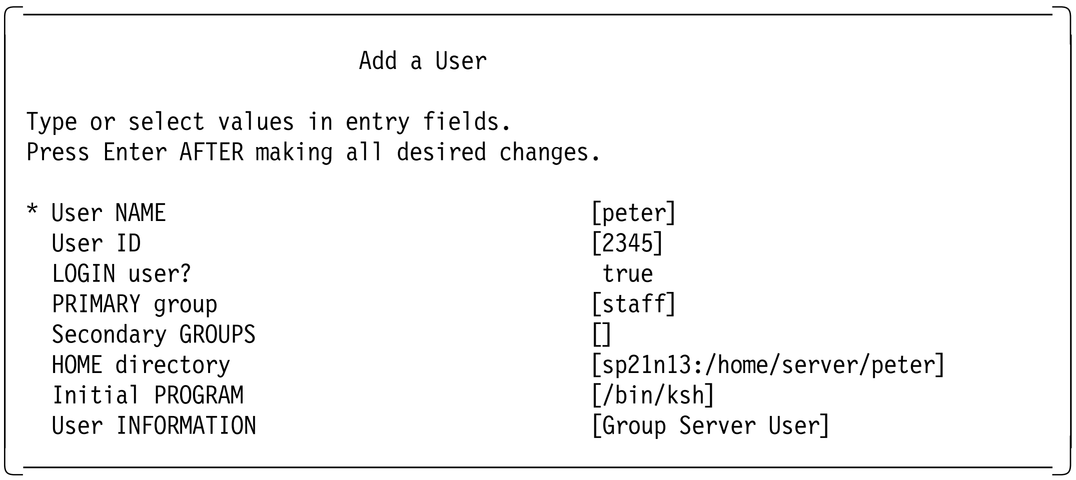
1. Go to the SMIT panel to add a new user. You can reach it from two different sections of SMIT, either through:

   ```
   └► Security & Users
       └► 9076 SP Users
   ```

   or,

   ```
   └► 9076 SP System Management
       └► 9076 SP Users
   ```

   Selecting the option **Add a User** brings up the dialogue shown in Figure 69 on page 190.



```
                          Add a User

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

* User NAME                                   [peter]
  User ID                                     [2345]
  LOGIN user?                                  true
  PRIMARY group                               [staff]
  Secondary GROUPS                            []
  HOME directory                              [sp21n13:/home/server/peter]
  Initial PROGRAM                             [/bin/ksh]
  User INFORMATION                            [Group Server User]
```

Figure 69. Adding an SP User

The command used by SMIT is the **spmkuser** command. It creates the user's home directory, (/home/server/peter) on node sp21n13, sets the initial program of the user to /bin/ksh, updates the /etc/group and the /etc/passwd files on the Control Workstation and adds an Amd map for this user (if Amd Configuration is set to "true" in the Site Environment Information panel of SMIT).

Note: We were not able to make use of the field LOGIN user?. This field setting should alter the /etc/security/user file, but we did not see any changes occurring. This is because, with SP, the LOGIN user? signifies changes to /etc/password. To change the /etc/security/user file, you should use **spacs_cntl**, AIX smit or edit the file. Remember, this should be done on a per-user basis.

2. To make the user's home directory available throughout your SP, you must use NFS (or AFS). Update the /etc/filesystems file on all nodes except for the node containing the physical home directory and **export** the filesystem containing the physical home directory.

   Recommended, however, in this kind of environment is the use of Amd. Amd, built on top of NFS, takes care of maintaining the multiple /etc/filesystem files. The **spmkuser** command automatically creates an entry for the new user in the /etc/amd/amd-maps/amd.u file, with the correct key values. To activate these maps on your SP, you must replicate the map files and refresh the amd daemon. Refer to 13.3, "The Berkeley Automounter (Amd)" on page 194 for details.

   Note: For read-write accessibility in a network, the exported filesystem containing the physical home directory should look like this:

   For our example on node sp21n13, in the /etc/exports file:

   ```
   /home     -root=sp21n03:sp21n04:sp21n05,
             -access=sp21n03:sp21n04:sp21n05
   ```

   Or, if using *netgroups*:

   ```
   /home     -root=sp21n03:sp21n04:sp21n05,
             -access=<netgroup-name>
   ```

3. For restricting user logins on certain nodes or groups of nodes, use the /etc/security/user file (as mentioned earlier). But, remember that the /etc/security/user file should be specific to each node. So, normally, this

file should not be distributed through the File Collections. However, if the following are true, you may use the File Collections as the distribution mechanism:

a. The same users are defined on the set of nodes, and
b. You will not be changing specific users′ access on any node in the set.

But, you *cannot*:

a. Use the **spacs_cntl** command. Because it works on a per-node basis and alters /etc/security/user directly, these changes would be lost.
b. Run job submissions on the set of nodes. The Resource Manager uses spacs-cntl to alter user access on a per-node basis to reserve nodes for parallel jobs. If File Collections were used, the modifications would be lost.

4. At this stage, you have defined the user. However, only the Control Workstation can handle a login for this user at this point. To make the user known to the system, either use NIS or File-Collection Management for distribution of the user configuration files. If using File Collections, the default value for system configuration-file refresh is one hour. When using NIS, you must rebuild the NIS maps and distribute them.

   Refer to the 13.5, "Network Information Service" on page 223 and 13.4, "SP File Collections" on page 206 for configuration and usage details on these tools.

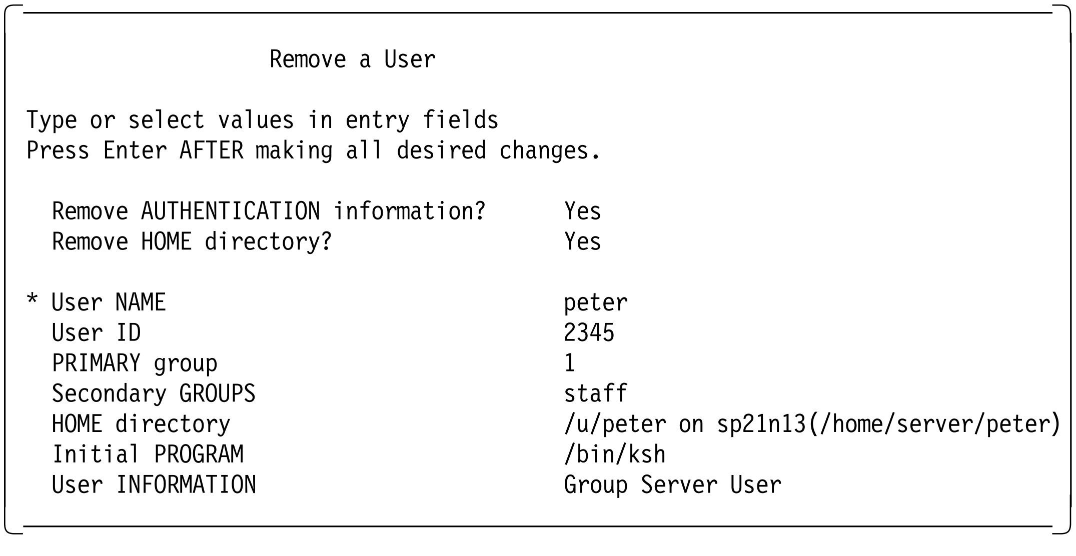
   Note: For user-login restrictions, you may maintain several copies of the /etc/security/user file or use the **spacs_cntrl** command on each node. For details, refer to 13.4, "SP File Collections" on page 206.

13.1.2 Deleting SP Users

As is usual when deleting or destroying material, removing users from the system is much easier. Go to SMIT and select the following from the 9076 SP Users panel:

└→ Remove a User

After specifying a user, you are prompted with the panel shown in Figure 70.



```
                    Remove a User

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

  Remove AUTHENTICATION information?      Yes
  Remove HOME directory?                  Yes

* User NAME                               peter
  User ID                                 2345
  PRIMARY group                           1
  Secondary GROUPS                        staff
  HOME directory                          /u/peter on sp21n13(/home/server/peter)
  Initial PROGRAM                         /bin/ksh
  User INFORMATION                        Group Server User
```

Figure 70. Deleting an SP User

Removing AUTHENTICATION is related to the login password information, not to Kerberos. Removing HOME directory removes the directory and its contents. The other lines are used to confirm that this is the user intended for removal. The command used by SMIT is **sprmuser**.

13.1.3 Changing SP Users

To change the attributes of an SP user, specify the following SMIT option from the `9076 SP Users` panel:

└► `Change/Show Characteristics of a User`

The SMIT panel shown in Figure 71 allows you to change all values (except the user name).

Note: As mentioned before, a change to the LOGIN user? field did not seem to be reflected in the `/etc/security/user` file.

```
Change/Show Characteristics of a User

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

* User NAME                                    peter
  User ID                                     [2345]
  LOGIN user?
  PRIMARY group                               [1]
  Secondary GROUPS                            [staff]
  HOME directory                              [sp21n11:/home/database/peter]
  Initial PROGRAM                             [/bin/csh]
  User INFORMATION                            [Group Database User]
```

Figure 71. Changing an SP User

The command used by SMIT is **spchuser**. It ensures, in this example, that the new home directory of this user will be moved from **sp21n13** to **sp21n11**, because this user has been moved from the Server group to the Database group. This command does not transfer the user′s files from one node to the other, however. This change will also affect the Initial Program, defined in the `/etc/passwd` file. Each change of user credentials must, of course, be passed to File-Collection Management, to NIS or to both.

13.1.4 Changing SP Users′ Passwords

When a new user is created, a default eight-character password is randomly generated and placed in its unencrypted form in `/usr/lpp/ssp/config/admin/newpass.log`. The SP administrator should communicate this password to the user. Alternatively, the administrator can set a different password using the **passwd** command and inform the user of the selected temporary password. In either case, the user must change the password the first time he logs in. This is reflected in the `/etc/security/passwd` file by the ADMCHG flag in the user′s stanza.

13.2 Access Control

This section describes how a systems administrator can implement user access control. Your SP may be configured in such a way that users are only allowed to log into certain nodes and are restricted from logging in on all others.

13.2.1 User Control

The AIX login attributes are stored in the /etc/security/user file. The login process determines from the following whether a user is allowed to log into that node:

- The **login** attribute
- The **rlogin** attribute

If these values are set to "true," the user can login or rlogin to that node. If set to "false," permission is refused.

There are two ways to maintain the access control files for the different groups of nodes:

1. Maintaining multiple /etc/security/user files, using the File-Collection Management tool for distribution.
2. Using the **spacs_cntrl** command. With this command, you can permit or deny user login access to a particular node or group of nodes.

The **spacs_cntrl** command must be executed on each node where access control for a user or group of users must be set. In our example, explained in more detail in 13.4, "SP File Collections" on page 206, we have defined three groups of nodes with exclusive login of the mentioned users. Other users, not belonging to a group, must be excluded from login to that group. This can be defined using the spacs_cntrl command as follows:

spacs_cntrl block <user-name>

or,

spacs_cntrl <file-name> **block**

where <file-name> contains a row of usernames. This file may also be included in File-Collections Management.

Allowing users to log into nodes or groups of nodes can be configured by using the **unblock** option of the spacs_cntrl command.

13.2.2 Job Control

Job control is particularly useful in the case of parallel jobs running in so-called User Space mode. In User Space mode, the parallel program interacts with other nodes using the Message Passing Libraries, communicating over the switch exclusively. Refer to 10.1, "The Resource Manager" on page 143 for an overview of the Resource Manager.

13.3 The Berkeley Automounter (Amd)

With NFS, remote filesystems can be mounted and accessed as if they were local. Although this is extremely useful, it does add some complexity to system administration. First, most likely, filesystems commonly mounted by NFS will be added to the `/etc/filesystems` configuration file on each host. This introduces more work for the system administrator as this file is not supported by NIS and must be manually maintained on each NFS client. Secondly, once a filesystem has been mounted, whether it is accessed or not, it remains mounted until manually unmounted. With many mounted filesystems, this increases the risk of hanging local processes should an NFS server crash. These are two of the problems that automounters attempt to solve.

An *automounter* is a tool that mounts filesystems on demand when they are first referenced and subsequently unmounts them after a period of inactivity. Automounters also add another level of transparency to the network. The binding of the local targets to physical location is not performed until the target is referenced. That is, unlike explicit mounts, clients do not have a static idea of where the targets are located. The physical location is resolved only at reference time.

The use of Amd to mount filesystems provides several clear advantages:

- Entries to `/etc/filesystems` are reduced in number, easing the burden on the system administrator.
- Amd automatically creates any necessary mount points.
- The use of selection criteria within the Amd configuration files allows the same files to be used throughout the network.
- The SP File-Collection technology or NIS can be used to distribute the Amd configuration files so that they may be kept in one central place.
- Since quiescent filesystems are automatically unmounted, the risk of NFS server crashes affecting the local host are diminished.
- For replicated read-only filesystems, Amd may choose from among a list of valid NFS servers, thereby removing a single-server dependency.

Use of the Berkeley Software Distribution automounter, Amd, on the SP is an alternative to Sun Microsystem′s automounter. Although either (or neither) may be used, Amd is the default. Amd is not public domain, but is freely available under license.

13.3.1 Operation

Amd operates by altering the way in which the kernel views the normal UNIX namespace of mounted filesystems. Mount points, called *automount points*, are added to this namespace by way of the `mount()` system call. For NFS mounts, this call takes among its arguments a host name and a socket address. The named host is the one providing, or exporting, the filesystem. The socket address is the server′s IP address and the well-known NFS port number 2049.

When Amd makes the mount system call, however, it specifies, not a foreign server and socket address, but a local process ID and the socket address on which that process is listening. That process is, of course, the Amd daemon itself.

To the kernel, then, all filesystems managed by Amd seem to be normal NFS-mounted filesystems. No modifications or additions to the kernel code are necessary. However, instead of passing NFS requests to a remote server, they are directed toward the local Amd process, which must interpret and respond to them (see Figure 72).

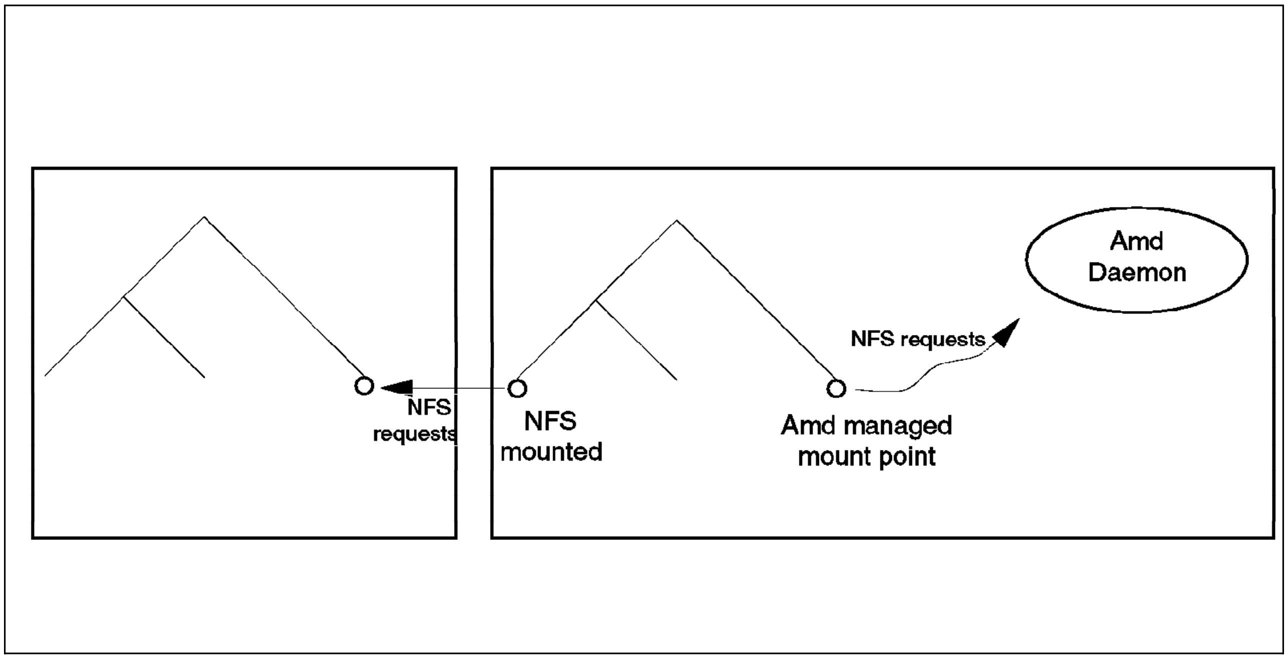


Figure 72. Amd-managed Mount Point

As Amd receives the NFS RPC requests, it first determines whether or not the requested volume is mounted. If not, the binding of the local target to the physical location takes place. That is, a runtime decision is made on how and from where to mount the filesystem. This decision is based on options specified on either the command line or in files called Amd mount maps.

When Amd must perform an NFS mount, it mounts the appropriate remote filesystem into a staging area, by default /a. Amd then responds to the kernel that a symbolic link exists at the local target pointing to the staging area. Two types of management are commonly used:

- **Indirect** - emulates a directory of symbolic links (illustrated in Figure 73 on page 196)
- **Direct** - emulates a single symbolic link (illustrated in Figure 74 on page 196)

Indirect management is useful when mounting several filesystems with a common pathname prefix, as seen on the client. For example, the home directory /u is often implemented with indirect management.

Direct management is used to define point-specific, non-uniform mount points. For example, /usr/man should be implemented as a direct map. Using indirect management for this directory would obscure the existing entries in /usr.

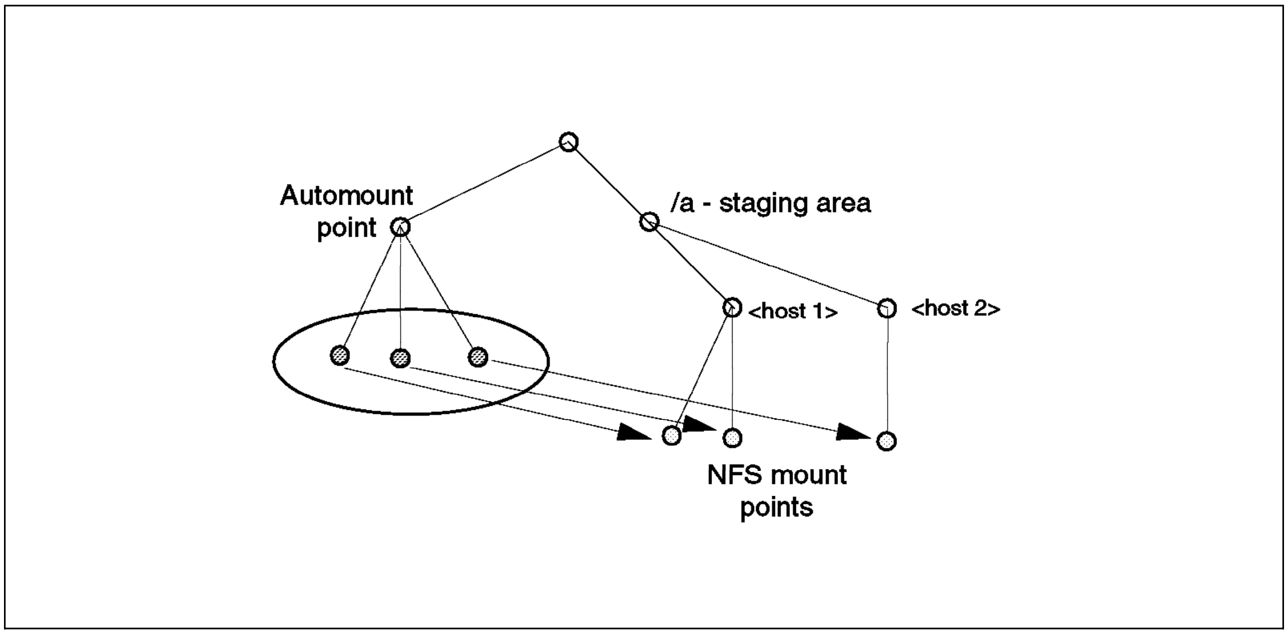


Figure 73. Indirect Filesystem Mount

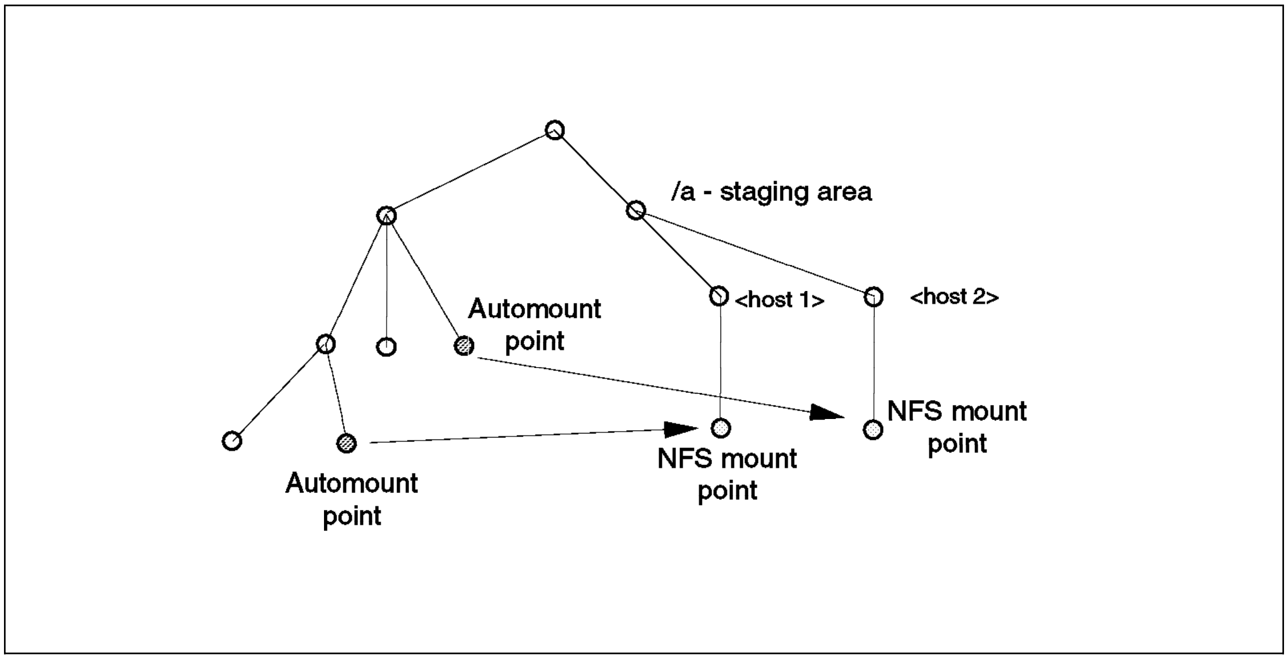


Figure 74. Direct Filesystem Mount

Each of the automount points has an associated Amd map. These maps contain key-value pairs, which describe the local targets and their physical locations. Any target may contain one or more locations and selection criteria to choose among them. For example, consider an Amd-managed mount point /myapp/bin. This directory contains executables that are specific to machine architecture. The Amd map can direct RS/6000 clients to mount from RS/6000 servers, SUN-4 clients to mount from SUN-4 servers and so on. Selection criteria within the map will resolve the appropriate physical location.

As another example, some read-only filesystems are replicated on several hosts, for example, man pages. Amd maps allow for multiple locations. If one host is unavailable, the next may serve the client.

A key feature of automounters is the unmounting of quiet filesystems. Besides avoiding a large and increasing number of mounts to the system, clients are isolated from server crashes in the sense that, if filesystems they "automounted" are unmounted at the time, no side effects can occur. Each mount that Amd performs has an associated time-to-live interval; the default is 120 seconds. After the interval has expired, the target-to-location binding is dropped and the filesystem is unmounted. If the filesystem is still active, the device will be busy and the unmount will fail. In this case, the target-location mapping is recalled and the time-to-live interval is reset and extended.

Amd takes special care to ensure that all of its operations do not block continued service. Since it is a single process responsible for handling all requests, Amd is built on top of a non-blocking RPC library. In worst case, on system calls that may block, Amd forks a process to handle the action.

The non-blocking library is also used to implement the notion of keep-alive intervals. For filesystems that are mounted, Amd attempts to keep track of their hosts′ status. Every certain interval, an NFS dummy request is sent to the servers. If there is no response to this ping, the server is considered down and any user request from that server would fail. The pings are continued so that, should the server become active again, the client can pass on the request. However, should the server remain down and the user make a request, if another location was specified, Amd will mount that server′s filesystem and pass the request to it.

13.3.2 Amd Mount Maps

In order for the Amd daemon to determine how to provide access to a filesystem, each automount point has a corresponding mount map. The maps provide the resolution to the true directory being referenced.

These maps can be established in a variety of manners using:

- Regular files
- NIS maps
- The password file

13.3.2.1 File Maps

File maps are the default on SP2 systems. The structure of a file map is illustrated in Figure 75.

```
#amd map
key1         location1 location2
key2         location1 location2 \
             location3
```

Figure 75. Amd Map Structure

Consider the Amd map in Figure 76 on page 198.

```
#Amd map - amd.u
/defaults       rfs:=/home;type:=nfs;sublink:=${key}

myuser         host==sp21sw03;type:=link;fs:=/home \
               host!=sp21sw03;rhost:=sp21sw03;rfs:=/home
```

Figure 76. Sample amd.u

/defaults is a special key whose location denotes default values for the rest of the map. This Amd map, amd.u, manages the filesystem /u. If a user on host sp21sw03 were to enter the command # cd /u/myuser, according to the map, Amd would perform a symbolic link to /home/myuser.

On the other hand, if the user is on another host, Amd mounts the remote filesystem referenced to a staging area (by default, this area is /a). Then, a symbolic link would join the referenced filesystem to the mounted filesystem. When the statement # cd /u/myuser is executed, Amd performs steps equivalent to:

```
# mount sp21sw03:/home/myuser /a/home/myuser
# ln -s /a/home/myuser /u/myuser
```

Amd, therefore, appears to the client simply as a symbolic link to a directory.

13.3.2.2 NIS Maps

If you are already using NIS, you can include Amd maps in the set of files maintained by NIS. Figure 77 on page 199 shows a portion of the NIS makefile, which handles the Amd map, amd.u. If you wish to use Amd maps serviced by NIS, you must manually edit the makefile to include stanzas similar to those in the referenced example.

13.3.2.3 Password Maps

Amd also allows you to use password files as mount maps. Any time the map name is /etc/passwd, Amd manipulates users' home directories into Amd maps.

A given home directory of the form /dir1/dir2/../dirN/username is parsed into the following Amd variables: ${rfs} is /dir1/dirN, ${rhost} is dirN. ... .dir2 and ${sublink} is username.

So, for example, a password file entry such as:

```
jim:|:201:0::/home/com/ibm/pok/itso/sp2n09/jim:/bin/ksh
```

would use an Amd map entry equivalent to:

```
rfs:=/home/sp2n09;rhost:=sp2n09.itso.pok.ibm.com;sublink:=jim
```

13.3.3 Map Formats

Given the previous examples of Amd maps, this section provides a more formal description of them.

Amd resolves binding of local targets to physical targets by first matching a key to the local target then selecting a physical target from an associated location list. Generally, the key is some component of the local target pathname. Variable expansion or prefixing may alter this, however. See Figure 78 on page 199 for the format of an Amd map.

```
# NIS Makefile

DIR=/etc
DOM=domainname
NOPUSH=""
YPDIR=/usr/etc/yp
YPDBDIR=/var/yp
YPPUSH=$(YPDIR)/yppush
MAKEDBM=$(YPDIR)/makedbm
:
NISAMDDIR=/etc/amd/amdnis
amd.u.time: $(AMDNISDIR)/amd.u
    -@sed -e "/#.*$$//" -e "/¬$$/d" $(AMDNISDIR)/amd.u |
    awk '{ \
          for (i=1;i<=NF;i++) \
                if (i==NF) { \
                      if (substr($$i,length($$i),1)=="\\") \
                            printf("%s",substr($$i,1,length($$i)-1)); \
                      else \
                            printf("%s\n",$$i); \
                } \
                else \
                      printf("%s ",$$i); \
          }' | \
    $(MAKEDBM) - $(YPDBDIR)/amd.u
    touch $(YPTSDIR)/amd.u.time
    echo "updated amd.u"; \
    if [ | $(NOPUSH) ]; then \
          $(YPPUSH) amd.u
          echo "pushed amd.u"; \
    else \
          echo "couldn't find $(AMDNISDIR)/amd.u"; \
    fi
```

Figure 77. NIS Stanza for amd.u

```
key1    location-list1
key2    location-list2
```

Figure 78. Map Format

A BNF-like syntax for the location list is shown in Figure 79 on page 200.

The location-selection specifies a list of possible targets. If separated by the "||" operator, the first location-selection selected will be the only one attempted, regardless of whether the mount fails or succeeds.

Specific locations are chosen by the selection component. Predefined built-in selectors must be used here (see 13.3.3.2, "Selectors" on page 201).

Once a location has been selected, options specify information about the file system so that it may be mounted (see 13.3.3.3, "Options" on page 201).

The "-" prefix denotes defaults to be used for all subsequent locations. When encountered, all previous defaults, including those specified on the `/defaults` line, are discarded.

```
location-list:
        location-selection
        location-list white-space || white-space location-selection
location-selection:
        location
        location-selection white-space location
location:
        location-info
        -location-info
        -
location-info:
        selection-or-option
        location-info;selection-or-option
        ;
selection-or-option:
        selection
        option-assignment
selection:
        selector==value
        selector!=value
option-assignment:
        option:=value
white-space:
        space
        tab
```

Figure 79. Location Format

As an example, referring back to Figure 76 on page 198:

- `myuser` is the key.
- `host==sp21sw03;type:=link;fs:=/home host!=sp21sw03;rhost:=sp21sw03;rfs:=/home` is a location-selection.
- `host==sp21sw03;type:=link;fs:=/home` is a location.
- `host==sp21sw03` is a selection.
- `type:=link` is an option-assignment.

13.3.3.1 Variable Substitution

Amd allows for the use of variables within the mount maps. Environment variables as well as predefined variables of the form $var can be used.

Operators can be specified to manipulate pathnames and domain names. The "/," when used as a prefix, as in `${/path}`, evaluates to the basename component. For example, if `${path}` were set to `/home/sp21sw03/clive`, `${/path}` would evaluate to `clive`. On the other hand, as a suffix, "/" returns the pathname component. In this case, `${path/}` would evaluate to `/home/sp21sw03`.

The "." operator acts in a similar fashion with dotted domain names. If `${rhost}` were set to `sp21cw0.itsc.pok.ibm.com`, `${rhost.}` would evaluate to `sp21cw0` and `${.rhost}` to `itsc.pok.ibm.com`.

13.3.3.2 Selectors

Selectors in Amd maps are used to select the target to mount. Selectors are evaluated left to right and use operators == and != for equality and non-equality, respectively.

The following selectors are static. Once Amd has been started on a particular host, they will evaluate to the same result on each lookup:

arch	The machine architecture for which the running version of Amd was compiled. The command **/etc/amd/amd -v** displays this value. For the SP2 nodes, this value is `ibm6000`.
autodir	The staging area, by default `/a`. The command line option **-a** can be used to explicitly set this value.
byte	The hardware byte ordering.
cluster	The name of the current cluster. This value defaults to the value of `${domain}`, but can be explicitly set with command line option **-C**.
domain	The domain name. This value is taken from the host's hostname, but can be explicitly set with the **-d** command line option.
host	The host name. This value is taken from the leading portion of the host's hostname.
hostd	The fully-qualified domain name formed by `${host}.${domain}`.
karch	A hook for the kernel architecture. The default value is taken from `${arch}`, but can be explicitly set with the **-k** command-line option.
os	The operating system for which the running version of Amd was compiled. The command line option **-v** displays this value.

In most standard configurations involving the SP2, the above selectors will not be used.

Conversely, the following selectors may be used quite frequently. The value of these selectors is dynamic and may vary for each map lookup:

key	The name to be resolved. For example, if access is made to `/home/bill` and `/home` is an automount point, `${key}` is set to `bill`.
map	The mount map name.
path	The entire pathname of the name being resolved.
wire	The primary network interface.

13.3.3.3 Options

Map options are used to specify more options with which to perform the mount of a particular target. They are specified in the form `option:=value`.

delay	The number of seconds to wait before attempting to mount. This option is typically used in situations where you would want to mount a primary server first.
fs	Denotes the local mount point. Default: ${autodir}/${rhost}${rfs}.

opts	Used to specify options to the mount system call. Default: "rw,defaults."
remopts	Specifies mount options for remote hosts that exist across a gateway.
sublink	Specifies the subdirectory of the mounted filesystem which is the actual reference. This option is used to reduce the number of mounts.
type	Denotes the type of filesystem to be used. For example, possible values include: link, nfs, auto and so on.

Refer to the man pages for a complete description of all Amd options.

Note: The man pages are bundled with the source in /usr/lpp/ssp/public/amd920824upl75.tar.Z. They can be extracted with the commands:

```
zcat amd920824upl75.tar.Z |tar -xvf - amd920824upl75/amd/amd.8
zcat amd920824upl75.tar.Z |tar -xvf - amd920824upl75/amq/amq.8
```

They must then be placed in the man pages directory /usr/man/man8. They are also bundled in the Amd reference amd920824upl75/doc/amdref.texinfo (formatted in TeX).

13.3.4 Runtime Administration

Amd runtime administration is controlled by the program **amq** located in /etc/amd. In addition, amq can return state information about various facets of Amd processing.

By default, with no arguments, amq returns information regarding each automount point′s filesystem type, mount map and mount point (see Figure 80).

```
/          root    ″root″      sp21n10:(pid15603)
/u/jim     direct  amd.jim     /u/jim
```

Figure 80. Sample amq Output

Consult the man pages for full details.

13.3.5 Amd on the SP

On the SP, Amd can be enabled or disabled through the SMIT panels or with the **spsitenv** command. This can be done at installation time or any time thereafter. Any changes will take effect on each node after it is rebooted.

To see if Amd is currently configured, enter:

```
# splstdata -e | grep AMD_CONFIG
```

13.3.5.1 Starting Amd

When configured, Amd starts automatically upon system reboot. The perl script /usr/lpp/ssp/install/bin/services_config, which is called from /etc/rc.sp, determines that Amd is configured and calls /usr/lpp/ssp/install/bin/amd_config. This perl script invokes /etc/amd/amd_start, which starts Amd.

Defaults: The perl script **amd_start** starts Amd with most command line defaults:

```
system("nice --4 $amd -t 16.120 -x all -l $amdlog -p > /etc/amd/amd.pid @amdargs
") && exit(1);
```

The parameters used above do the following:

- `-t` sets the RPC timeout and retransmit intervals corresponding to the `timeo` and `retrans` mount options.
- `-x` specifies verbose logging.
- `-l` specifies the log file.
- `-p` echoes the Amd process ID.

The list variable @amdargs contains a list of automount points and the corresponding mount map. By default, the script looks in the `/etc/amd/amd-maps` directory for any file of the form `amd.*`. These files are taken to mount maps and their extensions as the corresponding automount points. This makes it difficult to automatically handle an automount point such as `/usr/man`, since you could not create a file `amd.usr/man`. For cases such as this, you must edit `amd_start` and place an entry like the following in the line that starts Amd:

```
system("nice --4 $amd -t 16.120 -x all -l $amdlog -p > /etc/amd/amd.pid @amdargs
 /usr/man /etc/amd/amd-maps2/amd.man -type:=direct") && exit(1);
```

Of course, the mount map should exist in a directory other than `/etc/amd/amd-maps`.

If modified, the `amd_start` script must also be copied to the nodes. This procedure is described in section 13.3.5.5, "Distributing the Start Script" on page 204.

13.3.5.2 Stopping Amd

Amd should be stopped using either the SIGTERM or SIGINT signal.

SIGTERM (15)	Stops Amd and unmounts top-level automount points. Filesystems currently automounted remain NFS-mounted.
SIGINT (2)	Similar to SIGTERM, but also attempts to unmount any automounted filesystems.

Examining the amd_start script, we see that the Amd process ID is captured in `/etc/amd/amd.pid`. So, we can use the following script to stop Amd:

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

if (system("cat /etc/amd/amd.pid | xargs kill -SIGINT"))
{
     print "\nAmd not stopped !\n\n";
}
else
{
     print "\nAmd stopped and filesystems unmounted.\n\n";
}
```

13.3.5.3 Refreshing Amd

If existing maps have been changed, you can force Amd to reread the maps with the script `/etc/amd/refresh_amd`.

13.3.5.4 Logging Errors

By default, all Amd log messages on a node are written to /var/adm/SPlogs/amd/amd.log. If you wish to change this, you must edit the amd_start script.

13.3.5.5 Distributing the Start Script

Whenever changes are made to the /etc/amd/amd_start, the script must be copied to the SP nodes. Note that you should be modifying this script on the Control Workstation and distributing from there. This is simply the easiest method. To do so:

- If you are using File Collections:
  1. Put the file in the node.root secondary collection to be served by the power_system File Collection using:

     ```
     # cp /etc/amd/amd_start /share/power/system/3.2/etc/amd/amd_start
     ```

  2. Update any boot/install servers by running the following command on each boot/install server:

     ```
     # supper update
     ```

  3. Update the nodes by running the following command on each node:

     ```
     # supper update user.admin
     ```

  4. Stop and restart Amd on each node.
- If you are not using File Collections:
  1. Copy the file to all nodes with the following:

     ```
     # dsh -w /tmp/sp2.allnodes ″rcp sp2cw:/etc/amd/amd_start /etc/amd/.″
     ```

     Note that /tmp/sp2.allnodes is a working-collective file containing all nodes and sp2cw is the Control Workstation.
  2. Stop and restart Amd on each node.

13.3.6 Examples

Following are several examples to further illustrate the functions of Amd.

1. What is the difference between the two maps in Figure 81 and Figure 82 on page 205?

```
#Amd map - amd.u
/defaults       rfs:=/home;type:=nfs;sublink:=${key}

userA         host==sp21sw01;type:=link;fs:=/home \
              host!=sp21sw01;rhost:=sp21sw01;rfs:=/home
userB         host==sp21sw01;type:=link;fs:=/home \
              host!=sp21sw01;rhost:=sp21sw01;rfs:=/home
```

Figure 81. Sample amd.u With sublink Option

```
#Amd map - amd.u
/defaults       rfs:=/home;type:=nfs

userA         host==sp21sw01;type:=link;fs:=/home/userA \
              host!=sp21sw01;rhost:=sp21sw01;rfs:=/home/userA
userB         host==sp21sw01;type:=link;fs:=/home/userB \
              host!=sp21sw01;rhost:=sp21sw01;rfs:=/home/userB
```

Figure 82. Sample amd.u

The map in Figure 81 on page 204 uses the **sublink** option. On any host other than sp21sw01, the remote filesystem /home from host sp21sw01 would be mounted if a user entered:

```
# cd /u/userA
# cd /u/userB
```

Only one mount would be performed. The sublink specifies the subdirectory within the mounted filesystem to which the reference should point.

The map in Figure 82, on the other hand, would require two physical mounts from host sp21sw01, /home/userA and /home/userB.

Hence, sublink can be used to reduce the number of physical mounts.

2. How can you automount man pages in /usr/man without obscuring other directories in /usr?

   The *direct* filesystem type appears as a symbolic link to a mounted filesystem. Consider the map in Figure 83.

```
#Amd map - amd.man

usr/man       type:=nfs;rfs:=/usr/man;rhost=sp21cw0
```

Figure 83. Direct Mount Map

   The *direct* filesystem type must be specified on the command line; for example:

```
# amd /usr/man amd.man -type:=direct
```

3. The man pages are replicated on several servers.

   By specifying the list of hosts in the map, Amd can handle a replicated filesystem. When Amd attempts to mount from the first server, if it is busy and does not respond in time, Amd will try the next. To set this up, alter the **amd.man** as in Figure 84.

```
#Amd map - amd.man

usr/man       -type:=nfs;rfs:=/usr/man
              rhost=sp21cw0 rhost=sp21sw07
```

Figure 84. Map for Replicated Man Pages

4. How can you see all existing mounts created by Amd?

   Amd run-time management is performed with the **Amq** command. With no arguments, Amq produces output similar to that shown in Figure 85.

```
/        root    ″root″           risc0039:(pid17707)
/u       toplvl  amd.u            /u
/u/bob   nfs     risc0041:/home   /a/risc0041/home/bob
/u/jim   link    ./home/jim       /home/jim
```

Figure 85. Existing Amd Mounts

   This output says that `amd.u` manages the `/u` automount point. Additionally, automount points `/u/bob` and `/u/jim` are currently active.

13.4 SP File Collections

This section of the book looks at File Collections on the SP. File Collections (an integral part of the SP system-management tools) are a very powerful mechanism for easily maintaining copies of important files across an SP system.

The following topics are discussed:

- Overview of SP File Collections
- Using and Managing SP File Collections
- Configuring File Collections for our scenario
- Experiences with File Collections

13.4.1 Overview of SP File Collections

A File Collection is very simply a set of defined files that is managed and controlled by the *supper* program. It can be any set of files or directories that exist on the Control Workstation. The fact that these files have been defined to be in a File Collection, means that duplicate copies on other nodes can automatically be maintained in an easy and efficient manner. File Collections are "switched on" by default when you install an SP.

File Collections are most useful when you want particular files replicated transparently across some or all of your SP nodes. They are used for this purpose by a number of the SP systems-management tools. For example, by default, the SP uses File Collections for maintaining key user-administration and configuration files. You can tailor the way in which the system File Collections work and very easily add your own user-defined ones.

A File Collection is defined in a special File-Collection directory. Within that directory are stored a number of details specific to that particular File Collection. You do not store all of the files that you wish to distribute within the File-Collection directory; the files can be anywhere on the system. The File-Collection directory lists the files and directories, and describes any rule to define them. There is only one master copy of the files.

We set up and manage File Collections with specific File-Collection or supper commands. These commands are discussed when we look at how to implement File Collections.

The master files are stored in one place only. Very often this is the Control Workstation, but it does not have to be. A File Collection is said to be “resident” on a particular system. In other words, this is the master location for the files; any updates to the files in this location are replicated to other nodes. Updates to files in other locations are, of course, overwritten by the supper process at predefined intervals. It would be wrong to update files anywhere other than where they are resident.

File Collections can be *primary* or *secondary*. A primary File Collection can contain a secondary file collection. Consider a case where you have created a File Collection called `myprimefc` and, within that, you defined a secondary File Collection called `mysecondfc`. You could install the `myprimefc` collection onto a node and all of the files that have been defined within that File Collection will be installed on that node. The files which comprise `mysecondfc` File Collection would also be installed on the node. However, they would not be available on that node. They could, however, be served to another node from there. This avoids having to install the files in their “real” or “final” location.

So, secondary File Collections are useful where you need a second tier or level of distributing file collections. This is particularly helpful when using Boot/Install servers within your SP or when partitioning the system into groups.

There are four File Collections that are delivered and automatically installed on an SP. These standard File Collections are:

sup.admin	This is a primary file collection that resides on the Control Workstation. It contains a number of files that define all of the other File Collections, as well as the programs that are used to load and manage the File Collections.
user.admin	This is also a primary File Collection. It contains a number of important files related to user management. Files that describe the valid user IDs on the system, passwords, home directory and Amd information are stored here. When you create an SP user ID using the SMIT panels, it is this File Collection that ensures that this information is replicated across all nodes if required.
power_system	This File Collection is used for files that are system- or node-dependent. By default it only contains a secondary file collection, `node.root`.
node.root	This is a secondary File Collection containing key files that are node-specific. An example of a file that could be stored here is the `.profile` file that describes a user′s profile; it may be node-dependent.

Again, by default, the Control Workstation is the master server for all of the default file collections. That is, the master copy of all files originates from the Control Workstation.

If any Boot/Install servers are set up to do SP node installation, they will also be File-Collection servers.

Let us consider some examples.

If you have installed all of your SP nodes directly from the Control Workstation (in other words, you have not used any Boot/Install servers), then all File Collections will be automatically updated from the Control Workstation directly to all of the nodes.

Alternatively, if you set up node 1 as a Boot/Install server for all other nodes, then, at install time, the other nodes will install their node images from node 1. Additionally, all File Collections will also be served by node 1 to all other nodes. So, the Control Workstation would update node 1 only, while node 1 would be responsible for updating all other nodes.

This structure or hierarchy for File Collections is the default and can be changed, if required.

As you are probably aware, File Collections are not the only way in which to maintain duplicate files on multiple nodes. For example, Network File System (NFS) allows you to share files across a TCP/IP network and could provide a similar kind of function. For user-management files, Network Information System (NIS) can also perform the task of keeping these files up to date.

So when should you use File Collections as opposed to other mechanisms?

File Collections are updated at predefined intervals. If you access one of the files in the interim period before it has been updated, you see the original one. So, File Collections would not be used where second-by-second access to updated files is needed. They are, however, very suitable for files that change reasonably often, but where it is not critical that you see up-to-date information. An example might be a company telephone directory. One master copy can be maintained and File Collections used to update the duplicate copies every four hours, let us say. Between times, if a user accesses one of the copies, he or she sees the original directory. But, that should not be a problem in this case.

If you must have absolutely up-to-date information at all times, you would not use File Collections. Another mechanism, such as NFS, would be better. Remember, however, that NFS puts additional CPU and network load onto the system and every request to read an NFS-mounted file results in I/O across the network.

In the case of File Collections, you have the small overhead of an update from time to time, but all subsequent file accesses are local, not across the network.

13.4.2 Using and Managing SP File Collections

Let us begin by looking at the existing File Collections delivered with the SP system.

13.4.2.1 System-Delivered File Collections

At install time, the filecoll_config parameter in the SP SMIT panel that allows us to set Site Environment Information is set to “true” (indicating that File Collections are to be used). This could be switched to false, if required, but it is not recommended. It is possible to tailor or add to the default File Collections, but, since many system-management functions make use of them, do not “remove” them unless you are sure of the consequences.

The /var/sysman/sup directory contains the definitions for the file collections. Directories exist for each of the default File Collections delivered with the system:

- node.root
- power_system
- sup.admin
- user.admin

If you look in the /var/sysman/sup/user.admin directory, for example, you see the typical set of control files provided with any File Collection:

last	This file is a system-maintained list of the files and directories that have been updated.
list	This file contains details about the files that are actually part of this file collection. Rather than merely list them, however, it may be easier to define rules for identifying the files. For example, you could define a collection of all of the files within a directory or all files on the system that have .myfile at the end of the name. The supper program scans the filesystems as specified to find these files.
lock	This is an empty file used by supper to lock a file collection at update time, preventing more than one update of a collection at the same time.
prefix	This is the starting point that supper uses when searching or scanning for files. If the root (/) directory is specified, all files on the system will be scanned.
refuse	In this file, you can specify particular files that must not be updated by the supper process. In some cases, it might be easier to define your files for update by using wildcards, then use the refuse file to exclude particular files from that list.
scan	This file contains a list of all of the files in this File Collection. If it exists, it is used by the supper process as the definitive list of all files. If it does not exist, supper does a scan to determine which files should be included. It is normally recommended that you perform a scan to create this file at the time that the File Collection is created or modified. This means that a scan need not be performed each time an update takes place thereafter. If a scan has been performed and the scan file exists, you must ensure that it is up-to-date. If it is there, it will be used even if the File Collection has been modified and additional files have been added to the File Collection in the list file. Using the scan file can minimize the system overhead when performing supper updates.
supperlock	This file is similar to the lock file and is used by supper to lock a File Collection during update.
when	This contains the time of the last File Collection update. It protects against accidental update with older versions.

This is a full list of the control files that can be used to define and control File Collections All of these files may not necessarily be found in the directory for every File Collection. The most important file is the `list` file; it really tells you the most about a File Collection.

Apart from the directories for each of the File Collections, the `/var/sysman/sup` directory also contains a number of other files:

.active	This describes the active volume group.
.resident	This lists all File Collections. If you add your own File Collection, it must be added here.
lists	This is a directory that contains links to each of the `list` files in each of the individual File-Collection directories.
refuse	This file is similar to the `refuse` files in each of the File-Collection directories. Any files listed here are excluded from all File Collections.
supfilesrv.pid	This contains the process number or ID of the supfilesrv process.

13.4.2.2 Reporting on the Status of File Collections

The supper command has a number of subcommands that allow you to do a wide variety of things. Let us look first at the supper subcommands that allow you to report information about existing file collections.

Note: The supper command can be found in the `/var/sysman` directory.

The various subcommands that can be used with supper to report information about File Collections are:

status	Reports the name, resident status and access point for all File Collections. It also reports the name and size of associated file systems.
files	Lists the resident files resulting from an update or install of a File Collection.
serve	Lists the File Collections that can be served from this machine.
where	Lists the current boot/install server (and, therefore, File Collection server) for File Collections on this machine.
when	Shows the last update time for all resident collections.
diskinfo	Shows the available disk space and active volumes for this machine.
log	Summary of the current or most-recent supper session.
rlog	Raw output of the current or most-recent supper session.

Depending on your File Collections setup, it makes sense to run some of these subcommands on a Control Workstation, some on a Boot/Install server and others on the SP nodes themselves. There is no harm done if you run these anywhere; they just report information. But, you must understand your File-Collections configuration before you can fully interpret the information that these commands return to you.

The **where** subcommand is particularly useful in helping you to understand the File Collections hierarchy on any SP.

You could, for example, run the following command on a node:

```
supper where user.admin
```

It returns the hostname of the File-Collection server for this particular File Collection. (If no special change was made to the installation process, the previous command returns the default server for the File Collection, which is the Control Workstation, of course).

Full details of the supper command and it various subcommands can be found in *IBM 9076 Scalable POWERparallel Systems SP2 Administration Guide*.

13.4.2.3 Updating an Existing File Collection

Imagine that you have changed one of the files that is in an existing File Collection. How does the updated file get propagated to the other nodes in the SP?

Let us consider an example in which you are using an SP with 12 nodes. Node 1 is an install server for the other 11 nodes. Therefore, node 1 is also the File-Collections server for the other 11 nodes. The Control Workstation is responsible for updating node 1 and node 1 is responsible, in turn, for updating the other nodes.

Figure 86 on page 212 shows this File-Collection hierarchy.

If you add an SP user on the Control Workstation using the SMIT panel, an entry for this new user is added to the file /etc/passwd on the Control Workstation. Other files are also updated, but let us concentrate on the /etc/passwd file as this defines the user′s ID, without which a user cannot log into a node.

Having added this user ID on the Control Workstation, the /etc/passwd file is immediately updated. This new user can log into the Control Workstation and get access to the system straight away. However, out on all of the nodes, the "old" /etc/passwd file is still in existence and this new user ID is not yet listed. If the user should try to log into one of these nodes at this stage, he or she would fail to gain access.

You need to propagate the updated /etc/passwd file to all of the nodes.

In fact, the user.admin File Collection, which contains the /etc/passwd file and other default File Collections, is automatically updated every hour by the *cron* daemon. The *cron* entry in the file /var/spool/cron/crontabs/root file was created automatically at install time. The default update time of one hour can, of course, be changed if required.

Let us assume, however, that you wish to force the updates on the nodes to happen immediately.

You would use the supper command with the **update** subcommand to carry out these updates.

The **supper update** command is a "pull" rather than a "push" command. This means you must run the update command on the nodes to "suck" the updated files in your File Collection from the Control Workstation onto the nodes.

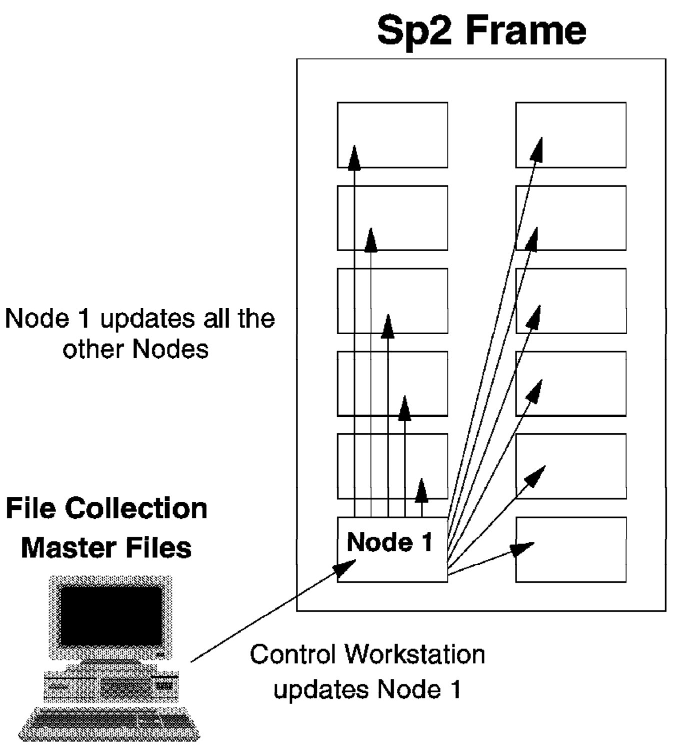

Sp2 Frame
Node 1 updates all the other Nodes
File Collection Master Files
Node 1
Control Workstation updates Node 1

Figure 86. A Typical File-Collection Hierarchy

It would, of course, be simpler to use the **dsh** command to run this command on all nodes simultaneously. If run on the command line of the Control Workstation, the following command performs an update on each node:

```
dsh -a /var/sysman/supper update user.admin
```

However, consider your File-Collection hierarchy carefully before running this command. In our example, there is one thing that you must look at carefully. The hierarchy has node 1 as a Boot/Install server and, therefore, also a File-Collections server. When nodes 2 through 12 are updated, they get their updated information from node 1. However, at this time, node 1 has the old information; it has not yet been updated!

The first time that you run the dsh command, node 1 gets updated correctly, but the other nodes get an old copy from node 1. If you run it again, the remaining nodes will get the latest copy from node 1.

There are two alternatives here; you could either:

1. Run a **dsh** of the update command on all of the nodes twice

   -or-

2. Selectively update node 1 and update the remaining nodes afterwards

The **supper update** command can be used in this way with any File Collection if files have been changed or edited within the File Collection.

In some cases, it is appropriate for an administrator to run the update commands for File Collections that he or she has changed immediately after editing the files. These updates can easily be run as scripts, of course. In other cases, it is appropriate to use cron for updating File Collections at specified times of the day.

13.4.2.4 Creating a New File Collection

We now look at adding your own File Collection containing application or user files of some kind.

In our example, you wish to create a File Collection that is a primary collection. The files will reside on the Control Workstation and you want to propagate them to all nodes.

The files that you wish to propagate are all located in the directory /dogs. If you look in that directory, you see a number of files:

- westies
- labrador_retrievers
- german_shepherds
- dalmations
- border_collies
- as400
- siamese
- do_it
- my_script

Each of these is a normal AIX file. Most are text files that are updated from time to time. One of the files, my_script, is an executable script file. Any updates to these files must be reflected across the system.

In addition, you have a few other requirements. You do not want to replicate the file siamese as this is not a dog; it must be excluded. You also want to allow for any other files added into this directory. These additional files should also be replicated, unless otherwise specified.

Finally, your last requirement is that when the file called do_it is updated, the shell script called my_script should be executed. This script runs the **date** command, putting the output in a file in the same directory. It could do anything. We use it as a simple example to get you going with File Collections, one that will be easy to test.

To build a File Collection with these characteristics, you must follow a number of simple steps, which we describe one by one below.

1. Build the File Collection:

   a. Identify the Files that we want to Collect:

      The files you want to collect and distribute are all of those in the directory /dogs on the Control Workstation, except for the file siamese which is to be excluded.

b. Create a File-Collection Directory:

This directory should be made under the var/sysman/sup directory. If you give it the name my_file_collection, its full path name would be: /var/sysman/sup/my_file_collection.

Note that this directory will not contain the actual files that you wish to replicate. These are all in /dogs. It will only contain the master files that describe this File Collection.

c. Create the Master Files:

In the directory that you have just created, you can create all of the master files for this File Collection. The easiest way to do this is to copy an existing set from a previously-created File Collection.

To copy across those from the sup.admin File Collection, use the command:

```
cp /var/sysman/sup/sup.admin/*
/var/sysman/sup/my_file_collection
```

You are now ready to modify these File-Collection control files.

Modify the list file first, adding in what you want (after having removed what was in the one that you copied across, of course). It should look like the file in Figure 87 when you have finished.

```
symlinkall
omitany *~
upgrade ./dogs
omit ./dogs/siamese
always ./dogs/westies
execute /dogs/my_script (./dogs/do_it)
```

Figure 87. Sample File-Collection list File

This contents of this file say:

- Where they exist, copy links rather than resolving them.
- Omit files that start with the backup notation.
- Upgrade all files in the directory /dogs.
- Omit or exclude the file /dogs/siamese.
- Override any omit instruction (if there is one) and always update the file /dogs/westies.
- Execute the script /dogs/my_script if the file /dogs/do_it has been changed.

This should meet all of your requirements. The other files that were copied across should be fine, but you may wish to check them to see that they look okay and to understand what they do.

d. Add a link to the lists file:

The lists file in the /var/sysman/sup directory contains a symbolic link to the list file in each File-Collection directory.

Add the following link with this command:

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

e. Update the file.collections file:

   Edit the /var/sysman/sup/file.collections file and add the following line:

   ```
   primary my_file_collection - / - / EDO power no
   ```

   Going through this command field by field:

   - This is a primary file collection.
   - Its name is my_file_collection.
   - No filesystem is associated with this File Collection. If it is a primary file collection, a file system can be created if required.
   - This is the name of the directory under which the files are normally accessed (root (/) in this case). It is possible for a file on the server to have a different path name on the SP node when it is installed.
   - A file-system size is not specified (−).
   - The starting point for the directory search when looking for ./dogs is the root (/) directory. You could ask supper to start searching somewhere else if wanted.
   - The SUP options E, D and O indicate:
     – E means the list file can have execute statements.
     – D deletes files that are no longer in the collection.
     – O means that all files (even those older than the last update) should be evaluated.
   - **power** specifies that the system architecture is RISC System/6000.
   - This File Collection cannot be installed on a different architecture.

f. Update the .resident file:

   This file contains a list of all File Collections. Add yours to the list.

g. Build the scan file if required:

   This is optional. As discussed earlier, if the files are scanned to create a scan file, update time will be faster. However, if you do create this file, you are responsible for keeping it up to date by scanning the File Collection each time you add or delete files.

   If you wish to run the **scan** command, enter:

   ```
   /var/sysman/supper scan my_file_collection
   ```

   A scan file will be created.

2. Install the File Collection

   a. Update the sup.admin File Collection:

      This File Collection contains information about other File Collections on the SP.

      This File Collection must be updated with the information about your newly-created File Collection. For example, the sup.admin File Collection contains the file.collections and .resident files. These must be propagated to all nodes to tell them about the new File Collection. To do this, run the following command twice:

      ```
      dsh -a sup.admin
      ```

      This is the lazy way of doing things. You could update node 1 first, if preferred, then update the other nodes.

   b. Run the **supper install** command:

      The **supper install** command must be run on each node where you want to install the file collection.

For the file collection in our example, run the command:

```
/var/sysman/supper install my_file_collection
```

You could use **dsh** from the Control Workstation to run this on all nodes. If you did not want this File Collection installed on any particular node, simply do not run this command on that particular node.

c. Add the **supper update** command to the `crontabs` file:

If you want this File Collection to be updated automatically at predefined intervals, put an entry in the `crontab` file for root.

For example, to update the File Collection every hour at 15 minutes past the hour, you would proceed as shown in the following:

1) `crontab -l > /tmp/cron`
2) Edit `/tmp/cron` to add:

```
15 * * * * /var/sysman/supper update my_file_collection
```

3) `crontab -r /tmp/cron`
4) `rm /tmp/cron`

This should be stored in the `crontabs` file on each node, not on the Control Workstation.

(If you prefer to run the update command “manually,” you clearly would not have to put a cron entry in this file).

3. Test the File Collection

To check that the new File Collection functions as designed, you can test it in a number of ways. (In each case, after making the changes indicated, you must run the **supper update** command on the boot/install servers to propagate the updated files. Check for updates occurring on the nodes of your SP):

- Were all of the files except `siamese` copied?
- Add a file to the `/dogs` directory, and run the **supper update** command on all nodes. Does it get copied across?
- If you edit or change the `do_it` file and update the File Collections, does the script `my_script` get run?
- If you remove a file and update the File Collections, does the file get removed from the nodes?

If each of these tests works, everything is looking good!

13.4.2.5 Modifying the File-Collection Hierarchy

The delivered default File Collections have a hierarchy as shown in Figure 86 on page 212.

You can modify this hierarchy in several ways, if you wish.

For example, if you want a Boot/Install server to act as the master server for certain File Collections, you could sever the link from the Control Workstation to that a Boot/Install server from a File-Collections point of view. This Boot/Install server can then serve any group of nodes that you wish to define.

This breaking of the File-Collections link is referred to as taking a File Collection “offline.”

This can be achieved with the **supper offline** command.

If you decide to change the hierarchy in this way, you must make careful note of how updates will occur. It is potentially possible to have two "master" files for certain File Collections. If you do this deliberately because you have chosen to do so, that is fine. However, it is possible to cause confusion if this process is not carefully managed.

13.4.3 Configuring File Collections for Our Scenario

In our case, as discussed in Chapter 1, "System-Management Project Environment" on page 3, we have divided our SP nodes into groups. Since we have somewhat different requirements within each group, we want to use File Collections to help us manage this process.

In real life, of course, most users will not have such strange scenarios as those we outline here (such as using both NIS and File Collections at the same time). We set up these kinds of environments to demonstrate the flexibility of the tools to cope with a wide range of situations.

To refresh your memory, Figure 88 on page 218 shows the partitioning of our system and how we have grouped the nodes.

We will look at each node group in turn, studying the changes that we must make to the File Collections to achieve our aims.

13.4.3.1 Node-Group S

Group S is the Server set of nodes. Some changes must be made to our File-Collection setup to support our environment.

Figure 89 on page 219 shows how Group S will be configured.

There are users (bill, peter and jim) in this group of nodes who are not allowed to login anywhere else on the system. They will be barred from other nodes.

We have decided to use NIS for user and password management of these nodes.

These are the changes that we must make to the standard File Collections to support these objectives:

1. Since we have chosen to use NIS, we must remove the following files from the `user.admin` File Collection:

   - /etc/passwd
   - /etc/group
   - /etc/security/group
   - /etc/security/passwd

   This can be done either by leaving the files out of the File Collection `list` file or by adding them into the `refuse` file. This prevents these files from being updated on all nodes.

2. We will let NIS manage the nodes, but we will also add these files back into an additional File Collection for those nodes where we want user and password information updated by File Collections.

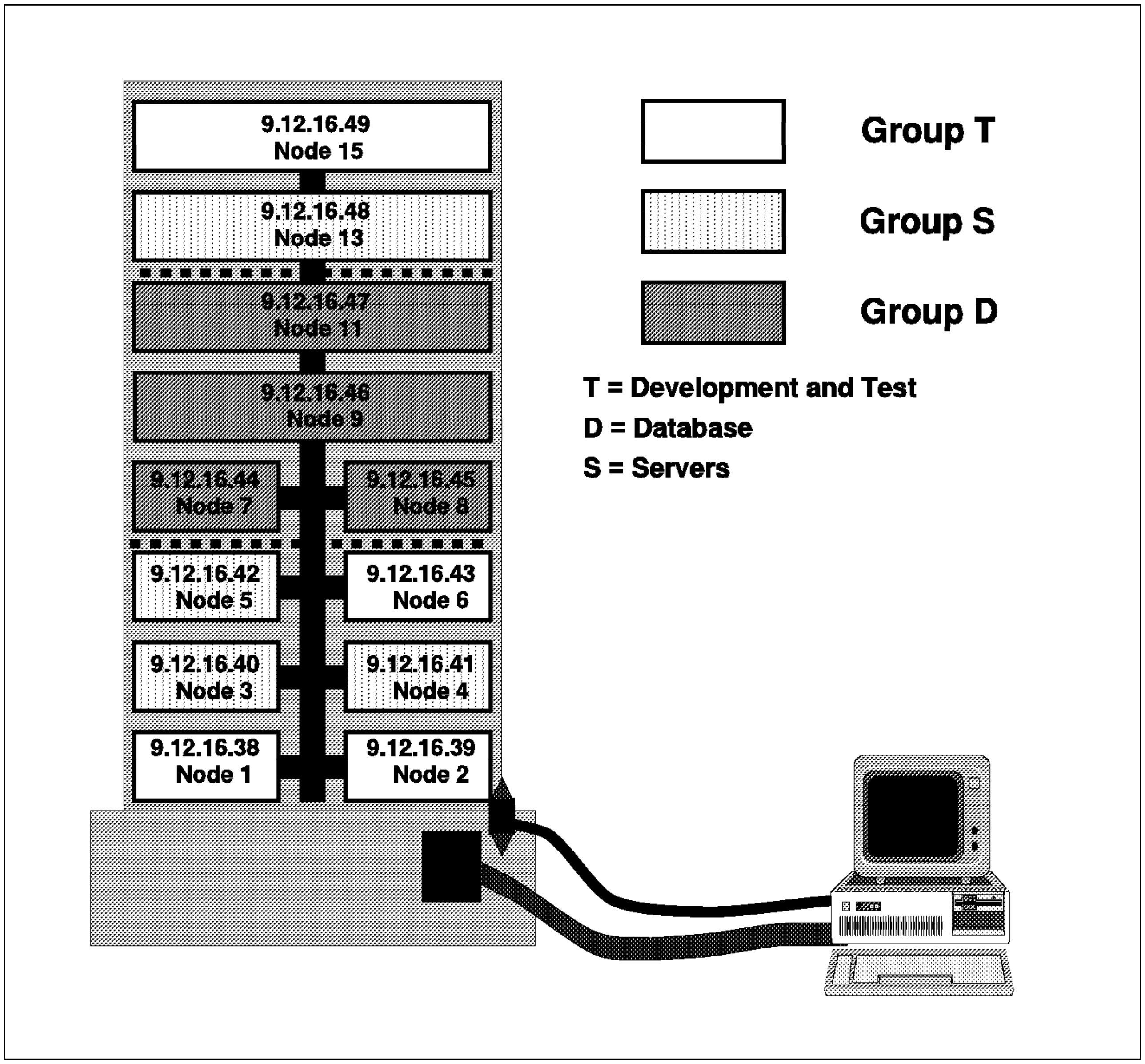


Figure 88. SP Groups and Node Partitioning

So, we add an additional group_S File Collection that has specific information for these nodes. Any files can go into this new file collection, but, in particular, we want to add:

- The original AIX passwd executable or command file. As we are using NIS on these nodes, users can change their passwords locally, without logging into the Control Workstation to do so. We should, therefore, replace the SP executable with the standard one.
- The /etc/security/limits file. It describes the limitations for these users and is specific to these nodes.

3. We also want to leave all of the Amd information relating to home directories and other Amd mounts in this File Collection.

The group-specific File Collection will be updated on these nodes with the **supper update** command run through **cron**.

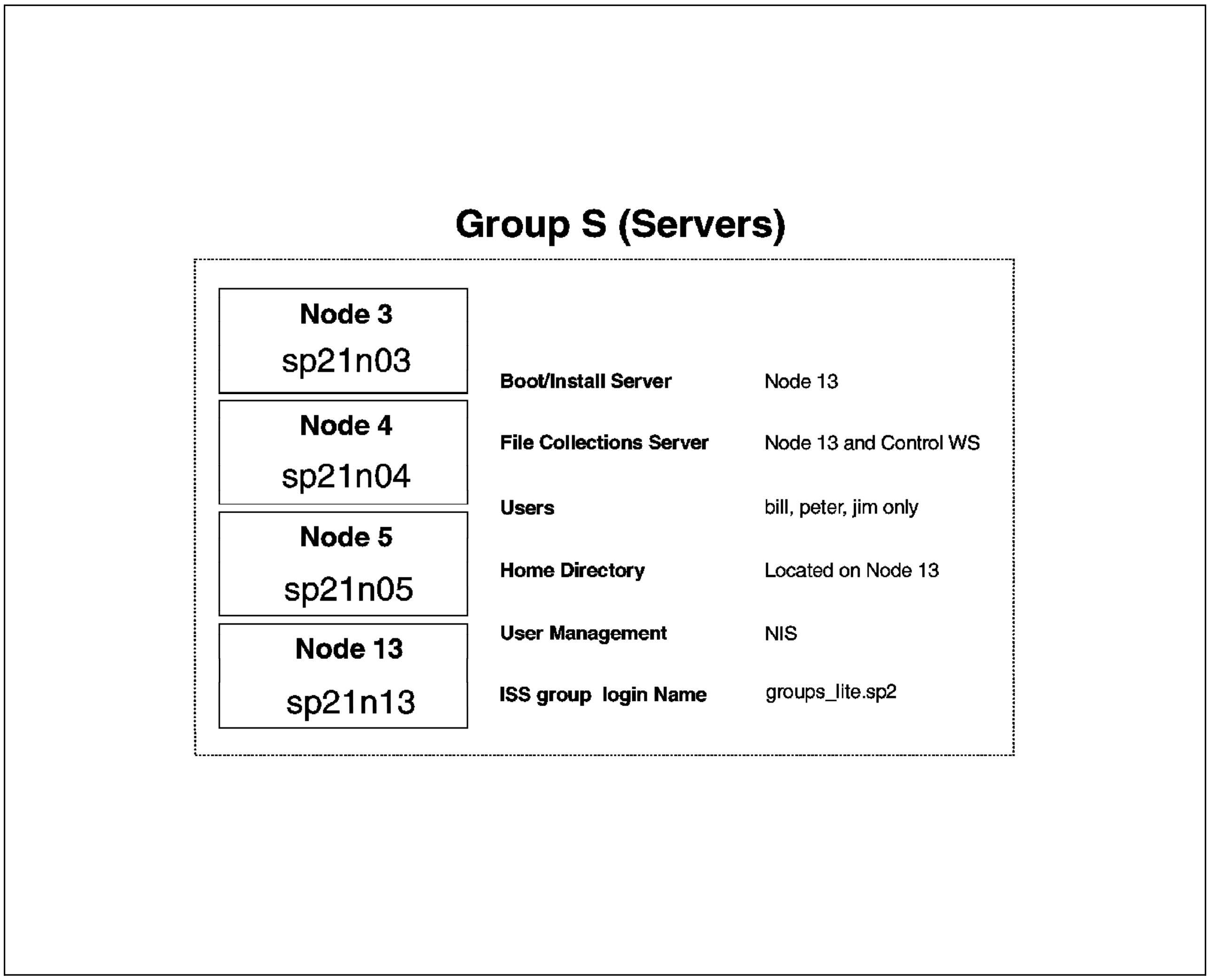


Figure 89. Node-Group S Configuration

13.4.3.2 Node-Group T

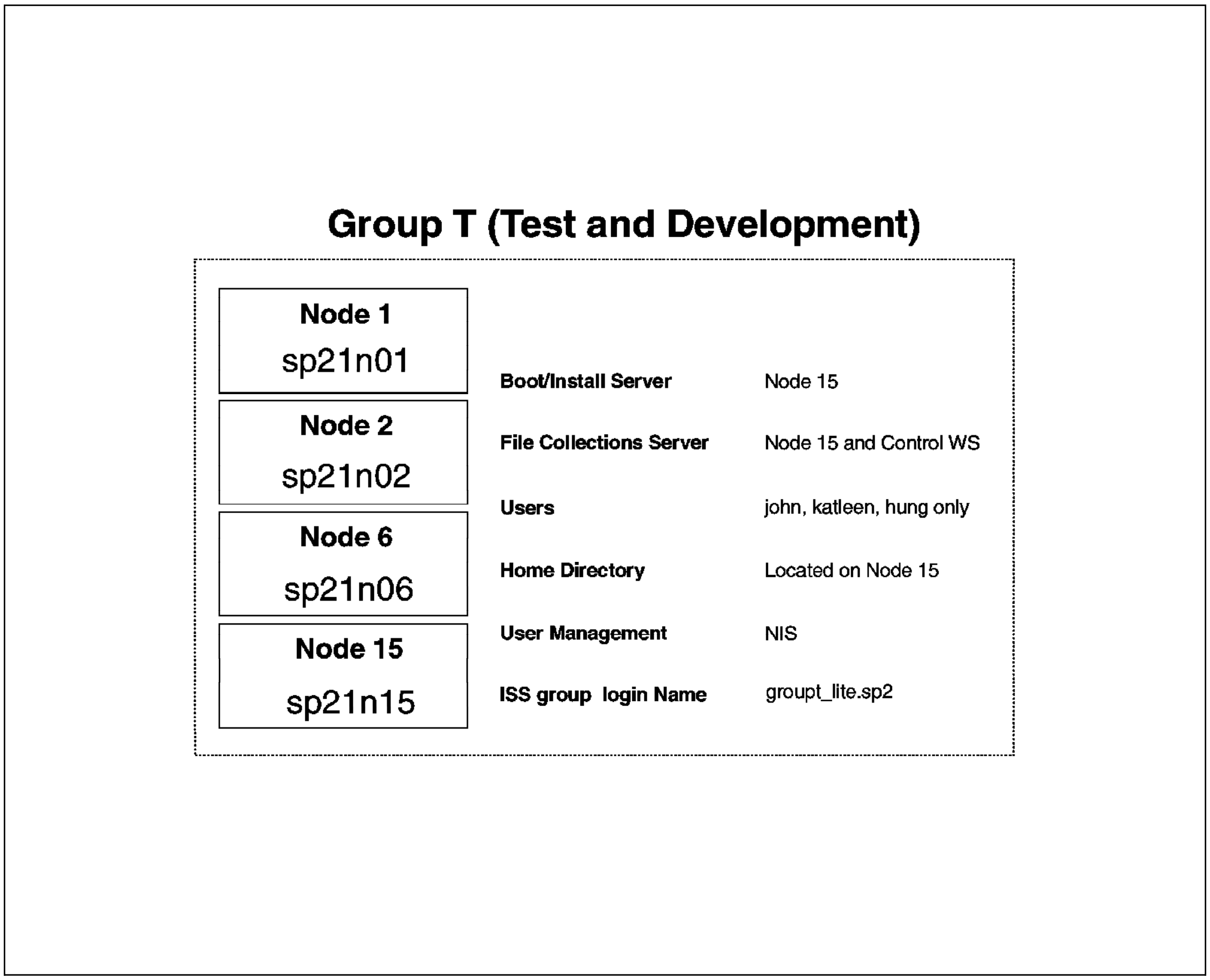
We have configured Group T similar to Group S, with a new File Collection that has files particular to this group. In this case, node 15 is the Boot/Install server and the File-Collection server for the other three nodes.

Again, we have decided to use NIS to manage these nodes from a user and password point of view, so we have configured the `user.admin` File Collection in the same way as for Group S. Figure 90 on page 220 shows how Group T will be configured.

13.4.3.3 Node-Group D

Group D is the database group of nodes. We have chosen to configure group D to have its own set of users. These users are only be allowed to log into these nodes; they will not be allowed access to other nodes within the SP. Node 11 has been set up as both a Boot/Install server and a File-Collections server for the other three nodes within this group,

We have also chosen to use File Collections to update the user and password information on the nodes in this group rather than NIS. Figure 91 on page 221 shows how group D will be configured.



Group T (Test and Development)

Node 1 sp21n01

Node 2 sp21n02

Node 6 sp21n06

Node 15 sp21n15

Boot/Install Server	Node 15
File Collections Server	Node 15 and Control WS
Users	john, katleen, hung only
Home Directory	Located on Node 15
User Management	NIS
ISS group login Name	groupt_lite.sp2

Figure 90. Node-Group T Configuration

What things must we do from a File-Collections point of view to achieve these requirements?

1. We must modify the standard File Collections in exactly the same way as for the other two groups of nodes.
2. We must add a `group_D1` File Collection, adding back in the user files that we took out for the sake of NIS (see 13.4.3.1, "Node-Group S" on page 217) and a `group_D2` File Collection to provide group- or application-specific files across these nodes.
3. We must take the `group_D2` File Collection *offline* so that node 11 becomes the master server for this File Collection.

   Figure 92 on page 222 shows how an application File Collection could be set up to run offline.

Having this application File Collection (`group_D2`) allows us to keep the master copies on one of the nodes rather than on the Control Workstation. This is likely to be a real-life requirement. For example, users maintaining application configuration files may need to replicate them across other nodes.

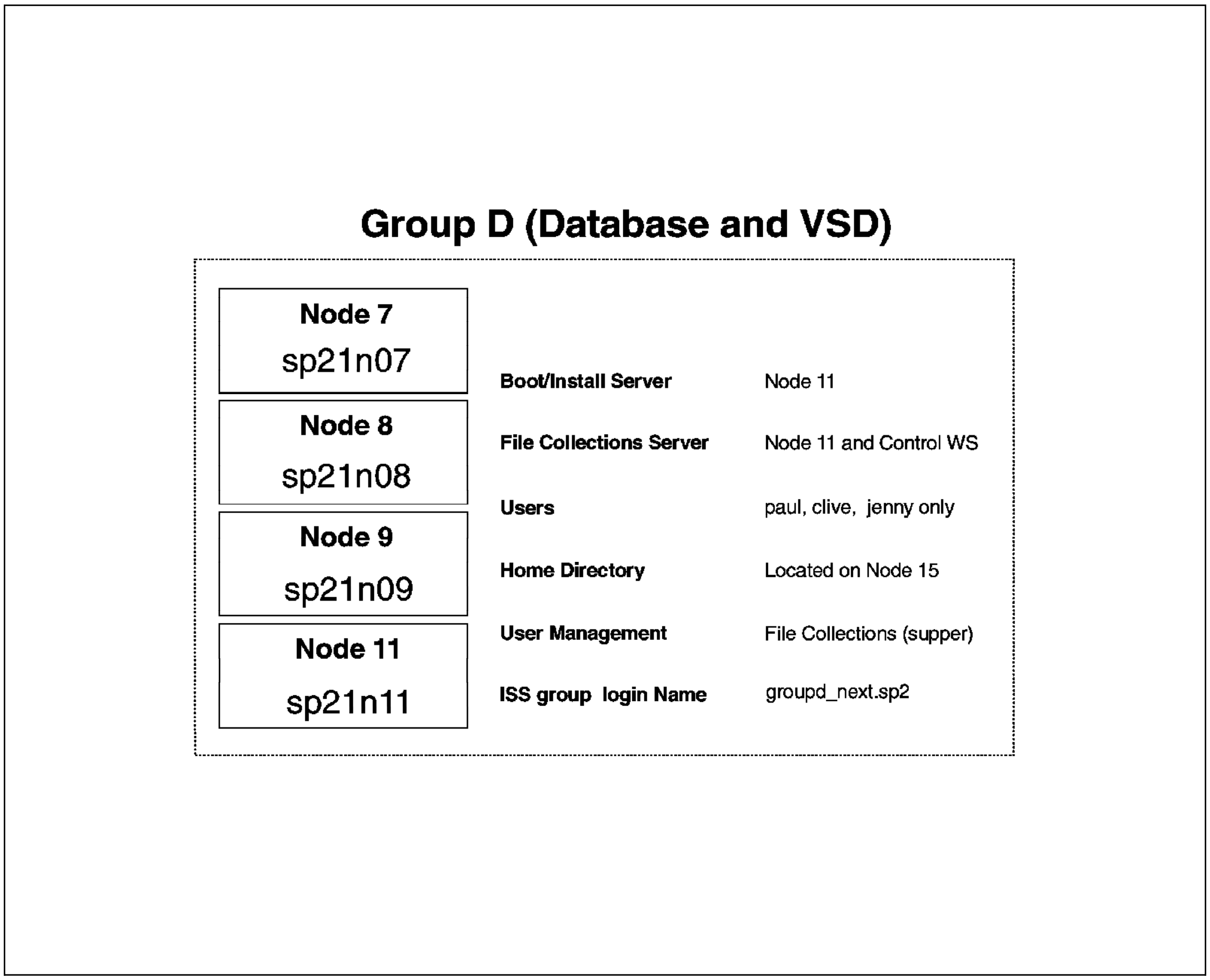


Figure 91. Node-Group D Configuration

We will not copy the standard AIX `/etc/passwd` executable to these nodes; we will continue to use the SP version. Users must change their passwords at the Control Workstation.

13.4.4 Experiences with File Collections

We found File Collections a very powerful tool for replicating information across the SP. In some cases, such as with NIS, there is overlap with other tools that can often achieve the same functions. This is not a problem in itself, but can lead to confusion. It is better, of course, to have two tools to do something rather than none!

The limitations that we encountered were:

- The File-Collection hierarchy matches the hierarchy of the Boot/Install process only. It is not possible to have a File-Collection server unless it is also a Boot/Install server. There are cases where this is not ideal.

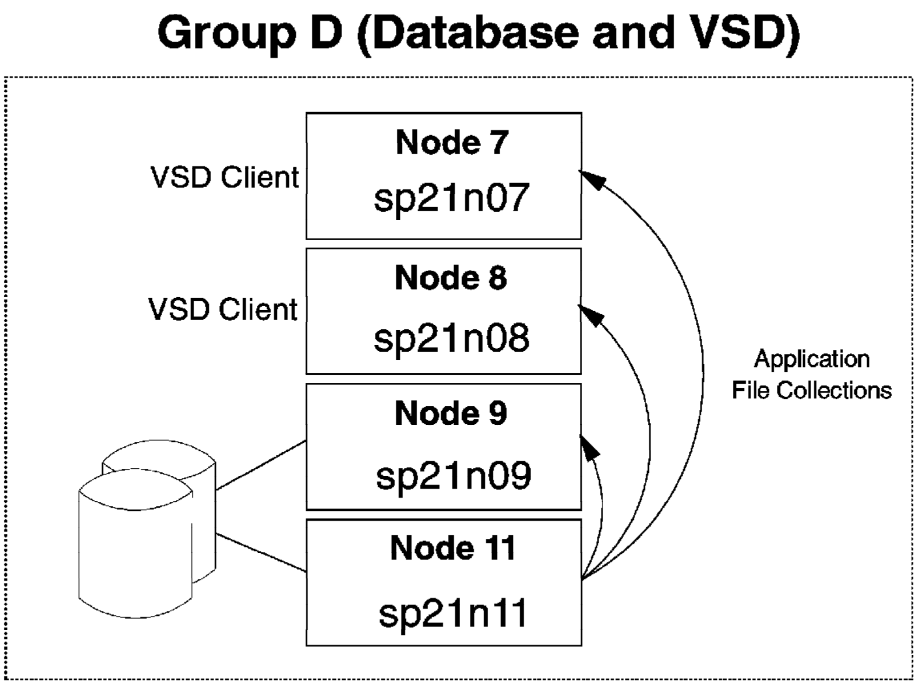


Figure 92. An Application File Collection Running Offline

You might have a Control Workstation that installs all of the nodes within a relatively small SP system. None of the nodes are Boot/Install servers in this scenario. However, you might wish to have a File-Collection hierarchy that is not as simple as this. You must set up some nodes at Boot/Install servers to achieve this, even though you do not want to install from them.

This is perfectly possible, of course. You could change nodes to be Boot/Install servers, allowing you to run our File Collections as required. If you had to re-install nodes, you could set all of the Boot/Install servers back again in the SDR on the Control Workstation.

- Although we could change the SDR setting on the Control Workstation to redefine Boot/Install servers, this information was not reflected on the nodes themselves until we did a node *customize*. That means setting the SDR and rebooting the node (as a Network Boot) to redefine that node, making it not quite so simple to change the File-Collections hierarchy instantly.

We found File Collections to be a very useful way of maintaining our configuration files in a LAN-Consolidation environment such as ours.

13.5 Network Information Service

This section describes the use of the Network Information Service (NIS). Although an SP is a machine containing multiple RISC System/6000 nodes, you do not want to maintain an SP as multiple computers, but as one system. NIS is one of the tools that can make the daily operations of an SP simple and easy.

13.5.1 Description of NIS

NIS is a distributed database containing system configuration files. By using NIS, these files will look the same throughout a network or, in this case, throughout your SP machine. NFS and NIS are packaged together. Since the SP install image includes NFS, NIS comes along, as well. The most commonly-used implementations of NIS are based upon the distribution maps containing the information from the /etc/hosts file and the user-related files: /etc/passwd, /etc/group and /etc/security/passwd.

13.5.1.1 Advantages of NIS

NIS allows a systems administrator to maintain system configuration files in one place. These files need only be changed once, then propagated to the other nodes. From a user′s point of view, the password and user credentials are the same throughout the network. This means that the user only needs to maintain one password. When the user′s home directory is maintained on one machine and made available through NFS, the user′s environment is also easier to maintain.

From an SP point of view, a NIS solution removes the SP restriction of changing user′s passwords on the Control Workstation. When you would use File Collections for system configuration file distribution (assuming the /etc/security/passwd file is part of that Collection as well), users have to change their password on the Control Workstation. When using NIS, you can control user password management accross the SP, from any given node.

Note: When configuring File Collections to true in the Site Environment, the configuration process renames the /usr/bin/passwd executable to /usr/bin/passwd.orig. In its place, a new /usr/bin/passwd file is created requesting you to change your password on the Control Workstation, when executing /usr/bin/passwd. Renaming /usr/bin/passwd.orig to /usr/bin/passwd would bring back the original executable.

13.5.1.2 NIS Terminology

The NIS environment consists of three major components:

The Domain Each machine that is part of the NIS environment is a member of the NIS domain. A larger network can contain multiple domains to separate different user environments or different projects.

The NIS Server A NIS server provides the system configuration information to the network it is serving. A NIS server can be recognized by the presence of a running *ypserv* daemon. When clients request a service, the ypserv daemon answers those requests by searching the /etc/yp/<domainname> directory. This directory contains two files for each service, a .dir and a .pag file, representing the *key* and *value* of the service.

There are two different kinds of servers. One is the Master server, which controls all physical changes to system configuration files. This machine also serves as the replication server. If Slave servers are configured, the Master server has a *ypupdated* daemon that forces an update on the Slave servers (using the **ypxfer** command) when a change to the configuration files occurs. The Master server may also run the *yppasswdd*, allowing users to update their passwords anywhere in the NIS domain.

Note: Each domain can only host one Master server.

A Slave server functions as a copy of your Master server, making the NIS services more available in the case of Master-server malfunction. Should the Master-server machine fail, login procedures in the network can continue as normal; only updates to the NIS database are not possible (ypupdated is not available). Also, when serving a large number of workstations, a Slave server can balance the load of the Master server. A Slave server is mandatory if your NIS domain must serve different networks (that is, going over routers) NIS Clients cannot communicate to a server over a router.

The NIS Client Any other machine in the NIS domain that accesses the NIS distributed facilities. A Client can be recognized by the *ypbind* daemon. As mentioned earlier, each network segment in the NIS domain must have at least one server. The ypbind process searches for its server at boot/startup time by issuing a broadcast on the segment where it resides. Broadcasts cannot pass over routers.

Figure 93 on page 225 illustrates a possible NIS implementation on an SP.

13.5.1.3 Using NIS and/or File Collections

The SP is provided with another tool to insure that system configuration files look the same throughout your SP network, *File-Collection Management.* When configuring the SDR, you are asked if you want to use this facility. When answered affirmatively, the Control Workstation configures a mechanism for you that will periodically update the system configuration files (you specify the interval). The files included in that configuration are:

- All files in the directory /share/power/system/3.2
- Some of the supper files
- The Amd maps
- The user administration files:
  - /etc/group
  - /etc/passwd
  - /etc/security/group
  - /etc/security/passwd
- /etc/acct/holidays

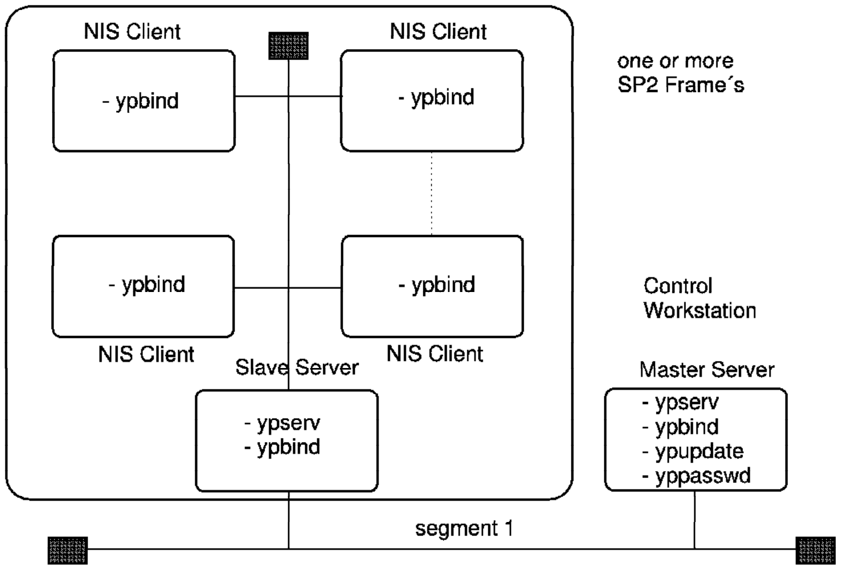


Figure 93. A NIS Configuration on the SP

This setting also causes the replacement of the /usr/bin/passwd command by a dummy command, indicating that you must update your user password on the Control Workstation. In terms of user administration, the File-Collection Management system is an alternative to using NIS for users who are not familiar with NIS or do not want to use it.

However, even though File Collections are implemented, NIS can still be used in parallel for user administration. When doing so, the File Collections should exclude the user administration files and the /usr/bin/passwd command should be restored (the original passwd command is stored as /usr/bin/passwd.orig).

For more information on File Collections see 13.4, “SP File Collections” on page 206.

13.5.1.4 Using NIS Netgroups and/or Interactive Session Support

NIS provides the ability to hierarchically group users and/or hosts. These groups can be used as classifications in the /etc/passwd and /etc/exports files to define login permissions or to mount remote filesystems. The file that contains the definitions is stored in the /etc/netgroup file. An example of the /etc/netgroup file is provided in 13.5.3.1, “Use of Netgroups” on page 229.

Interactive Session Support (ISS) does the same where assigning a login host is concerned. Based on certain criteria (defined by the System Administrator or by Loadleveler), ISS determines the node to which the next login request should be directed. ISS can work together with NIS.

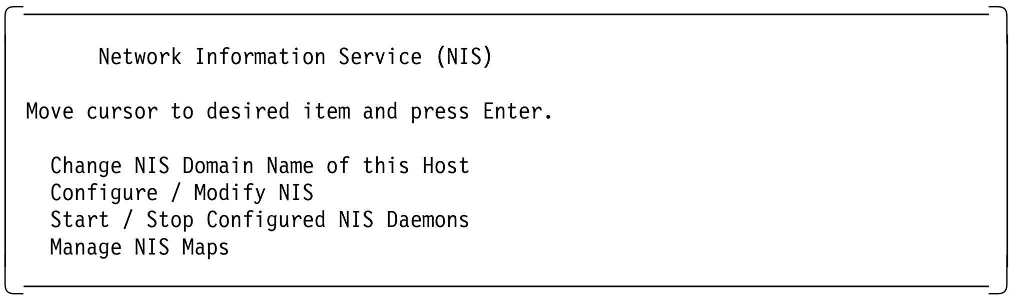
For further information on ISS see 10.2, “SP Nameservers and Interactive Session Support” on page 144.

13.5.2 How to Set Up NIS

Using the SMIT panels, you can set up NIS, manage it and control the NIS daemons. The NIS-associated SMIT panels can be found by selecting the following:

```
└─►Communications Applications and Services
   └─►NFS
      └─►Network Information Service (NIS)
```

Figure 94 shows the NIS panel.



```
       Network Information Service (NIS)

Move cursor to desired item and press Enter.

  Change NIS Domain Name of this Host
  Configure / Modify NIS
  Start / Stop Configured NIS Daemons
  Manage NIS Maps
```

Figure 94. NIS SMIT Panel

13.5.2.1 General Configuration

Each node in a NIS environment must be a member of a domain and each node can only belong to one domain at a time. So, you must change or set the domainname on each node. Either go to the appropriate SMIT panel and specify the domainname:

```
└─►Change NIS Domain Name of this Host
```

Or, use the command **/usr/bin/domainname** <**cluster-name**>. This will take effect immediately, or at system reboot, or both. This configuration is set in the /etc/rc.nfs file, where the comment characters are removed from the lines to set a NIS domain.

13.5.2.2 Configuring a Master Server

In your planning, you must decide whether (1) you will have Slave servers and (2) you will allow users to change their passwords anywhere in the network. With these decisions made, you can now configure the Master server. Go to the appropriate SMIT panel via:

```
└─►Configure / Modify NIS
  └─►Configure this Host as a NIS Master Server
```

The panel with which to configure the Master server appears in Figure 95 on page 227.

Specify the hosts that will be configured as Slave server and, if any, specify yes to start the *ypupdated*. In most cases, the *yppasswdd* is started, as well. These configuration settings can also be found in the /etc/rc.nfs file.

```
        Configure this Host as a NIS Master Server

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

   HOSTS that will be slave servers                  [sp21n01]
 * Can existing MAPS for the domain be overwritten?    yes
 * EXIT on errors, when creating master server?        yes
 * START the yppasswdd daemon?                         yes
 * START the ypupdated daemon?                         yes
 * START the ypbind daemon?                            yes
 * START the master server now,                        both
      at system restart, or both?
```

Figure 95. Configuring NIS Master Server

By default, the NIS Master server maintains the following files:

- /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

Any changes to these files must be propagated to clients and slave servers. This can be done using SMIT:

```
└─►Manage NIS Maps
    └─►Build / Rebuild Maps for this Master Server
```

Either specify a particular NIS map by entering the name representing the filename or leave the default value of **all**, then press Enter. You can also do this manually, by changing to directory /etc/yp and entering the command **make all** or **make** <**map-name**>. This propagates the maps to all NIS clients and transfers all maps to the Slave servers.

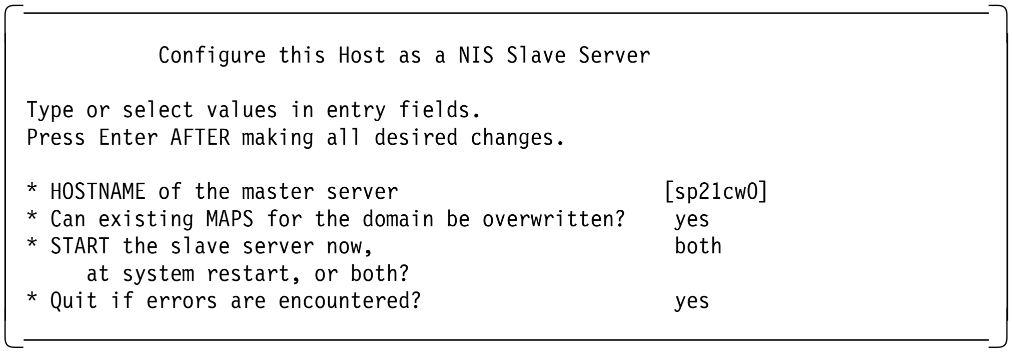
13.5.2.3 Configuring a Slave Server

A Slave server looks and feels the same as a Master server, with one exception: it is a read-only server. Therefore, it is not capable of updating any NIS maps. Making a Slave server implies that all NIS maps will be physically present on the node configured as the Slave server. As with a Master server, the NIS map files on a Slave server can be found in the directory /etc/yp/<domainname>.

To configure a Slave server, go through the SMIT panels:

```
└─►Configure / Modify NIS
    └─►Configure this Host as a NIS Slave Server
```

Figure 96 on page 228 shows the resulting dialogue.



```
          Configure this Host as a NIS Slave Server

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

* HOSTNAME of the master server                      [sp21cw0]
* Can existing MAPS for the domain be overwritten?    yes
* START the slave server now,                         both
     at system restart, or both?
* Quit if errors are encountered?                     yes
```

Figure 96. Configuring a NIS Slave Server

Configuring a Slave server starts the ypbind daemon, which searches for a server in the network running ypserv. Shortly afterwards, the ypserv daemon of the Slave server itself will start.

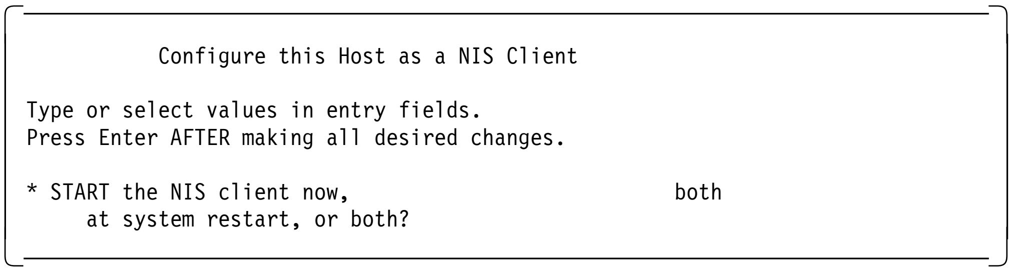
In a lot of situations, the Slave server must also be able to receive and serve login requests. If that is the case, the Slave server must also be configured as a NIS client. Refer to 13.5.2.4, "Configuring a NIS Client" for configuration details.

13.5.2.4 Configuring a NIS Client

A NIS client retrieves its information from the first server it contacts. The process responsible for establishing the contact with a server is ypbind. To set up a NIS client, go through the following SMIT panels:

```
 └─►Configure / Modify NIS
    └─►Configure this Host as a NIS Client
```

Just pressing Enter in the dialogue shown in Figure 97 is sufficient:



```
          Configure this Host as a NIS Client

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

* START the NIS client now,                           both
     at system restart, or both?
```

Figure 97. Configuring a NIS Client

If you are familiar with the command-line format of this command, you could instead specify: **/usr/sbin/mkclient -B**. This configuration step not only starts the ypbind daemon, but also adds new lines to the /etc/passwd and /etc/group files. In the /etc/passwd file, the **mkclient** command adds the string **+::0:0:::**, the escape sequence of NIS for resolving the user's credentials. In the /etc/group file, a similar-looking string is added (**+:**) which is the escape sequence of NIS for resolving group characteristics.

The NIS maps for a client are retrieved from the "bound" server. As soon as a user logs into a NIS system, the login process searches the /etc/passwd file from top to bottom. If the user has an entry in the /etc/passwd file, the user will login as usual. If the user does not have a physical entry in the /etc/passwd file and the login process encounters the **+::0:0:::** sequence, the login process is completed by NIS.

13.5.3 Managing NIS

As described in the previous sections, setting up NIS is a fairly simple task. Most of the work is related to the design of your domain, the naming conventions you will use and deciding which nodes will be NIS clients or Slave servers. Once these definitions are clear, configuring NIS is easy. There are, however, a few points you must keep in mind:

- Configure NIS in stages. Make sure you define your Master server first, then your Slave servers and, finally, the clients. Also, when going over routers, first configure the "master" segment, then the Slave servers in the other segments and finish with the clients. An example of how to set up a Slave server in another segment is described in 13.5.5, "Suggestions" on page 231.
- Any changes to the system configuration files managed by NIS will only take effect after an enforced update of the NIS databases. It is not automatic, unless you add an update command to cron.

13.5.3.1 Use of Netgroups

By default, NIS is not provided with *netgroups*.

Netgroups are logically-bundled nodes or users that can be granted or denied access to certain AIX services. The file used to define a netgroup is /etc/netgroup. The /etc/netgroup syntax is (**-,user,domain**) or (**host,-,domain**) where domain can be blank (commas are required). This results in having *host*-netgroups and *user*-netgroups.

Host-netgroups can be used, for example, in the /etc/exports file, where access can be granted to a netgroup: **-access=group_d**.

User-netgroups may be used in the /etc/passwd file, where you can specify that only netgroup *team* can log in: +**@team**. Also, specific to the SP, netgroups can be used to specify SP access control for groups of users with the **spacs_cntrl** command.

An example of the /etc/netgroup file is shown in Figure 98.

```
group_d    (sp21n07,-,) (sp21n08,-,)
           (sp21n09,-,) (sp21n11,-,)
group_s    (sp21n03,-,) (sp21n04,-,)
           (sp21n05,-,) (sp21n13,-,)
group_t    (sp21n01,-,) (sp21n02,-,)
           (sp21n06,-,) (sp21n15,-,)

allnodes   group_d group_s group_t

allsp2     allnodes (sp21cw0,-,)

users      (-,clive,) (-,peter,)
           (-,jim,) (-,kathleen,)
           (-,john,) (-,bill,)
team       (-,hung,) users

world      (-,guest,)
```

Figure 98. Sample /etc/netgroup File

The example in Figure 98 was used in our environment where group_d represents the Database and VSD nodes, group_s represents the general servers and group_t represents the Test and Development nodes.

13.5.3.2 Domain Name Server (DNS) and NIS

Since NIS replicates the /etc/hosts file, host resolution may also be done using NIS. There are four ways AIX can do host resolution:

No DNS, No NIS In this case, the /etc/hosts file must provide the answer to a request of a node in the network.

No DNS, but NIS Only NIS host maps are referenced. If NIS does not have the answer, the /etc/hosts file is not referenced.

DNS, No NIS For using DNS, the /etc/resolv.conf file must be configured. A request to a foreign host invokes TCP/IP to resolve the host on the nameserver. If the nameserver cannot resolve the hostname, TCP/IP will refer to the /etc/hosts file, possibly with a short delay.

DNS and NIS As with the previous point, if DNS cannot resolve the host, NIS is referred to for hostname or hostaddress resolution. If NIS cannot resolve the request, the /etc/hosts file is not referenced.

13.5.3.3 Configuring a Slave Server Behind a Router

When NIS must serve parts of its domain behind a real router, you must configure this Slave server as follows:

- Set the domainname.
- Start the ypbind daemon.
- Bind to the server with a Servers IP-address: **ypset 192.12.100.137** (so the node knows where to find its resources).
- Create the Slave server with **ypinit -s** <**master-name**>
- Start the (slave-) server daemon: **ypserv**.

13.5.4 NIS Security

NIS has a bad name as far as security is concerned. However, for the majority of the problems, that is history. In the past, it was possible to rebind any client to any server in the network. In AIX 3.2.5, a PTF is available that allows only the local root userid to perform the action **ypset -ypsetme**. Also, it used to be possible to bind any foreign host to a particular domain by simply setting your domain to the one to which you wanted to add yourself, starting the ypbind process and connecting the ypbind to the server with ypset. This is no longer possible either as you can now exclusively allow only certain hosts to bind to a server. This can be set in the file /etc/yp/securenets. Default installations do not provide a file containing the hostnames of the trusted hosts in the /etc/yp directory. One can be created with the syntax:

```
hostname
```

where hostname represents the fully-qualified hostname [that is, with extension of the (nameserver-) domain].

13.5.5 Suggestions

This section gives you a few hints and tips that may be useful in your day-to-day operations with NIS.

- Refer to the “NIS and NFS” publication of O′Reilly and Associates for more detailed information on NIS (and NFS).
- Do not define root or any other system-default user ID to NIS.
- Before making system backups of your nodes, remove the +::0:0::: entry from the `etc/passwd` file.
- For security, change the NIS entry in the `/etc/passwd` file to +:*:0:0:::. The “*” will insure that no “+” user can login without a password when NIS stops running.
- To reduce NIS network load and to increase the availability of the NIS services throughout your SP, make all clients in the SP Slave servers. When running as Slave servers, the nodes can boot independently from any other server, provided that you update the `/etc/rc.nfs` file. Add a line to the end of the file that looks like this:

  ```
  ypset localhost
  ```

  Also, change the entry that starts up the ypbind daemon to:

  ```
          startsrc -s ypbind -a ″-ypsetme″
  ```

  Stop and start the ypbind daemon with the -ypsetme option or reboot the node.
- When configuring a large number of NIS nodes, use the commands generated with SMIT (smitty -x) as arguments for the **dsh** command.
- Adapt the boot sequence of your SP to the hierarchy of the NIS environment: Master server first, followed by the Slave server ending with the clients.

Appendix A. Known Command and Script Errors

This appendix describes some PSSP commands and scripts containing minor bugs. The functions of the commands and script are not explained in detail. A one-liner will be used for that, if necessary. For each command, we provide the proper reference to where detailed information may be found.

These bugs cause a slightly different behavior than expected. In the descriptions of the bugs, we explain this behavior and provide a work-around.

A.1 amd_start

If you configured your SP to use the automounter daemon (Amd), the amd daemon is started for you automatically. The default configuration creates a sample amd map, stored in the /etc/amd/amd-maps direcory as amd.u. The script responsible for starting the amd daemon is the **amd_start** Perl script in the /etc/amd directory.

This script should start amd as follows:

```
amd -t 16.120 -x all -l $log -p <watch-point> <amd-map>
                         ......<watch-point> <amd-map>
```

However, the $log variable is empty. At the beginning of this script, there is a $amdlog variable that is set to /var/adm/SPlogs/amd/amd.log, but wrongly referenced.

This causes the amd daemon to start as follows:

```
amd -t 16.120 -x all -l -p /u /etc/amd/amd-maps/amd.u ........
```

As a result, the **-p** file in the root directory becomes the log file. This file is quite impossible to read, since the dash (**-**) is interpreted as soon as you try to enter **vi /-p**.

Go through the next two steps to circumvent this behavior:

1. Edit the *start_amd* script and change the following line:

   ```
   system("nice --4 $amd -t 16.120 -x all -l $log -p > /etc/amd/amd.pid...
   ```

   to

   ```
   system("nice --4 $amd -t 16.120 -x all -l $amdlog -p > /etc/amd/amd.pid...
   ```

2. Before removing the "-p" file, you might want to save it using:

   ```
   mv find / -name "-p" -xdev -print /tmp/amd.log
   ```

   Remove the "-p" file from the root directory with:

   ```
   del find / -name "-p" -xdev -print
   ```

Refer to 13.3, "The Berkeley Automounter (Amd)" on page 194, to the *IBM 9076 Scalable POWERparallel Systems SP2 Administration Guide* and the *IBM 9076 Scalable POWERparallel Systems SP2 Installation Guide* for more detailed information on Amd.

A.2 Setup_server

The setup_server script is more or less the core script of the PSSP software. It configures a lot of things. This section covers two configuration files in particular that can become corrupted in certain situations and under certain conditions.

The files considered are the:

- `/etc/bootptab` file
- `/tftpboot` directory

The condition is: if you have defined hostnames of non-fixed length with a number as a designator at the end.

Example:

```
sp2n1
  ⋮
sp2n10
  ⋮
sp2n100
  ⋮
```

Each time you change definitions in the SDR, the setup_server script removes old information first. The problem is that the setup_server removes the old information using a wildcard as the hostname.

Let us assume, using the previously mentioned hostnames, that we change the SDR information for node `sp2n1`. The setup_server command will remove definitions for `sp2n1*`, as well as those for the range sp2n10 to sp2n19, and for sp2n100 to sp2n199 and so on.

There are two ways to circumvent or prevent this problem:

1. Choose fixed-length hostnames.

   Using our previous example, you should configure your hostnames in this manner:

   ```
   sp2n001
     ⋮
   sp2n010
     ⋮
   sp2n100
     ⋮
   ```

   This circumvention is the most simple way to fix the problem and is the recommended way of doing so.

2. Change the setup_server command in the function Unconfig_Install_Svr in the **sed** command, deleting entries in `/etc/bootptab` file and, in the **rm** command, removing old `/tftpboot` entries.

   This is, however, not recommended. Any changes to the setup_server would be nullified with the next Program Temporary Fix and you may affect the consistency of your environment in case of error.

Refer to *IBM 9076 Scalable POWERparallel Systems SP2 Installation Guide* and *IBM 9076 Scalable POWERparallel Systems SP2 Administration Guide* for details on how to use and configure the setup_server command.

A.3 firstboot.ssp

When installing the SP nodes, a system image is propagated from the control workstation or from an install server. Each node installation is followed by a customization phase. This phase is invoked when the node is rebooting just after the initial installation of the install image. The standard way for AIX to initiate the customization phase is to create a *firstboot* file in the /etc/ directory. This file is executed by one of the earlier entries in /etc/inittab:

```
fbcheck:2:wait:/usr/lib/dwm/fbcheck
```

The firstboot file for installing an SP node contains many steps to ensure that the node fits in the parallel suite. You may customize this file to your own needs. You must do this by creating a file in /tftpboot called *firstboot.cust*. This file is merged with the */usr/lpp/ssp/install/config/firstboot.ssp* file and executed as /etc/firstboot at the time a node reboots after installation.

One of the lines in the file */usr/lpp/ssp/install/config/firstboot.ssp* has a little bug, causing the installation log to be incomplete:

```
echo ″exec 2>&1 > $LOGDIR/$CLIENTN.firstboot.log″ >> /etc/firstboot
```

Stdout is not specified until the latter part of the line, causing stderr messages to disappear to stdout without knowing what stdout is. As a consequence, no install error messages are logged.

Changing the line as shown causes the installation messages to appear in the installation log file:

```
echo ″exec > $LOGDIR/$CLIENTN.firstboot.log 2>&1″ >> /etc/firstboot
```

Refer to the *IBM 9076 Scalable POWERparallel Systems SP2 Installation Guide* and the *IBM 9076 Scalable POWERparallel Systems SP2 Administration Guide* for details on the firstboot files.

Appendix B. New Features For ssp.clients

This section is a copy of ssp.clients.README, provided with APAR IX49179, describing the enhancements to the spp.clients product.

B.1 Description

This separately-installable option provides the client commands for the Kerberos-authenticated services, the System Monitor and 9076 SP remote commands rsh and rcp. It also includes the authentication commands that users need to manage the tickets allowing access to those facilities.

This is an implementation of the Kerberos-authentication system (Version 4) based on the MIT Project Athena distribution.

B.1.1 Installation

This option is a prerequisite for installing the ssp.basic and ssp.gui options on the Control Workstation for your 9076 SP system and for installing ssp.authent on the Control Workstation or other RISC/6000 workstation that will be an authentication server. Also install this option on RISC/6000 workstations from which you expect to perform 9076 SP administration tasks using the authenticated facilities listed above.

B.1.2 Kerberos Supports Multiple Interfaces

PSSP APAR IX49179 modifies the design of authentication services so the network name of any interface on the control workstation or a node may be used as the target of sysctl, dsh, rsh, or rcp. Prior to this change, the Kerberos-authenticated servers could support only a single service instance. These services could only be addressed using the network interface corresponding to the node′s standard hostname. This not only inconvenienced users, but caused the Estart command to fail in certain cases, as reported by APAR IX48966.

The modified server interface allows requests to be received on any available interface. This requires that a Kerberos service principal be defined for each interface, using the short form of its network name. Server key files for the nodes (and control workstation) now contain entries for each of these service instances.

To take advantage of this change, you need only recustomize the nodes. This can be accomplished by setting the bootp response to ″customize″ (see step 21 of the Installation Guide) using either smit or the spbootins command. For example, to customize a single rack of thin nodes, run ″spbootins -r customize 1 1 16″. This will create new srvtab files for all of the nodes and the control workstation.

Once this is done, network boot the nodes so that the new srvtab files are distributed to all the nodes. If you have not changed any hostnames or network interfaces on the nodes, you may use an alternative method to distribute the srvtab files, such as as rcp or ftp. The srvtab file for each node is located in the /tftpboot directory on the boot-install server for the node, with a filename of <hostname>-new-srvtab; where <hostname> is the short form of the node′s standard hostname. If you have configured your authentication realm with the control workstation as a PSSP authentication server, you may remove the ..-new-srvtab file after copying it to the node.

WARNING: If you use afs authentication, or use Kerberos servers but not on the control workstation, NEVER remove the ..-new-srvtab files in /tftpboot on the control workstation. In this case you may, however, remove the boot-install server′s copy, when the boot-install server is not the control workstation.

If you use an alternative to network boot, you should also take the following additional steps:

1. comment out each node′s entry in the /etc/bootptab entry on the boot-install server for the node.
2. Use the SDRChangeAttributes command to set the bootp_response attribute for each node from ″customize″ to ″disk″.

Following application of this fix, the Kerberos database and srvtab files will automatically be updated whenever you add or rename a network interface on a node (using smit or the spadaptrs command), or change a node′s hostname (using smit or the sphostnam command), and subsequently customize the node (using smit or the spbootins command).

If you make these kinds of changes on the control workstation, Kerberos files are updated by running the setup_server command.

B.1.3 Additional Information About srvtab File Changes

If a node has three network interfaces named as follows:

```
  en0     r2n5.abc.org
  css0    r2n5sw.abc.org
  fi0     r2n5fi.finet.abc.org
```

and the hostname (initial_hostname SDR attribute) is ″r2n5sw″, the name of the srvtab file created in /tftpboot is r2n5sw-new-srvtab.

To display this node′s srvtab, use:

```
ksrvutil -f /tftpboot/r2n5sw-new-srvtab list (on its boot server)
```

or

```
ksrvutil list                               (on the node)
```

and the output would appear like:

```
Version    Principal
   2     rcmd.r2n5@ABC.ORG
   2     rcmd.r2n5sw@ABC.ORG
   2     rcmd.r2n5fi@ABC.ORG
```

B.1.4 Recreating a Missing or Corrupted srvtab File

If the control workstation is a PSSP authentication server, the normal way to create (or recreate) a srvtab file is to customize the node. If you would rather perform this task manually, use the following procedure:

```
# Change to the /tftpboot directory.
cd /tftpboot

# Create separate srvtab files for each instance.
/usr/lpp/ssp/kerberos/etc/ext_srvtab -n r2n5 r2n5sw r2n5fi

# Combine the instance files into a single
# file for the hostname.
# Skip this step if the node has only the en0 adapter

/bin/cat r2n5-new-srvtab r2n5sw-new-srvtab r2n5fi-new-srvtab >srvtab-$$
/bin/rm r2n5-new-srvtab r2n5sw-new-srvtab r2n5fi-new-srvtab
/bin/mv srvtab-$$ r2n5sw-new-srvtab
```

When you are using a Kerberos server on another workstation, you can use this same procedure on that system, and transfer the output file to the node using ftp, or login to the node as root and use rcp to copy (pull) the file from the server. Unless you already have a usable /etc/krb-srvtab file on the node, you cannot use rcp on the server to copy (push) the file to the node.

Appendix C. Diagnosing Authentication Problems

Users of the Kerberos-authenticated services have only a few ways of interacting with the authentication services of the 9076 SP2 system. In general, the sequence of operations required to perform an administrative task using these services consists of:

1. Identifying yourself to the authentication service. This process obtains a ticket-granting ticket for the client principal based on either a user′s Kerberos password or a service′s private key. The former is achieved interactively using the **kinit** command. Background processes running as root and executing shell scripts can use the **rcmdtgt** command to get a ticket as service principal **rcmd.***hostname*.

2. Invoking a client command for one of the authenticated services, such as *sysctl* or *spmon*. The client command uses Kerberos facilities to obtain a service ticket, the credentials that it passes to the application server to identify the invoking principal.

3. Using the **kdestroy** command to terminate the authenticated state by destroying any tickets found in a ticket cache file belonging to the client. If you do not use **kdestroy** or **rm** to remove the ticket cache file, then tickets it contains can be reused until they expire and are automatically removed by the next successful **kinit** command.

Error messages on stderr are the principal diagnosis tool for the user who experiences problems using these facilities. Errors reported by the authentication services themselves generally have the message number prefix *2502*, *2503*, or *2504*.

C.1.1 Problems Establishing a User Principal′s Identity

When identifying yourself to the authentication service, these are the most common error messages that you might encounter:

C.1.1.1 Kerberos Principal Unknown

You did not enter a principal name defined in the database. Perhaps you misspelled it, or the administrator did so when entering it into the database. This error can occur when using a secondary authentication server if the primary database has not been propagated since your principal was added. Check with the administrator responsible for maintaining the authentication service to determine if this is the case.

C.1.1.2 Bad Kerberos Name Format

You probably entered *name.*admin in response to the prompt “Kerberos name:”. You are not allowed to specify an instance in addition to the name. If you want to enter the name and instance together, enter them as the command-line argument when you invoke **kinit.**. To force **kinit** to prompt you separately for the instance, invoke it with the **-i** flag.

C.1.1.3 Incorrect Kerberos Password

You entered the wrong password or your password was recently changed and has not yet been propagated to the secondary authentication server you are using. If you incorrectly entered your password, just try again. Otherwise, if you suspect an out-of-date database, contact the administrator of your authentication service. The recovery action in this situation is to force an update to occur, without waiting for the normal **cron** process.

C.1.2 Forcing the Propagation of Database Changes

To force the propagation of database changes:

1. Issue the **kinit** command, specifying as your principal name any user principal listed in the root user′s *.klogin* file on the primary server. The administrative principal name that was used to set up authentication on the primary server can be used, or any other user principal the administrator has subsequently added to the file.
2. Issue the following command to remotely perform the database propagation from the primary server to all secondary servers:

   ```
   /usr/lpp/ssp/rcmd/bin/rsh primary /usr/kerberos/etc/push-kprop
   ```

   where *primary* is the hostname of the primary server.

   Successful propagation is reported by a message for each secondary server hostname. If unsuccessful, review the *kpropd.log* file (see information on daemon log files below).

C.1.3 Problems Establishing a Service Principal′s Identity

The authentication service also provides commands that can be used to allow background processes to invoke authenticated services when the **kinit** command cannot be issued, because no user is logged in to the client AIX system. Various 9076 SP2 installation and system management scripts use the **rcmdtgt** command to identify themselves to the authentication system and thereby obtain a ticket-granting ticket. This procedure should not fail; if it does, the most likely error message reported will be:

C.1.3.1 Incorrect Kerberos Password

If this error occurs during installation or when performing administrative tasks requiring remote execution on 9076 SP2 nodes, it can indicate one of several error conditions:

- The process is not running as root. It cannot read the server key file, `/etc/krb-srvtab`, on the client system. You must, if authorized, log in as root and retry the failing task.
- The server key file is out-of-date with respect to the authentication database.
- The server key file does not exist.

C.1.4 Comparing Service Key Versions

An administrator running as **root** can compare the versions of the server keys in the server key file and the database using the **ksrvutil** or **klist** command on the server system and by examining a dump of the authentication database on the primary authentication server, when AFS is not being used.

Use the **ksrvutil** or **klist** command to show the version numbers of service keys in /etc/krb-srvtab. The following example shows the use of **ksrvutil** to display the key versions for service principals on a Control Workstation:

```
# ksrvutil list
Version    Principal
   2     rcmd.cwktr@XYZ.ABC.COM
   2     hardmon.cwktr@XYZ.ABC.COM
   2     rcmd.cwkfddi@XYZ.ABC.COM
   2     hardmon.cwkfddi@XYZ.ABC.COM
   2     rcmd.cwken1@XYZ.ABC.COM
   2     hardmon.cwken1@XYZ.ABC.COM
   2     rcmd.cwksta@XYZ.ABC.COM
   2     hardmon.cwksta@XYZ.ABC.COM
#
```

The following example shows the use of **klist** to display the key versions for service principals on a 9076 SP2 node:

```
# klist -srvtab
Server key file:   /etc/krb-srvtab
Service         Instance        Realm      Key Version
----------------------------------------------------
rcmd            node3fi         XYZ.ABC.COM      1
rcmd            node3tr         XYZ.ABC.COM      1
rcmd            node3sw         XYZ.ABC.COM      1
rcmd            node3en         XYZ.ABC.COM      1
#
```

You can determine the versions of service keys in the authentication database by locating the entry for the target service principal in a dump of the 9076 SP2 authentication database. If you have secondary authentication servers, or if you use the procedure for backing up your database that is recommended in the *9076 SP2 Administration Guide*, the database dump can be found in file /var/kerberos/database/slavesave on the primary server host.

For example, if you encountered an authentication failure attempting to **rsh** to node3sw; you could inspect the database entry as follows:

```
# grep "^rcmd node3sw " /var/kerberos/database/slavesave
rcmd node3sw 255 1 2 0 a49bf286 d45c6560 200001010459 199503301502 root admin
#
```

The fifth field on the line is the key version number. In this example, the key version number is two (2).

When the key file, /etc/krb-srvtab, does not exist on the server system, or has the wrong key version, you must re-create the file. When the Control Workstation is a 9076 SP2 authentication server, customizing a 9076 SP2 node will automatically create a new server key file for the node. If customizing the node for this purpose is too disruptive, or if the system whose key file must be replaced is not a 9076 SP2 node, follow the procedures below.

C.1.5 Re-creating Server Key Files

You can use authentication administration commands to re-create an erroneous or missing server key file. Each system with RS/6000 SP authentication service installed has its own unique service key file, containing the encrypted keys for the service instances that are available on the system.

On the Control Workstation and other IBM RISC System/6000 workstations that have client services installed and initialized, the file contains entries for services *rcmd* and *hardmon*. Separate instances of these principals are defined for each network interface on the system, where the instance-name is the short form of the network name. So, for example, on a client system with token-ring and FDDI interfaces named `wksta5t.xyz.abc.com` and `wksta5f.def.abc.com`, the following principals are defined:

```
hardmon.wksta5f
hardmon.wksta5t
rcmd.wksta5f
rcmd.wksta5t
```

The service key files on these systems are created by the **setup_authent** command. On the 9076 SP2 nodes, only the **rcmd** entries are included, and the files are created by the **setup_server** command. They are kept in the **/tftpboot** directory on the Control Workstation, with filenames of the form *hostname*-**new-srvtab**, where *hostname* is the short form of the hostname for each node.

When the Control Workstation is a 9076 SP2 authentication server, these files are retained there only until copied to the node during network boot (or to the node′s boot-install server, if the node boots from another node). A new server key file is generated any time the node is set up for a network boot.

When the Control Workstation is not configured as an authentication server, or when AFS authentication is used, the server key files for the 9076 SP2 nodes are not removed from the */tftpboot* directory on the Control Workstation, once created.

If they are deleted or corrupted, or if you choose to change keys for any reason, the following procedures should be followed to create new key files. In these procedures, "*instance1* ..." are the network names (short form) of all the system′s interfaces, and *hostname* is the short form of the system′s hostname.

C.1.5.1 Replacing an Authentication Server′s File

To re-create the file for a workstation (Control Workstation or other) that is configured as an authentication server, the root user should use the following procedure.

```
# create new key files in the /tmp  directory for each instance
cd /tmp
/usr/kerberos/etc/ext_srvtab -n instance1 ...

# combine the key files into a single file
/bin/cat instance1-new-srvtab ...  >/etc/krb-srvtab
/bin/rm -f instance1-new-srvtab  ...

# make sure the key file is readable only by root
```

```
/bin/chmod 400 /etc/krb-srvtab
```

C.1.5.2 Replacing a Client Workstation′s File

When a workstation is not an authentication server, the root user can use the remote commands to perform the same function on a server system and move the file to the local system. The principal name specified on the **kinit** command must be in the root user′s *.klogin* file on the server:

```
# get a ticket-granting ticket to allow use of rsh/rcp
/usr/kerberos/bin/kinit principal

# create the new key files in /tmp on the server
/usr/lpp/ssp/rcmd/bin/rsh server cd /tmp\; /usr/kerberos/etc/ext_srvtab -n instance1 ...

# copy the files we created to the local /tmp directory
/usr/lpp/ssp/rcmd/bin/rcp server:/tmp/instance1-new-srvtab ... /tmp

# delete the files on the server
/usr/lpp/ssp/rcmd/bin/rsh server /bin/rm -f /tmp/instance1-new-srvtab ...

# combine the local files into a single file
cd /tmp
/bin/cat instance1-new-srvtab ...  >/etc/krb-srvtab
/bin/rm -f instance1-new-srvtab ...

# make sure the key file is readable only by root
/bin/chmod 400 /etc/krb-srvtab
```

C.1.5.3 Replacing a 9076 SP2 Compute Node′s File

The most straightforward way to replace a node′s key file is to customize the node, using the **spbootins** command or SMIT. However, if you prefer, you can instead use procedures similar to the preceding example. If the Control Workstation is a 9076 SP2 authentication server, the root user, logged into the 9076 SP2 node whose server key file needs to be replaced, can use the procedure described above for client systems. Specify the Control Workstation hostname as the *server*. When the Control Workstation is not an authentication server, the server key file must be re-created at the server and then placed in the */tftpboot* directory on the Control Workstation. For this, the root user should be logged into the Control Workstation.

When the authentication server is another workstation running the 9076 SP2 server or another MIT Kerberos Version 4 implementation, use the following procedure:

```
# get a ticket-granting ticket to allow use of rsh/rcp
/usr/kerberos/bin/kinit principal

# create the new key files in /tmp on the server
/usr/lpp/ssp/rcmd/bin/rsh server cd /tmp\; /usr/kerberos/etc/ext_srvtab -n instance1 ...

# copy the files we created to the Control Workstation
/usr/lpp/ssp/rcmd/bin/rcp server:/tmp/instance1-new-srvtab ...  /tmp

# delete the files on the server
/usr/lpp/ssp/rcmd/bin/rsh server /bin/rm -f /tmp/instance1-new-srvtab ...
```

```
# combine the local files into a single file in /tftpboot, whose
# name is hostname-new-srvtab.
cd /tmp
/bin/cat instance1-new-srvtab ...  >/tftpboot/hostname-new-srvtab
/bin/rm -f instance1-new-srvtab ...

# make sure the key file is readable only by root
/bin/chmod 400 /tftpboot/hostname-new-srvtab
```

The new key file can then be installed on the node by either:

- Customizing the node
- Logging into the node as root, and using **rcp** to copy the file from /tftpboot on the Control Workstation to /etc/krb-srvtab
- Using **ftp** to copy the file (not a secure method)

C.1.5.4 Replacing a Server Key File Using AFS Servers

When an AFS authentication server is being used, follow this procedure and be sure you are logged into the Control Workstation as root.

```
# change the service password to a new known but random value
# repeat this step for each instance (i.e. short network name)
kas setpassword -name rcmd.instance1  -new_password any-random-password \
  -kvno 1 -admin_username afs-admin-name -password_for_admin afs-admin-passwd


# use the same principals and passwords to create a srvtab file
/usr/kerberos/bin/ksrvutil -afs -f /tftpboot/hostname-new-srvtab add

# ksrvutil is an interactive program whose sequence of prompts and
# messages appear as follows:
Name: rcmd
Instance: instance1
Realm: <Enter>
Version number: 0
New principal: rcmd.instance1@realm; Version 0
Is this correct? y
Password:
Key successfully added.
Would you like to add another key?  (reply y until all instances have been entered)


# make sure the key file is readable only by root
/bin/chmod 400 /tftpboot/hostname-new-srvtab
```

C.1.6 Problems Using Authenticated Services

When you use any of the Kerberos-authenticated client/server applications to administer or control the 9076 SP2 system, the error messages you receive on authentication failures will vary according to the application. If you are using the System Monitor command-line interface, for example, you might see error messages such as the following if spmon is unable to obtain credentials from the authentication service:

```
0026-706 Cannot obtain service ticket for hardmon.cwksta
Kerberos error code is 76, Kerberos error message is:
     2504-076 Kerberos ticket file was not found.
spmon: 0026-001 Opening session failed.
```

You have no tickets, expired or unexpired, or your KRBTKFILE environment variable specifies a nonexistent file.

```
0026-706 Cannot obtain service ticket for hardmon.cwksta
Kerberos error code is 32, Kerberos error message is:
     2504-032 Kerberos ticket expired.
spmon: 0026-001 Opening session failed.
```

You have tickets that have expired in the ticket cache file specified by your KRBTKFILE environment variable or defaulted.

If application error messages indicate probable authentication failure, use the **klist** command to check your authentication status. The command always displays the current active ticket cache file, whether specified by the KRBTKFILE environment variable or the default file, **/tmp/tkt***uid*.

The following is a listing of the default ticket cache file for the root user (uid 0):

```
# klist
Ticket file:    /tmp/tkt0
Principal:  root.admin@XYZ.ABC.COM

  Issued           Expires          Principal
Nov 12 16:26:11  Dec 12 16:26:11  krbtgt.XYZ.ABC.COM@XYZ.ABC.COM
Nov 12 16:26:46  Dec 12 16:26:46  hardmon.cwksta@XYZ.ABC.COM
Nov 12 16:45:15  Dec 12 16:45:15  rcmd.cwksta@XYZ.ABC.COM
#
```

The second line shows the Kerberos principal acting as client, to whom the tickets belong. This is the user principal you supplied to the **kinit** command, or the **rcmd**.*instance* service principal used by **rcmdtgt**. The list of tickets always begins with the ticket-granting ticket. The others are service tickets; in this case for the System Monitor service on the Control Workstation (**hardmon**) and the 9076 SP2 Remote Command service also on the Control Workstation (**rcmd**).

C.1.7 Problems with the Authentication Server Daemons

Other more or less common errors may involve the inability to communicate with the Kerberos authentication and administrative servers. If this is the case, check for network problems or interface and routing problems. You can also check the state of the server daemons themselves on systems running the 9076 SP2 authentication server. The primary server system should have *kerberos* and *kadmind* daemons running (from init).

The following shows the /etc/inittab entries for the Kerberos daemons on a primary server:

```
#lsitab kerb
kerb:2:respawn:/usr/kerberos/etc/kerberos
# lsitab kadm
kadm:2:respawn:/usr/kerberos/etc/kadmind -n
#
```

A secondary server system should have *kerberos* and *kpropd* daemons running (from init).

The following shows the /etc/inittab entries for the Kerberos daemons on a secondary server:

```
# lsitab kerb
kerb:2:respawn:/usr/kerberos/etc/kerberos -s
# lsitab kpropd
kpropd:2:respawn:/usr/kerberos/etc/run-kpropd
#
```

C.1.8 Authentication Daemon Log Files

Each Kerberos daemon program records errors and some status in a log file in the /var/adm/SPlogs/kerberos directory. Check these files if you suspect one or more of the daemons have terminated. They are designed to hang indefinitely or for several minutes, depending in some cases on command-line options, to prevent problems caused by endless respawning of a failing daemon.

The following is an example of the log file created by the kerberos daemon:

```
# cat /var/adm/SPlogs/kerberos/kerberos.log
13-Oct-94 07:53:07 Kerberos started, PID=8012

13-Oct-94 07:53:07 kerberos: 2503-604 Cannot verify master key.
13-Oct-94 07:53:07 Kerberos will pause so as not to loop init
13-Oct-94 08:47:33 Kerberos started, PID=13243
#
```

The following is an example of the log file created by the kadmind daemon:

```
# cat /var/adm/SPlogs/kerberos/admin_server.syslog
13-Oct-94 08:42:16 Kerberos admin server started, PID=9831
13-Oct-94 08:42:16 kadmind: 2503-101 error: 2504-318 Could not verify master key

13-Oct-94 08:42:16 Shutting down admin server
13-Oct-94 08:47:33 Kerberos admin server started, PID=13759
#
```

The following is an example of the log file created by the kpropd daemon:

```
# cat /var/adm/SPlogs/kerberos/kpropd.log
13-Oct-94 09:27:29

***** kpropd started *****
13-Oct-94 09:27:29 Established socket
13-Oct-94 09:27:31 Connection from cwksta.xyz.abc.com, 129.49.100.41
13-Oct-94 09:27:31 kpropd: Connection from rcmd.cwksta@XYZ.ABC.COM
13-Oct-94 09:27:31 File received.
13-Oct-94 09:27:31 Temp file renamed to /tmp/sdump10560
13-Oct-94 09:27:32

***** kpropd started *****
13-Oct-94 09:27:32 Established socket
13-Oct-94 10:19:09 Connection from cwksta.xyz.abc.com, 129.49.100.41
13-Oct-94 10:19:09 kpropd: Connection from rcmd.cwksta@XYZ.ABC.COM
13-Oct-94 10:19:09 File received.
13-Oct-94 10:19:09 Temp file renamed to /var/kerberos/database/slavedb
14-Oct-94 07:36:41 Connection from cwksta.xyz.abc.com, 129.49.100.41
14-Oct-94 07:36:41 kpropd: Connection from rcmd.cwksta@XYZ.ABC.COM
```

```
14-Oct-94 07:36:41 File received.
14-Oct-94 07:36:41 Temp file renamed to /var/kerberos/database/slavedb
#
```

Appendix D. README Files for Tcl and TclX

This chapter contains the descriptions of the command languages used for writing functions and procedures in sysctl, Tcl and TclX. The text of these sections is an exact copy of the README files contained in the public domain parts of the PSSP package. We have added them to this Redbook for your convenience as they are hard to find.

D.1 Tcl

by John Ousterhout University of California at Berkeley ouster@cs.berkeley.edu

Introduction This directory contains the sources and documentation for Tcl, an embeddable tool command language. The information here corresponds to release 7.3.

Documentation The best way to get started with Tcl is to read the draft of my upcoming book on Tcl and Tk, which can be retrieved using anonymous FTP from the directory ″ucb/tcl″ on ftp.cs.berkeley.edu. Part I of the book provides an introduction to writing Tcl scripts and Part III describes how to write C code that uses the Tcl C library procedures.

The ″doc″ subdirectory in this release contains a complete set of manual entries for Tcl. Files with extension ″.1″ are for programs (for example, tclsh.1); files with extension ″.3″ are for C library procedures; and files with extension ″.n″ describe Tcl commands. The file ″doc/Tcl.n″ gives a quick summary of the Tcl language syntax. To print any of the man pages, cd to the ″doc″ directory and invoke your favorite variant of troff using the normal -man macros, for example

```
ditroff -man Tcl.n
```

to print Tcl.n. If Tcl has been installed correctly and your ″man″ program supports it, you should be able to access the Tcl manual entries using the normal ″man″ mechanisms, such as

```
man Tcl
```

Compiling and installing Tcl This release should compile and run ″out of the box″ on any UNIX-like system that approximates POSIX, BSD, or System V. I know that it runs on workstations from Sun, DEC, H-P, IBM, and Silicon Graphics, and on PC′s running SCO UNIX and Xenix. To compile Tcl, do the following:

- Type ″./configure″ in this directory. This runs a configuration script created by GNU autoconf which configures Tcl for your system and creates a Makefile. The configure script allows you to customize the Tcl configuration for your site; for details on how you can do this, see the file ″configure.info″
- Type ″make″. This will create a library archive called ″libtcl.a″ and an interpreter application called ″tclsh″ that allows you to type Tcl commands interactively or execute script files.

- If the make fails then you′ll have to personalize the Makefile for your site or possibly modify the distribution in other ways. First check the file ″porting.notes″ to see if there are hints for compiling on your system. If you need to modify Makefile, there are comments at the beginning of it that describe the things you might want to change and how to change them.
- Type ″make install″ to install Tcl binaries and script files in standard places. You′ll need write permission on /usr/loc al to do this. See the Makefile for details on where things get installed.
- At this point you can play with Tcl by invoking the ″tclsh″ program and typing Tcl command. However, if you haven′t installed Tcl then you′ll first need to set your TCL_LIBRARY variable to hold the full path name of the ″library″ subdirectory.

  If you have trouble compiling Tcl, I′d suggest looking at the file ″porting.notes″. It contains information that people have sent me about changes they had to make to compile Tcl in various environments. I make no guarantees that this information is accurate, complete, or up-to-date, but you may find it useful. If you get Tcl running on a new configuration, I′d be happy to receive new information to add to ″porting.notes″. I′m also interested in hearing how to change the configuration setup so that Tcl compiles on additional platforms ″out of the box″.

Test suite There is a relatively complete test suite for all of the Tcl core in the subdirectory ″tests″. To use it just type ″make test″ in this directory. You should then see a printout of the test files processed. If any errors occur, you′ll see a much more substantial printout for each error. See the README file in the ″tests″ directory for more information on the test suite.

Summary of changes in recent releases Tcl 7.3 is a minor release that includes only one change relative to Tcl 7.1 (it fixes a portability problem that prevented tclMain.c from compiling on some machines due to a missing R_OK definition). Tcl 7.3 should be completely compatible with Tcl 7.1 and Tcl 7.0.

Tcl 7.2 was a mistake; it was withdrawn shortly after it was released.

Tcl 7.1 is a minor release that consists almost entirely of bug fixes. The only feature change is to allow no arguments in invocations of ″list″ and ″concat″. 7.1 should be completely compatible with 7.0.

Tcl 7.0 is a major new release that includes several new features and a few incompatible changes. For a complete list of all changes to Tcl in chronological order, see the file ″changes″. Those changes likely to cause compatibility problems with existing C code or Tcl scripts are specially marked. The most important changes are summarized below.

Tcl configuration and installation has improved in several ways:

1. GNU autoconf is now used for configuring Tcl prior to compilation.

2. The ″tclTest″ program no longer exists. It has been replaced by ″tclsh″, which is a true shell-like program based around Tcl (t clTest didn′t really work very well as a shell). There′s a new program ″tcltest″ which is the same as ″tclsh″ except that it includes a few extra Tcl commands for testing purposes.

3. A new procedure Tcl_AppInit has been added to separate all of the application-specific initialization from the Tcl main program. This should make it easier to build new Tcl applications that include extra packages.

4. There are now separate manual entries for each of the built-in commands. The manual entry ″Tcl.n″, which used to describe all of the built-ins plus many other things, now contains a terse but complete description of the Tcl language syntax.

Here is a list of all incompatibilities that affect Tcl scripts:

1. There have been several changes to backslash processing:

   - Unknown backslash sequences such as ″*″ are now replaced with the following character (such as ″*″); Tcl used to treat the backslash as an ordinary character in these cases, so both the backslash and the following character would be passed through.

   - Backslash-newline now eats up any white space after the newline, replacing the whole sequence with a single space character. Tcl used to just remove the backslash and newline.

   - The obsolete sequences Cx, Mx, CMx, and e no loner get special treatment.

   - The ″format″ command no longer does backslash processing on its input string.

   You can invoke the shell command below to locate backslash uses that may potentially behave differently under Tcl 7.0. This command will print all of the lines from the script files ″*.tcl″ that may not work correctly under Tcl 7.0:

   ```
   egrep ′($) |
   (\\[^][bfnrtv\0-9{}$ ;″])′
   *.tcl
   ```

   In some cases the command may print lines that are actually OK.

2. The ″glob″ command now returns only the names of files that actually exist, and it only returns names ending in ″/″ for directories.

3. When Tcl prints floating-point numbers (e.g. in the ″expr″ command) it ensures that the numbers contain a ″.″ or ″e″ so that they don′t look like integers.

4. The ″regsub″ command now overwrites its result variable in all cases. If there is no match, then the source string is copied to the result.

5. The ″exec″, ″glob″, ″regexp″, and ″regsub″ commands now include a ″--″ switch; if the first non-switch argument starts with a ″-″ then there must be a ″--″ switch or the non-switch argument will be treated as a switch.

6. The keyword ″UNIX″ in the variable ″errorCode″ has been changed to ″POSIX″.

7. The ″format″ and ″scan″ commands no longer support capitalized conversion specifiers such as ″%D″ that aren′t support ed by ANSI sprintf and sscanf.

Here is a list of all of the incompatibilities that affect C code that uses the Tcl library procedures. If you use an ANSI C compiler then any potential problems will be detected when you compile your code: if your code compiles cleanly then you don′t need to worry about anything.

1. Tcl_TildeString now takes a dynamic string as an argument, which is used to hold the result.

2. tclHash.h has been eliminated; its contents are now in tcl.h.

3. The Tcl_History command has been eliminated: the ″history″ command is now automatically part of the interpreter.

4. The Tcl_Fork and Tcl_WaitPids procedures have been deleted (just use fork and waitpid instead).

5. The ″flags″ and ″termPtr″ arguments to Tcl_Eval have been eliminated, as has the ″noSep″ argument to Tcl_AppendElement and the TCL_NO_SPACE flag for Tcl_SetVar and Tcl_SetVar2.

6. The Tcl_CmdBuf structure has been eliminated, along with the procedures Tcl_CreateCmdBuf, Tcl_DeleteCmdBuf, and Tcl_AssembleCmd. Use dynamic strings instead.

7. Tcl_UnsetVar and Tcl_UnsetVar2 now return TCL_OK or TCL_ERROR instead of 0 or -1.

8. Tcl_UnixError has been renamed to Tcl_PosixError.

9. Tcl no longer redefines the library procedures ″setenv″, ″putenv″, and ″unsetenv″ by default. You have to set up special configuration in the Makefile if you want this.

Below is a sampler of the most important new features in Tcl 7.0. Refer to the ″changes″ file for a complete list.

1. The ″expr″ command supports transcendental and other math functions, plus it allows you to type expressions in multiple arguments. Its numerics have also been improved in several ways (e.g. support for NaN).

2. The ″format″ command now supports XPG3 %n$ conversion specifiers.

3. The ″exec″ command supports many new kinds of redirection such as >> and >&, plus it allows you to leave out the space between operators like < and the file name. For processes put into the background, ″exec″ returns a list of process ids.
4. The ″lsearch″ command now supports regular expressions and exact matching.
5. The ″lsort″ command has several new switches to control the sorting process (e.g. numerical sort, user-provided sort function, reverse sort, etc.).
6. There′s a new command ″pid″ that can be used to return the current process ids or the process ids from an open file that refers to a pipeline.
7. There′s a new command ″switch″ that should now be used instead of ″case″. It supports regular expressions and exact matches, and also uses single patterns instead of pattern lists. ″Case″ is now deprecated, although it′s been retained for compatibility.
8. A new dynamic string library has been added to make it easier to build up strings and lists of arbitrary length. See the manual entry ″DString.3″.
9. Variable handling has been improved in several ways: you can now use whole-array traces to create variables on demand, you can delete variables during traces, you can upvar to array elements, and you can retarget an upvar variable to stop through a sequence of variables. Also, there′s a new library procedure Tcl_LinkVar that can be used to associate a C variable with a Tcl variable and keep them in sync.
10. New library procedures Tcl_SetCommandInfo and Tcl_GetCommandInfo allow you to set and get the clientData and callback procedure for a command.
11. Added ″-errorinfo″ and ″-errorcode″ options to ″return″ command; they allow much better error handling.
12. Made prompts in tclsh user-settable via ″tcl_prompt1″ and ″tcl_prompt2″ variables.
13. Added low-level support that is needed to handle signals: see Tcl_AsyncCreate, etc.

Tcl newsgroup There is a network news group ″comp.lang.tcl″ intended for the exchange of information about Tcl, Tk, and related applications. Feel free to use the newsgroup both for general information questions and for bug reports. I read the newsgroup and will attempt to fix bugs and problems reported to it.

Tcl contributed archive Many people have created exciting packages and applications based on Tcl and made them freely available to the Tcl community. An archive of these contributions is kept on the machine harbor.ecn.purdue.edu. You can access the archive using anonymous FTP; the Tcl contributed archive is in the directory ″pub/tcl″. The archive also contains an FAQ (″frequently asked questions″) document that provides solutions to problems that are commonly encountered by TCL newcomers.

Support and bug fixes I′m very interested in receiving bug reports and suggestions for improvements. Bugs usually get fixed quickly (particularly if they are serious), but enhancements may take a while and may not happen at all unless there is widespread support for them (I′m trying to slow the rate at which Tcl turns into a kitchen sink). It′s almost impossible to make incompatible changes to Tcl at this point.

The Tcl community is too large for me to provide much individual support for users. If you need help I suggest that you post questions to comp.lang.tcl. I read the newsgroup and will attempt to answer esoteric questions for which no-one else is likely to know the answer. In addition, Tcl support and training are available commercially from NeoSoft. For more information, send e-mail to ″info@neosoft.com″.

Tcl release organization Each Tcl release is identified by two numbers separated by a dot, e.g. 6.7 or 7.0. If a new release contains changes that are likely to break existing C code or Tcl scripts then the major release number increments and the minor number resets to zero: 6.0, 7.0, etc. If a new release contains only bug fixes and compatible changes, then the minor number increments without changing the major number, e.g. 7.1, 7.2, etc. If you have C code or Tcl scripts that work with release X.Y, then they should also work with any release X.Z as long as Z > Y.

Beta releases have an additional suffix of the form bx. For example, Tcl 7.0b1 is the first beta release of Tcl version 7.0, Tcl 7.0b2 is the second beta release, and so on. A beta release is an initial version of a new release, used to fix bugs and bad features before declaring the release stable. Each new release will be preceded by one or more beta releases. I hope that lots of people will try out the beta releases and report problems back to me. I′ll make new beta releases to fix the problems, until eventually there is a beta release that appears to be stable. Once this occurs I′ll remove the beta suffix so that the last beta release becomes the official release.

If a new release contains incompatibilities (e.g. 7.0) then I can′t promise to maintain compatibility among its beta releases. For example, release 7.0b2 may not be backward compatible with 7.0b1. I′ll try to minimize incompatibilities between beta releases, but if a major problem turns up then I′ll fix it even if it introduces an incompatibility. Once the official release is made then there won′t be any more incompatibilities until the next release with a new major version number.

Compiling on non-UNIX systems The Tcl features that depend on system calls peculiar to UNIX (stat, fork, exec, times, etc.) are now separate from the main body of Tcl, which only requires a few generic library procedures such as malloc and strcpy. Thus it should be relatively easy to compile Tcl for non-UNIX machines such as MACs and DOS PC′s, although a number of UNIX-specific commands will be absent (e.g. exec, time, and glob). See the comments at the top of Makefile for information on how to compile without the UNIX features.

D.2 Extended Tcl

*** README file for Extended Tcl 7.3a ***

Extended TCL Extended Tcl (TclX), is a set of extensions to Tcl 7.3, the Tool Command Language invented by Dr. John Ousterhout of the University of California at Berkeley. Tcl is a powerful, yet simple embeddable programming language. Extended Tcl is oriented towards Unix system programming tasks, with many additional interfaces to the Unix operating system, It is upwardly compatible with Tcl 7.3. You take the Extended Tcl package, add it to Tcl 7.3, and from that you get Extended Tcl. (Berkeley Tcl is not included in this distribution, obtain it from ftp.cs.berkeley.edu).

Support is also included for building a Tk 3.6 wish shell (wishx) with the Extended Tcl command set.

While this TclX distribution is tested with Tcl 7.3 and Tk 3.6, it will probably work with new versions of Tcl & Tk with little or no changes.

Extended Tcl was designed and implemented by Karl Lehenbauer (karl@NeoSoft.com) and Mark Diekhans (markd@Grizzly.com), with help in the early stages from Peter da Silva (peter@NeoSoft.com).

As with Berkeley Tcl, all of Extended Tcl is freely redistributable, including for commercial use and resale.

Please read the file INSTALL carefully before building and installing Extended Tcl.

Features added by Extended TCL Here is a summary of the features added by Extended Tcl. For more details on the commands and functionality provided by Extended Tcl, see the manual page man/TclX.man.

- A shell, which provides an environment for developing and executing Tcl programs.
- Advanced Tcl code library facility that is compatible with standard Tcl auto-loading.
- General purpose commands which define new programming constructs, debugging and profiling facilities.
- Unix access commands provide access to many Unix system calls, including process management.
- File I/O commands provide added facilities for accessing and manipulating open files.
- File scanning facility that provides awk-like functionality.
- Extended list manipulation commands.
- Keyed lists, a type of list that provides functionality similar to C structures.

- Extended string and character manipulation commands.
- Time and date manipulation and conversion commands.
- A simple command for accessing TCP/IP based servers.
- X/PG based internationalization commands.

On-Line Help There is a help system included with Extended Tcl. It contains some documentation on every command in Berkeley Tcl, Extended Tcl and Tk. You can invoke it interactively from within Extended Tcl by typing ″help″.

Once you bring Tcl up and have gotten it to pass all the tests, try typing ″help help″ to learn how to use help.

There is also a Tk based help program ″tclhelp″.

Manual Pages Man pages in nroff/troff format are provided for all of Tcl and the extensions in the directory tclX7.3a*/man. Start with the TclX.man manual.

Extended TCL Version Naming Extended Tcl takes its version number from the corresponding version of Berkeley (Ousterhout) Tcl upon which it is based, with the addition of a trailing letter in case there are multiple releases of Extended Tcl within a single release of Berkeley Tcl. This release is TclX 7.3a.

Interfacing TCL and C+ + C++ programmers can include the file ″tcl++.h″ to dfine C++ classes that can be used to access a Tcl interpreter. This is based on Tcl C++ classes originally developed by Parag Patel. The files

```
src/tclXmain.c
tksrc/tkXmain.c
```

will compile under both C and C++. If your have a C++ compiler that requires the main to be compiled with C++ (g++ does not have this restriction), use these files renamed to have the correct suffix for C++ (usually .C).

Package Libraries Package Libraries are a Tcl source code management tool included in this release. Package libraries allow you to group Tcl procedures into logical bundles and create single files, libraries, that contain multiple packages. The package code provides a low-overhead means of automatically demand-loading a package on the first attempt to execute one of the procedures defined within it. As such, package libraries provide a mechanism to accommodate the creation of Tcl applications and libraries of a substantial size. The TclX library mechanism is a super-set of the mechanism provided with standard Tcl.

Linking TCL Applications To build a TclX based application containing C code, start with either:

```
       tclmaster/src/tclXAppInit.c for just TclX
or
       tkmaster/src/tkXAppInit.c for TclX & Tk
```

and add your application initialization. There are comments in the code to guide you. The ″main″ object file is already in the libraries, all you have to do is link a customized tclXAppInit.o or tkXAppInit.o file with your application and the Tcl libraries. For example:

```
cc tclXAppInit.o mystuff.a libtclx.a libtcl.a ${SYSLIBS} -o myapp

cc tkXAppInit.o mystuff.a libtkx.a libtk.a libtclx.a libtcl.a   \
   ${SYSLIBS} -o myapp
```

Each of those directories has a file SYSLIBS that contains the system libraries that the TclX configure script thinks you should use for linking programs. These file are in a format that can be included from a make file.

IMPORTANT NOTE: libtclx.a must be specified on the link command line before libtcl.a. If the order is reversed, incorrect command line parsing code will be used.

Support for Extended TCL We are committed to providing continuing support for Extended Tcl. Please send questions, bug reports, and bug fixes to:

```
tcl-project@NeoSoft.com
```

Compatibility with TclX 6.5c We have attempted to main backwards-compatibility with older versions of TclX. A few changes were made to enhance usability or fix problems that have introduced a few incompatibilities, these are listed below. Remember that multiple versions of Tcl may be installed on a system using the Tcl default file, so you don′t have to convert everything at once. See the CHANGES file for full details. These are the major incompatible changes:

- TclX auto loading has been made compatible with standard Tcl. The path is now found in the ″auto_path″ variable instead of ″TCLPATH″. The ″demand_load″ function has been renamed ″auto_load″ and ″load″ has been renamed ″auto_load_file″. The ″autoload″ procedure no longer exists.
- The variable programName was changed to argv0 for compatibility with Tcl.
- Removed outdated execvp proc, use execl directly. Removed ″flush″ option on ″cmdtrace″. Output lines are now flushed after each command is traced.
- Changed ″signal trap″ to edit the command, replacing %S with the signal name rather than storing in the global variable ″signalReceived″. This fixes bugs with signals received in a signal handler.

Where to get it Tcl & Tk is available via anonymous ftp from:

```
   ftp.cs.berkeley.edu:/ucb/tcl/[tcl7.3.tar.Z tk3.6.tar.Z]
or
   ftp.neosoft.com:/pub/tcl/distrib/[tcl7.3.tar.gz tk3.6.tar.gz]
or
   ftp.uu.net:languages/tcl/[tcl7.3.tar.Z tk3.6.tar.Z]
```

Extended Tcl 7.3a can be downloaded by anonymous FTP from:

```
   ftp.neosoft.com:/pub/tcl/distrib/tclX7.3a.tar.gz
or
   harbor.ecn.purdue.edu:tcl/extensions/tclX7.3a.tar.Z
```

A contributed sources archive resides on harbor.ecn.purdue.edu and is mirrored on ftp.ibp.fr for our friends in Europe. An Frequently Asked Questions (FAQ) document exists in this archive.

Remember to mail Extended Tcl problems and questions to tcl-project@NeoSoft.com not Dr. John Ousterhout.

Thanks A big thanks to all of the Extended Tcl users from all over the world who have helped us debug problems and given us valuable suggestions. A special thanks to John Ousterhout for Tcl, Tk and all the support he has given us.

NEOSOFT TCL Consulting NeoSoft, co-developers of Extended Tcl, provides commercial Tcl releases, support, training and consulting. NeoSoft can be reached by sending electronic mail to info@NeoSoft.com or by phoning +1 713 684 5969.

Appendix E. Sysctl Debug Output

This appendix shows the output from debugging a sysctl environment. The command used to get this information is:

```
/usr/lpp/ssp/bin/sysctl -sd -l /tmp/sysctl.log > /tmp/sysctl.log 2>&1
```

Note: Before issuing the command, configure the sysctl entry in /etc/inittab to *off*.

The output consists of three parts:

- The listing of the /etc/sysctl.conf file, with three changes to the default configuration.
- The listing of the /etc/remcmds.cmds file, containing the rcmds callback procedure and the rexec and rlog commands.
- The debug and security/audit information logged by the sysctl daemon. Note the authentication configuration and the callback configuration.

The /etc/sysctl.conf File

```
# This points to where sysctl resides
create var buildTop /usr/lpp/ssp AUTH

# Set up the environment for the rcmd sample programs
set env(PATH) /bin:/usr/bin:/usr/ucb:/etc:/usr/etc
set env(EDITOR) /usr/bin/vi

# Include the help commands, but override the default helpPath
create class help $buildTop/help/help.cmds
help:setPath $buildTop/help

# Include system information commands
create class sys $buildTop/samples/sysctl/sys.cmds

# Include process manipulation commands
include $buildTop/samples/sysctl/process.cmds

# Include file system commands
include $buildTop/samples/sysctl/filesys.cmds

# Include more file system commands (pdf and pfck)
include $buildTop/sysctl/bin/pdfpfck.cmds

# Include pfps commands
include $buildTop/sysctl/bin/pfps.cmds

create var STARTTIME [exec /bin/date] NONE

#setauth -cmd exec {ACL /etc/sysctl.exec.acl}
#setauth -cmd svcconnect NONE

create class rcmds /etc/remcmds.cmds
```

Figure 99. The /etc/sysctl.conf file.

The /etc/remcmds.cmds File

```
#
# rcmds Authorization Callback, just check Authentication
#
create proc rcmd_auth {cmdName} SYSTEM {
    global SCPRINCIPAL

    if [catch AUTH] {
      svclog "Denying \"$cmdName\" access for user $SCPRINCIPAL"
      error "Authorization denied"
    } else {
      svclog "Authorizing \"$cmdName\" access for user $SCPRINCIPAL"
      return "Authorization ok"
    }
}

#
# REXEC, backend for the rsh command
#
create proc rexec {args} {CLASS:rcmd_auth rexec} {
    global SCUSER

    # Log, for good measure
    svclog "rexec: user=$SCUSER, command=\"$args\""

    # Assume the remote user's identity
    id user $SCUSER

# Issue the command and return the exit status
system [join $args]
}
rename rcmds:rexec rexec
#
# RLOG a secure way of connecting to AIX on a remote host.
#
create proc rlog {args} {CLASS:rcmd_auth rlog} {
    global SCUSER

    # Log, for good measure
    svclog "rlog: user=$SCUSER, command=\"ksh -i\""

    # Assume the remote user's identity
    id user $SCUSER

    # Issue the command and return the exit status
    system "ksh -i"
}
rename rcmds:rlog rlog
```

Figure 100. The /etc/remcmds.cmds file

The Generated sysctld.log Output

```
<00000000> Detected licensed environment - continuing initialization.
<00000000> LC_MESSAGES: En_US
<00000000> Server successfully opened message catalog...
AUDIT[sec]: Setting callback for command "append" => {AUTH}
AUDIT[sec]: Setting callback for command "array" => {AUTH}
AUDIT[sec]: Setting callback for command "break" => {AUTH}
AUDIT[sec]: Setting callback for command "catopen" => {ACL}
AUDIT[sec]: Setting callback for command "catclose" => {ACL}
AUDIT[sec]: Setting callback for command "catgets" => {ACL}
AUDIT[sec]: Setting callback for command "case" => {AUTH}
AUDIT[sec]: Setting callback for command "catch" => {AUTH}
AUDIT[sec]: Setting callback for command "cd" => {ACL}
AUDIT[sec]: Setting callback for command "close" => {ACL}
AUDIT[sec]: Setting callback for command "cmdtrace" => {ACL}
AUDIT[sec]: Setting callback for command "concat" => {AUTH}
AUDIT[sec]: Setting callback for command "continue" => {AUTH}
AUDIT[sec]: Setting callback for command "eof" => {ACL}
AUDIT[sec]: Setting callback for command "error" => {AUTH}
AUDIT[sec]: Setting callback for command "eval" => {ACL}
AUDIT[sec]: Setting callback for command "exec" => {ACL}
AUDIT[sec]: Setting callback for command "exit" => {AUTH}
AUDIT[sec]: Setting callback for command "expr" => {AUTH}
AUDIT[sec]: Setting callback for command "file" => {ACL}
AUDIT[sec]: Setting callback for command "flush" => {ACL}
AUDIT[sec]: Setting callback for command "for" => {ACL}
AUDIT[sec]: Setting callback for command "foreach" => {ACL}
AUDIT[sec]: Setting callback for command "format" => {AUTH}
AUDIT[sec]: Setting callback for command "gets" => {ACL}
AUDIT[sec]: Setting callback for command "glob" => {ACL}
AUDIT[sec]: Setting callback for command "global" => {AUTH}
AUDIT[sec]: Setting callback for command "history" => {AUTH}
AUDIT[sec]: Setting callback for command "if" => {AUTH}
AUDIT[sec]: Setting callback for command "incr" => {AUTH}
AUDIT[sec]: Setting callback for command "info" => {ACL}
AUDIT[sec]: Setting callback for command "join" => {AUTH}
AUDIT[sec]: Setting callback for command "lappend" => {AUTH}
AUDIT[sec]: Setting callback for command "lindex" => {AUTH}
AUDIT[sec]: Setting callback for command "linsert" => {AUTH}
AUDIT[sec]: Setting callback for command "list" => {AUTH}
AUDIT[sec]: Setting callback for command "llength" => {AUTH}
AUDIT[sec]: Setting callback for command "lrange" => {AUTH}
AUDIT[sec]: Setting callback for command "lreplace" => {AUTH}
AUDIT[sec]: Setting callback for command "lsearch" => {AUTH}
AUDIT[sec]: Setting callback for command "lsort" => {AUTH}
AUDIT[sec]: Setting callback for command "open" => {ACL}
AUDIT[sec]: Setting callback for command "pid" => {ACL}
AUDIT[sec]: Setting callback for command "proc" => {SYSTEM}
AUDIT[sec]: Setting callback for command "puts" => {ACL}
AUDIT[sec]: Setting callback for command "pwd" => {ACL}
AUDIT[sec]: Setting callback for command "read" => {ACL}
AUDIT[sec]: Setting callback for command "regexp" => {AUTH}
AUDIT[sec]: Setting callback for command "regsub" => {AUTH}
AUDIT[sec]: Setting callback for command "rename" => {SYSTEM}
AUDIT[sec]: Setting callback for command "return" => {NONE}
AUDIT[sec]: Setting callback for command "scan" => {AUTH}
AUDIT[sec]: Setting callback for command "seek" => {ACL}
AUDIT[sec]: Setting callback for command "set" => {AUTH}
AUDIT[sec]: Setting callback for command "source" => {SYSTEM}
AUDIT[sec]: Setting callback for command "split" => {AUTH}
```

```
AUDIT[sec]: Setting callback for command ″string″ => {AUTH}
AUDIT[sec]: Setting callback for command ″switch″ => {AUTH}
AUDIT[sec]: Setting callback for command ″tell″ => {ACL}
AUDIT[sec]: Setting callback for command ″time″ => {ACL}
AUDIT[sec]: Setting callback for command ″trace″ => {ACL}
AUDIT[sec]: Setting callback for command ″unset″ => {AUTH}
AUDIT[sec]: Setting callback for command ″uplevel″ => {SYSTEM}
AUDIT[sec]: Setting callback for command ″upvar″ => {ACL}
AUDIT[sec]: Setting callback for command ″while″ => {ACL}
AUDIT[sec]: Setting callback for command ″server_open″ => {ACL}
AUDIT[sec]: Setting callback for command ″commandloop″ => {ACL}
AUDIT[sec]: Setting callback for command ″loop″ => {ACL}
AUDIT[sec]: Setting callback for command ″sync″ => {ACL}
AUDIT[sec]: Setting callback for command ″lassign″ => {AUTH}
AUDIT[sec]: Setting callback for command ″scanmatch″ => {ACL}
AUDIT[sec]: Setting callback for command ″readdir″ => {ACL}
AUDIT[sec]: Setting callback for command ″convertclock″ => {AUTH}
AUDIT[sec]: Setting callback for command ″nice″ => {ACL}
AUDIT[sec]: Setting callback for command ″scancontext″ => {ACL}
AUDIT[sec]: Setting callback for command ″min″ => {ACL}
AUDIT[sec]: Setting callback for command ″max″ => {ACL}
AUDIT[sec]: Setting callback for command ″chroot″ => {SYSTEM}
AUDIT[sec]: Setting callback for command ″copyfile″ => {ACL}
AUDIT[sec]: Setting callback for command ″lgets″ => {ACL}
AUDIT[sec]: Setting callback for command ″pipe″ => {ACL}
AUDIT[sec]: Setting callback for command ″fcntl″ => {ACL}
AUDIT[sec]: Setting callback for command ″select″ => {ACL}
AUDIT[sec]: Setting callback for command ″bsearch″ => {ACL}
AUDIT[sec]: Setting callback for command ″profile″ => {ACL}
AUDIT[sec]: Setting callback for command ″scanfile″ => {ACL}
AUDIT[sec]: Setting callback for command ″signal″ => {ACL}
AUDIT[sec]: Setting callback for command ″infox″ => {ACL}
AUDIT[sec]: Setting callback for command ″alarm″ => {ACL}
AUDIT[sec]: Setting callback for command ″dup″ => {ACL}
AUDIT[sec]: Setting callback for command ″times″ => {ACL}
AUDIT[sec]: Setting callback for command ″cequal″ => {AUTH}
AUDIT[sec]: Setting callback for command ″cexpand″ => {AUTH}
AUDIT[sec]: Setting callback for command ″cindex″ => {AUTH}
AUDIT[sec]: Setting callback for command ″clength″ => {AUTH}
AUDIT[sec]: Setting callback for command ″crange″ => {AUTH}
AUDIT[sec]: Setting callback for command ″csubstr″ => {AUTH}
AUDIT[sec]: Setting callback for command ″ctoken″ => {AUTH}
AUDIT[sec]: Setting callback for command ″ctype″ => {AUTH}
AUDIT[sec]: Setting callback for command ″chgrp″ => {ACL}
AUDIT[sec]: Setting callback for command ″chmod″ => {ACL}
AUDIT[sec]: Setting callback for command ″chown″ => {ACL}
AUDIT[sec]: Setting callback for command ″echo″ => {AUTH}
AUDIT[sec]: Setting callback for command ″execl″ => {ACL}
AUDIT[sec]: Setting callback for command ″flock″ => {ACL}
AUDIT[sec]: Setting callback for command ″fmtclock″ => {AUTH}
AUDIT[sec]: Setting callback for command ″fork″ => {ACL}
AUDIT[sec]: Setting callback for command ″frename″ => {ACL}
AUDIT[sec]: Setting callback for command ″fstat″ => {ACL}
AUDIT[sec]: Setting callback for command ″funlock″ => {ACL}
AUDIT[sec]: Setting callback for command ″getclock″ => {AUTH}
AUDIT[sec]: Setting callback for command ″id″ => {ACL}
AUDIT[sec]: Setting callback for command ″keyldel″ => {AUTH}
AUDIT[sec]: Setting callback for command ″keylget″ => {AUTH}
AUDIT[sec]: Setting callback for command ″keylkeys″ => {AUTH}
```

```
AUDIT[sec]: Setting callback for command ″keylset″ => {AUTH}
AUDIT[sec]: Setting callback for command ″kill″ => {ACL}
AUDIT[sec]: Setting callback for command ″lempty″ => {ACL}
AUDIT[sec]: Setting callback for command ″link″ => {ACL}
AUDIT[sec]: Setting callback for command ″lmatch″ => {AUTH}
AUDIT[sec]: Setting callback for command ″lvarcat″ => {AUTH}
AUDIT[sec]: Setting callback for command ″lvarpop″ => {AUTH}
AUDIT[sec]: Setting callback for command ″lvarpush″ => {AUTH}
AUDIT[sec]: Setting callback for command ″mkdir″ => {ACL}
AUDIT[sec]: Setting callback for command ″random″ => {AUTH}
AUDIT[sec]: Setting callback for command ″replicate″ => {AUTH}
AUDIT[sec]: Setting callback for command ″rmdir″ => {ACL}
AUDIT[sec]: Setting callback for command ″sleep″ => {ACL}
AUDIT[sec]: Setting callback for command ″system″ => {ACL}
AUDIT[sec]: Setting callback for command ″translit″ => {AUTH}
AUDIT[sec]: Setting callback for command ″umask″ => {ACL}
AUDIT[sec]: Setting callback for command ″unlink″ => {ACL}
AUDIT[sec]: Setting callback for command ″wait″ => {ACL}
AUDIT[cmd]: Creating command ″quit″, callBack = {AUTH}
AUDIT[sec]: Setting callback for command ″quit″ => {AUTH}
AUDIT[cmd]: Creating command ″safeexec″, callBack = {ACL}
AUDIT[sec]: Setting callback for command ″safeexec″ => {ACL}
AUDIT[cmd]: Creating command ″safesystem″, callBack = {ACL}
AUDIT[sec]: Setting callback for command ″safesystem″ => {ACL}
AUDIT[cmd]: Creating command ″safeargs″, callBack = {ACL}
AUDIT[sec]: Setting callback for command ″safeargs″ => {ACL}
AUDIT[cmd]: Creating command ″listfs″, callBack = {ACL}
AUDIT[sec]: Setting callback for command ″listfs″ => {ACL}
AUDIT[cmd]: Creating command ″statfs″, callBack = {ACL}
AUDIT[sec]: Setting callback for command ″statfs″ => {ACL}
AUDIT[cmd]: Creating command ″whoami″, callBack = {NONE}
AUDIT[sec]: Setting callback for command ″whoami″ => {NONE}
AUDIT[cmd]: Creating command ″svcversion″, callBack = {NONE}
AUDIT[sec]: Setting callback for command ″svcversion″ => {NONE}
AUDIT[cmd]: Creating command ″svcrestart″, callBack = {ACL}
AUDIT[sec]: Setting callback for command ″svcrestart″ => {ACL}
AUDIT[cmd]: Creating command ″svclogevent″, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command ″svclogevent″ => {SYSTEM}
AUDIT[cmd]: Creating command ″svcconnect″, callBack = {AUTH}
AUDIT[sec]: Setting callback for command ″svcconnect″ => {AUTH}
AUDIT[cmd]: Creating command ″svcpid″, callBack = {ACL}
AUDIT[sec]: Setting callback for command ″svcpid″ => {ACL}
AUDIT[cmd]: Creating command ″svcredirect″, callBack = {ACL}
AUDIT[sec]: Setting callback for command ″svcredirect″ => {ACL}
AUDIT[cmd]: Creating command ″svcdetach″, callBack = {ACL}
AUDIT[sec]: Setting callback for command ″svcdetach″ => {ACL}
AUDIT[cmd]: Creating command ″svclog″, callBack = {ACL}
AUDIT[sec]: Setting callback for command ″svclog″ => {ACL}
AUDIT[cmd]: Creating command ″acladd″, callBack = {ACL}
AUDIT[sec]: Setting callback for command ″acladd″ => {ACL}
AUDIT[cmd]: Creating command ″acldelete″, callBack = {ACL}
AUDIT[sec]: Setting callback for command ″acldelete″ => {ACL}
AUDIT[cmd]: Creating command ″aclcheck″, callBack = {ACL}
AUDIT[sec]: Setting callback for command ″aclcheck″ => {ACL}
AUDIT[cmd]: Creating command ″acllist″, callBack = {ACL}
AUDIT[sec]: Setting callback for command ″acllist″ => {ACL}
AUDIT[cmd]: Creating command ″aclcreate″, callBack = {ACL}
AUDIT[sec]: Setting callback for command ″aclcreate″ => {ACL}
AUDIT[cmd]: Creating command ″aclrecreate″, callBack = {ACL}
```

```
AUDIT[sec]: Setting callback for command "aclrecreate" => {ACL}
AUDIT[cmd]: Creating command "acldestroy", callBack = {ACL}
AUDIT[sec]: Setting callback for command "acldestroy" => {ACL}
AUDIT[cmd]: Creating command "checkauth", callBack = {ACL}
AUDIT[sec]: Setting callback for command "checkauth" => {ACL}
AUDIT[cmd]: Creating command "setauth", callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "setauth" => {SYSTEM}
AUDIT[cmd]: Creating command "getauth", callBack = {ACL}
AUDIT[sec]: Setting callback for command "getauth" => {ACL}
AUDIT[cmd]: Creating command "create", callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "create" => {SYSTEM}
AUDIT[cmd]: Creating command "include", callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "include" => {SYSTEM}
AUDIT[cmd]: Creating command "confadd", callBack = {ACL}
AUDIT[sec]: Setting callback for command "confadd" => {ACL}
AUDIT[cmd]: Creating command "confdelete", callBack = {ACL}
AUDIT[sec]: Setting callback for command "confdelete" => {ACL}
AUDIT[cmd]: Creating command "load", callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "load" => {SYSTEM}
AUDIT[cmd]: Creating command "unload", callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "unload" => {SYSTEM}
AUDIT[cmd]: Replacing command "rename", callBack = {SYSTEM}
AUDIT[sec]: Replacing callback for command "rename" => {SYSTEM}
AUDIT[cmd]: Replacing command "proc", callBack = {SYSTEM}
AUDIT[sec]: Replacing callback for command "proc" => {SYSTEM}
AUDIT[cmd]: Creating command "NONE", callBack = {AUTH}
AUDIT[sec]: Setting callback for command "NONE" => {AUTH}
AUDIT[cmd]: Creating command "AUTH", callBack = {AUTH}
AUDIT[sec]: Setting callback for command "AUTH" => {AUTH}
AUDIT[cmd]: Creating command "ACL", callBack = {AUTH}
AUDIT[sec]: Setting callback for command "ACL" => {AUTH}
AUDIT[cmd]: Creating command "SYSTEM", callBack = {AUTH}
AUDIT[sec]: Setting callback for command "SYSTEM" => {AUTH}
AUDIT[var]: Creating variable buildTop = "/usr/lpp/ssp", callBack = {AUTH}
AUDIT[sec]: Setting callback for variable "buildTop" => {AUTH}
DEBUG: Reading class file: class = help, file = /usr/lpp/ssp/help/help.cmds,
       callback = {SYSTEM}
AUDIT[sec]: Setting callback for class "help" => {SYSTEM}
AUDIT[var]: Creating variable help:helpPath = "/usr/lpp/ssp/lib/sysctl.cmds/help",
            callBack = {SYSTEM}
AUDIT[sec]: Setting callback for variable "help:helpPath" => {SYSTEM}
AUDIT[cmd]: Creating procedure help:findDirs, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "help:findDirs" => {SYSTEM}
AUDIT[cmd]: Creating procedure help:setPath, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "help:setPath" => {SYSTEM}
AUDIT[cmd]: Creating procedure help:addPath, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "help:addPath" => {SYSTEM}
AUDIT[cmd]: Creating procedure help:getPath, callBack = {ACL}
AUDIT[sec]: Setting callback for command "help:getPath" => {ACL}
AUDIT[cmd]: Creating procedure help:delPath, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "help:delPath" => {SYSTEM}
AUDIT[cmd]: Creating procedure help:checkCommands, callBack = {AUTH}
AUDIT[sec]: Setting callback for command "help:checkCommands" => {AUTH}
AUDIT[cmd]: Creating procedure help:help, callBack = {AUTH}
AUDIT[sec]: Setting callback for command "help:help" => {AUTH}
AUDIT[cmd]: Renamed command "help:help" => "help"
DEBUG: Reading class file: class = sys, file = /usr/lpp/ssp/samples/sysctl/sys.cmds,
       callback = {SYSTEM}
AUDIT[sec]: Setting callback for class "sys" => {SYSTEM}
```

```
AUDIT[sec]: Setting callback for variable "sys:sysinfo" => {SYSTEM}
AUDIT[cmd]: Creating procedure sys:info, callBack = {AUTH}
AUDIT[sec]: Setting callback for command "sys:info" => {AUTH}
AUDIT[cmd]: Creating procedure sys:host_name, callBack = {AUTH}
AUDIT[sec]: Setting callback for command "sys:host_name" => {AUTH}
AUDIT[cmd]: Creating procedure sys:os_version, callBack = {AUTH}
AUDIT[sec]: Setting callback for command "sys:os_version" => {AUTH}
AUDIT[cmd]: Creating procedure sys:os_type, callBack = {AUTH}
AUDIT[sec]: Setting callback for command "sys:os_type" => {AUTH}
AUDIT[var]: Creating variable procStat = "", callBack = {SYSTEM}
AUDIT[sec]: Setting callback for variable "procStat" => {SYSTEM}
AUDIT[cmd]: Creating procedure procStat, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "procStat" => {SYSTEM}
AUDIT[cmd]: Creating procedure pstat, callBack = {AUTH}
AUDIT[sec]: Setting callback for command "pstat" => {AUTH}
AUDIT[cmd]: Creating procedure pkill, callBack = {ACL}
AUDIT[sec]: Setting callback for command "pkill" => {ACL}
AUDIT[cmd]: Creating procedure df, callBack = {AUTH}
AUDIT[sec]: Setting callback for command "df" => {AUTH}
AUDIT[cmd]: Creating procedure fscheck, callBack = {AUTH}
AUDIT[sec]: Setting callback for command "fscheck" => {AUTH}
AUDIT[cmd]: Creating procedure pdf, callBack = {AUTH}
AUDIT[sec]: Setting callback for command "pdf" => {AUTH}
AUDIT[cmd]: Creating procedure pfck, callBack = {AUTH}
AUDIT[sec]: Setting callback for command "pfck" => {AUTH}
AUDIT[cmd]: Creating procedure pfps_owner, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "pfps_owner" => {SYSTEM}
AUDIT[cmd]: Creating procedure pfps_name, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "pfps_name" => {SYSTEM}
AUDIT[cmd]: Creating procedure pfps_tname, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "pfps_tname" => {SYSTEM}
AUDIT[cmd]: Creating procedure pfps_pty, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "pfps_pty" => {SYSTEM}
AUDIT[cmd]: Creating procedure pfps_leap, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "pfps_leap" => {SYSTEM}
AUDIT[cmd]: Creating procedure pfps_julian_day, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "pfps_julian_day" => {SYSTEM}
AUDIT[cmd]: Creating procedure pfps_chk_month, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "pfps_chk_month" => {SYSTEM}
AUDIT[cmd]: Creating procedure pfps_rtime, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "pfps_rtime" => {SYSTEM}
AUDIT[cmd]: Creating procedure pfps_stime, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "pfps_stime" => {SYSTEM}
AUDIT[cmd]: Creating procedure pfps_state, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "pfps_state" => {SYSTEM}
AUDIT[cmd]: Creating procedure pfps_cpu, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "pfps_cpu" => {SYSTEM}
AUDIT[cmd]: Creating procedure pfps_mem, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "pfps_mem" => {SYSTEM}
AUDIT[cmd]: Creating procedure pfps_kill, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "pfps_kill" => {SYSTEM}
AUDIT[cmd]: Creating procedure pfps_nice, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "pfps_nice" => {SYSTEM}
AUDIT[cmd]: Creating procedure pfps, callBack = {AUTH}
AUDIT[sec]: Setting callback for command "pfps" => {AUTH}
AUDIT[var]: Creating variable STARTTIME = "Tue Apr 25 14:09:24 EDT 1995",
            callBack = {NONE}
AUDIT[sec]: Setting callback for variable "STARTTIME" => {NONE}
DEBUG: Reading class file: class = rcmds, file = /etc/remcmds.cmds, callback = {SYSTEM}
```

```
AUDIT[sec]: Setting callback for class "rcmds" => {SYSTEM}
AUDIT[cmd]: Creating procedure rcmds:rcmd_auth, callBack = {SYSTEM}
AUDIT[sec]: Setting callback for command "rcmds:rcmd_auth" => {SYSTEM}
AUDIT[cmd]: Creating procedure rcmds:rexec, callBack = {rcmds:rcmd_auth rexec}
AUDIT[sec]: Setting callback for command "rcmds:rexec" => {rcmds:rcmd_auth rexec}
AUDIT[cmd]: Renamed command "rcmds:rexec" => "rexec"
AUDIT[cmd]: Creating procedure rcmds:rlog, callBack = {rcmds:rcmd_auth rlog}
AUDIT[sec]: Setting callback for command "rcmds:rlog" => {rcmds:rcmd_auth rlog}
AUDIT[cmd]: Renamed command "rcmds:rlog" => "rlog"
DEBUG: Server interpreter contains 204 commands
DEBUG: Using port 6680
Server starting (pid=15423)
```

At this moment, the sysctld daemon is started. The following section shows the AUDIT output of a user connecting to the sysctl server. The first part is the authorization of svcconnect.

```
<2f9d3ad5> AUDIT[var]: Creating variable SCPRINCIPAL = "root.admin@ITSC.POK.IBM.COM",
                       callBack = {NONE}
<2f9d3ad5> AUDIT[sec]: Setting callback for variable "SCPRINCIPAL" => {NONE}
<2f9d3ad5> AUDIT[var]: Creating variable SCUSER = "root", callBack = {NONE}
<2f9d3ad5> AUDIT[sec]: Setting callback for variable "SCUSER" => {NONE}
<2f9d3ad5> AUDIT[var]: Creating variable SCINSTANCE = "admin", callBack = {NONE}
<2f9d3ad5> AUDIT[sec]: Setting callback for variable "SCINSTANCE" => {NONE}
<2f9d3ad5> AUDIT[var]: Creating variable SCREALM = "ITSC.POK.IBM.COM", callBack = {NONE}
<2f9d3ad5> AUDIT[sec]: Setting callback for variable "SCREALM" => {NONE}
<2f9d3ad5> AUDIT[var]: Creating variable SCHOST = "sp21cw0.itsc.pok.ibm.com",
                       callBack = {NONE}
<2f9d3ad5> AUDIT[sec]: Setting callback for variable "SCHOST" => {NONE}
<2f9d3ad5> AUDIT[var]: Creating variable SCLHOST = "sp21cw0.itsc.pok.ibm.com",
                       callBack = {NONE}
<2f9d3ad5> AUDIT[sec]: Setting callback for variable "SCLHOST" => {NONE}
<2f9d3ad5> AUDIT[var]: Creating variable SCLPRINCIPAL = "rcmd.sp21cw0@ITSC.POK.IBM.COM",
                       callBack = {NONE}
<2f9d3ad5> AUDIT[sec]: Setting callback for variable "SCLPRINCIPAL" => {NONE}
<2f9d3ad5> AUDIT[var]: Creating variable SCLREALM = "ITSC.POK.IBM.COM", callBack = {NONE}
<2f9d3ad5> AUDIT[sec]: Setting callback for variable "SCLREALM" => {NONE}
<2f9d3ad5> AUDIT[var]: Creating variable SCSCRIPT = " ", callBack = {NONE}
<2f9d3ad5> AUDIT[sec]: Setting callback for variable "SCSCRIPT" => {NONE}
<2f9d3ad5> AUDIT[var]: Creating variable SCPRIVATE = "0", callBack = {NONE}
<2f9d3ad5> AUDIT[sec]: Setting callback for variable "SCPRIVATE" => {NONE}
<2f9d3ad5> AUDIT[var]: Creating variable SCMODE = "SOCKET", callBack = {NONE}
<2f9d3ad5> AUDIT[sec]: Setting callback for variable "SCMODE" => {NONE}
<2f9d3ad5> AUDIT[sec]: Setting callback for variable "env" => {SYSTEM}
<2f9d3ad5> AUDIT[log]: Executing svclogevent callback ...
<2f9d3ad5> root.admin@ITSC.POK.IBM.COM on sp21cw0.itsc.pok.ibm.com
<2f9d3ad5> AUDIT[sec]: Executing svcconnect callback ...
<2f9d3ad5> AUDIT[sec]: Checking object authorization callback for
                       command "svcconnect": AUTH
<2f9d3ad5> AUDIT[sec]: interp->result from object callback {AUTH} =
                       "Authorization OK.", retCode = 0
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "svcconnect" via object callback
<2f9d3ad5> DEBUG: Read 51 bytes from client
```

When the authorization of the svcconnect procedure is successful, the user will get the sysctl prompt back, returning the version.

```
<2f9d3ad5> ---------------------------------------------------------------
<2f9d3ad5> AUDIT[tcl]: Evaluating:
           return ″Sysctl (Version [svcversion]) on $SCLHOST″
<2f9d3ad5> AUDIT[sec]: Checking object authorization callback for
                        command ″svcversion″: NONE
<2f9d3ad5> AUDIT[sec]: interp->result from object callback {NONE} =
                        ″Authorization OK.″, retCode = 0
<2f9d3ad5> AUDIT[sec]: Authorization granted for command ″svcversion″ via object
                        callback
<2f9d3ad5> AUDIT[sec]: Checking object authorization callback for variable
                        ″SCLHOST″: NONE
<2f9d3ad5> AUDIT[sec]: interp->result from object callback {NONE} =
                        ″Authorization OK.″, retCode = 0
<2f9d3ad5> AUDIT[sec]: Authorization granted for variable ″SCLHOST″ via
                        object callback
<2f9d3ad5> AUDIT[sec]: Checking object authorization callback for
                        command ″return″: NONE
<2f9d3ad5> AUDIT[sec]: interp->result from object callback {NONE} =
                        ″Authorization OK.″, retCode = 0
<2f9d3ad5> AUDIT[sec]: Authorization granted for command ″return″ via object callback
<2f9d3ad5> DEBUG: Received MSGEOC control sequence
<2f9d3ad5> DEBUG: Read 49 bytes of result
<2f9d3ad5> DEBUG: Read 7 bytes from client
```

In the next sections, the commands entered by the user are logged, evaluated and, on completion of the authorization, executed. The commands that were used in this AUDIT output, were:

- whoami
- info commands
- rlog, the procedure we created
- df
- exit

```
<2f9d3ad5> ---------------------------------------------------------------
<2f9d3ad5> AUDIT[tcl]: Evaluating: whoami
<2f9d3ad5> AUDIT[sec]: Checking object authorization callback for
                        command ″whoami″: NONE
<2f9d3ad5> AUDIT[sec]: interp->result from object callback {NONE} =
                        ″Authorization OK.″, retCode = 0
<2f9d3ad5> AUDIT[sec]: Authorization granted for command ″whoami″ via
                        object callback
<2f9d3ad5> DEBUG: Received MSGEOC control sequence
<2f9d3ad5> DEBUG: Read 28 bytes of result
<2f9d3ad5> DEBUG: Read 14 bytes from client
<2f9d3ad5> ---------------------------------------------------------------
<2f9d3ad5> AUDIT[tcl]: Evaluating: info commands
<2f9d3ad5> AUDIT[sec]: Checking object authorization callback for
                        command ″info″: ACL
<2f9d3ad5> AUDIT[sec]: interp->result from object callback {ACL} =
                        ″Authorization OK.″, retCode = 0
<2f9d3ad5> AUDIT[sec]: Authorization granted for command ″info″ via
                        object callback
<2f9d3ad5> DEBUG: Received MSGEOC control sequence
<2f9d3ad5> DEBUG: Read 1543 bytes of result
<2f9d3ad5> DEBUG: Read 5 bytes from client
<2f9d3ad5> ---------------------------------------------------------------
```

```
<2f9d3ad5> AUDIT[tcl]: Evaluating: rlog
<2f9d3ad5> AUDIT[sec]: Checking object authorization callback for
                       command "rlog": rcmds:rcmd_auth rlog
<2f9d3ad5> Authorizing "rlog" access for user root.admin@ITSC.POK.IBM.COM
<2f9d3ad5> AUDIT[sec]: interp->result from object
            callback {rcmds:rcmd_auth rlog} = "Authorization ok", retCode = 0
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "rlog" via object callback
<2f9d3ad5> AUDIT[sec]: Bypassing authorization checks while executing proc "rlog"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "global" via proc "rlog"
<2f9d3ad5> AUDIT[sec]: Authorization granted for variable "SCUSER" via proc "rlog"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "svclog" via proc "rlog"
<2f9d3ad5> rlog: user=root, command="ksh -i"
<2f9d3ad5> AUDIT[sec]: Authorization granted for variable "SCUSER" via proc "rlog"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "id" via proc "rlog"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "system" via proc "rlog"
<2f9d3ad5> DEBUG: Read 2 bytes of child stderr
<2f9d3ad5> DEBUG: Read 5 bytes from client
<2f9d3ad5> AUDIT[sec]: Restoring authorization checks at completion of proc "rlog"
<2f9d3ad5> DEBUG: Received MSGEOC control sequence
<2f9d3ad5> DEBUG: Read 2 bytes of result
<2f9d3ad5> DEBUG: Read 3 bytes from client
<2f9d3ad5> ------------------------------------------------------------
<2f9d3ad5> AUDIT[tcl]: Evaluating: df
<2f9d3ad5> AUDIT[sec]: Checking object authorization
                       callback for command "df": AUTH
<2f9d3ad5> AUDIT[sec]: interp->result from object
                       callback {AUTH} = "Authorization OK.", retCode = 0
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "df" via object callback
<2f9d3ad5> AUDIT[sec]: Bypassing authorization checks while executing proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "global" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "lempty" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "if" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "listfs" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "set" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for variable "SCLHOST" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "puts" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "foreach" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "statfs" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "format" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "puts" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "statfs" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "format" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "puts" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "statfs" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "format" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "puts" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "statfs" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "format" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "puts" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "statfs" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "format" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "puts" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "statfs" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "format" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "puts" via proc "df"
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "return" via proc "df"
<2f9d3ad5> AUDIT[sec]: Restoring authorization checks at completion of proc "df"
<2f9d3ad5> DEBUG: Read 419 bytes of child stdout
<2f9d3ad5> DEBUG: Received MSGEOC control sequence
```

```
<2f9d3ad5> DEBUG: Read 0 bytes of result
<2f9d3ad5> DEBUG: Read 5 bytes from client
<2f9d3ad5> ------------------------------------------------------------
<2f9d3ad5> AUDIT[tcl]: Evaluating: exit
<2f9d3ad5> AUDIT[sec]: Checking object authorization callback for
                       command "exit": AUTH
<2f9d3ad5> AUDIT[sec]: interp->result from object
                       callback {AUTH} = "Authorization OK.", retCod = 0
<2f9d3ad5> AUDIT[sec]: Authorization granted for command "exit" via object callback
<2f9d3ad5> DEBUG: Reaped child 35729, status=0
<2f9d3ad5> DEBUG: Sending EOF message to client (0)
```

Appendix F. Sysctl and TclX Command Reference

Below is a compilation of all commands currently available under sysctl. Some commands that have a description come from the sysctl manual pages, which can be found by entering **help** `command` in a sysctl interactive session. The format is as follows:

- Command
- Default authorization callback
- Synopsis, that is, the command syntax
- Command description, in some cases with examples
- Related information, related keywords

Tcl Summary of Tcl language syntax.

DESCRIPTION

The following rules define the syntax and semantics of the Tcl language:

1. A Tcl script is a string containing one or more commands. Semi-colons and newlines are command separators unless quoted as described below. Close brackets are command terminators during command substitution (see below) unless quoted.
2. A command is evaluated in two steps. First, the Tcl interpreter breaks the command into words and performs substitutions as described below. These substitutions are performed in the same way for all commands. The first word is used to locate a command procedure to carry out the command, then all of the words of the command are passed to the command procedure. The command procedure is free to interpret each of its words in any way it likes, such as an integer, variable name, list, or Tcl script. Different commands interpret their words differently.
3. Words of a command are separated by white space (except for newlines, which are command separators).
4. If the first character of a word is double-quote (″) then the word is terminated by the next double-quote character. If semi-colons, close brackets, or white space characters (including newlines) appear between the quotes then they are treated as ordinary characters and included in the word. Command substitution, variable substitution, and backslash substitution are performed on the characters between the quotes as described below. The double-quotes are not retained as part of the word.
5. If the first character of a word is an open brace ({) then the word is terminated by the matching close brace (}). Braces nest within the word: for each additional open brace there must be an additional close brace (however, if an open brace or close brace within the word is quoted with a backslash then it is not counted in locating the matching close brace). No substitutions are performed on the characters between the braces except for backslash-newline substitutions described below, nor do semi-colons, newlines, close brackets, or white space receive any

special interpretation. The word will consist of exactly the characters between the outer braces, not including the braces themselves.

6. If a word contains an open bracket ([) then Tcl performs command substitution. To do this it invokes the Tcl interpreter recursively to process the characters following the open bracket as a Tcl script. The script may contain any number of commands and must be terminated by a close bracket (]). The result of the script (that means the result of its last command) is substituted into the word in place of the brackets and all of the characters between them. There may be any number of command substitutions in a single word. Command substitution is not performed on words enclosed in braces.

7. If a word contains a dollar-sign ($) then Tcl performs variable substitution: the dollar-sign and the following characters are replaced in the word by the value of a variable. Variable substitution may take any of the following forms:

   - **$name**: Name is the name of a scalar variable; the name is terminated by any character that is not a letter, digit, or underscore.
   - **$name**(**index**): Name gives the name of an array variable and index gives the name of an element within that array. Name must contain only letters, digits, and underscores. Command substitutions, variable substitutions, and backslash substitutions are performed on the characters of index.
   - **$**{**name**}: Name is the name of a scalar variable. It may contain any characters whatsoever except for close braces.

   There may be any number of variable substitutions in a single word. Variable substitution is not performed on words enclosed in braces.

8. If a backslash () appears within a word then backslash substitution occurs. In all cases but those described below the backslash is dropped and the following character is treated as an ordinary character and included in the word. This allows characters such as double quotes, close brackets, and dollar signs to be included in words without triggering special processing. The following table lists the backslash sequences that are handled specially, along with the value that replaces each sequence.

   - **a**: Audible alert (bell) (0x7).
   - **b**: Backspace (0x8).
   - **f**: Form feed (0xc).
   - **n**: Newline (0xa).
   - **r**: Carriage-return (0xd).
   - **t**: Tab (0x9).
   - **v**: Vertical tab (0xb).
   - <**newline**>**whiteSpace**: A single space character replaces the backslash, newline, and all white space after the newline.

This backslash sequence is unique in that it is replaced in a separate pre-pass before the command is actually parsed. This means that it will be replaced even when it occurs between braces, and the resulting space will be treated as a word separator if it is not in braces or quotes.

- : Backslash ().
- **000**: The digits 000 (one, two, or three of them) give the octal value of the character.
- **xhh**: The hexadecimal digits hh give the hexadecimal value of the character. Any number of digits may be present.

Backslash substitution is not performed on words enclosed in braces, except for backslash-newline as described above.

9. If a hash character (#) appears at a point where Tcl is expecting the first character of the first word of a command, then the hash character and the characters that follow it, up through the next newline, are treated as a comment and ignored. The comment character only has significance when it appears at the beginning of a command.
10. Each character is processed exactly once by the Tcl interpreter as part of creating the words of a command. For example, if variable substitution occurs then no further substitutions are performed on the value of the variable; the value is inserted into the word verbatim. If command substitution occurs then the nested command is processed entirely by the recursive call to the Tcl interpreter; no substitutions are performed before making the recursive call and no additional substitutions are performed on the result of the nested script.
11. Substitutions do not affect the word boundaries of a command. For example, during variable substitution the entire value of the variable becomes part of a single word, even if the variable′s value contains spaces.

INSTALL Tcl Installation Description of Tcl

DESCRIPTION

When Extended Tcl is installed, the standard runtime files are places in the Tcl master directory, which is configured when Tcl is built. This master directory normally contains the Tcl initialization file (on the SP: /usr/lpp/ssp/lib/tcl/init.tcl), the standard Tcl library file (tcl.tlib) and the help files. On the SP, the default Tcl installation does not contain a Tcl library file. However, if you want to install the Tcl library file, or any other part of Tcl, you can extract it from the tclX7.3a-p2 package in the `/usr/lpp/ssp/public` directory. This package (tclX7.3a-p2.tar.Z) is a compressed tar file.

The Tcl master directory is named after the version of Tcl it is associated with, for example /usr/local/tclX/7.4a. The path to the Tcl master directory is available from the info library command. The location of the Tcl master directory can be overridden with the TCL_LIBRARY environment variable.

The first step in initializing the Tcl shell is to locate the Tcl initialization file, init.tcl. If an environment variable TCLINIT exists, it contains the path to the Tcl initialization file. If the TCLINIT environment variable is not set, the file TclInit.tcl is used from the default Tcl master directory.

Tcl then evaluates the Tcl initialization file. The auto_path variable is initialized to the Tcl master directory and may be augmented by the initialization file or the application. Other procedures and variables used by the Extended Tcl shell are also defined by this file.

If the Tcl is invoked interactively, it will source a file named **.tclrc** in the user′s home directory, if it exists. Tcl is viewed primarily as a programming language, not an interactive shell, so the .tclrc is intended for use for loading development utilities, not to support applications, which should not have to rely on the user′s environment in such a manner.

The Extended Tcl Tk shell, **/usr/lpp/ssp/bin/wish**, has an additional master directory and initialization file. It use the environment variable TK_LIBRARY to override the default location of the Tk master directory. The default location of the Tk master directory on the SP is /usr/lpp/ssp/lib/tk.

This functionality is provided by Extended Tcl.

/usr/lpp/ssp/bin/tclsh Tclsh starts the interactive Tcl command interpreter.

AIX command

SYNOPSIS

tclsh [-qn] [-f] script [| [-c command] [args]

DESCRIPTION

Tclsh starts the interactive Tcl command interpreter. The Tcl shell provides an environment for writing, debugging and executing Tcl scripts. The functionality of the Tcl shell can be easily obtained by any application that includes Tcl.

The tclsh command, issued without any arguments, invokes an interactive Tcl shell, allowing the user to interact directly with Tcl, executing any Tcl commands at will and viewing their results.

If script is specified, then the script is executed noninteractively with any additional arguments, args, being supplied in the global Tcl variable ″argv″. If command is supplied, then this command (or semicolon-separated series of commands) is executed, with ″argv″ containing any args.

The Tcl-shell is intended as an environment for Tcl program development and execution. While it is not a fullfeatured interactive shell, it provides a comfortable environment for the interactive development of Tcl code. Note that the package library code described here overrides the unknown command provided as part of the standard Berkeley Tcl library facility, although Tcl source libraries coded to that standard can be loaded and used by Extended Tcl.

The following command line flags are recognized by the Tcl shell command line parser:

- **-q**: Quick initialization flag. The Tcl initialization file is not evaluated and the auto_path variable is not set. Tcl auto-load libraries will not be available.

- **-n**: No procedure call stack dump. The procedure call stack will not be displayed when an error occurs, only the error message. Useful in the #! line of already debugged scripts.
- **-f**: Takes the next argument as a script for Tcl to source, rather than entering interactive mode. The -f flag is optional. Normally the first argument that does not start with a ″-″ is taken as the script to execute unless the ″-c″ option is specified. Any following arguments are passed to the script via argv, thus any other Tcl shell commandline flags must precede this option.
- **-c**: Take the next argument as a Tcl command to execute. It may contain series of commands to execute, separated by ″;″. Any following arguments are passed in argv, thus, as with -f, any other Tcl shell flags must precede this option.
- **--**: Mark the end of the arguments to the Tcl shell. All arguments following this are passed in the Tcl variable argv. This is useful to pass arguments without attempting to execute a Tcl script.

The result string returned by a command executed from the Tcl-shell command line is normally echoed back to the user. If an error occurs, then the string result is displayed, along with the error message. The error message will be preceded by the string ″Error:″.

The set command is a special case. If the command is called to set a variable (that means with two arguments), then the result will not be echoed. If only one argument, the name of a variable, is supplied to set, then the result will be echoed.

If an unknown Tcl command is entered from the command line, then the Unix command path, specified in the environment variable PATH, will be searched for a command of the same name. If the command is found, it will be executed with any arguments remaining on the Tcl command line being passed as arguments to the command. This feature is provided to enhance the interactive environment for developing Tcl scripts.

Automatic execution of programs in this manner is only supported from the command line, not in script files or in procedures, to reduce confusion and mistakes while programming in Tcl. Scripts should use the Tcl exec or system commands to run Unix commands.

This functionality is provided by Extended Tcl.

The following variables are set and/or used by the Tcl shell.

- **argv0**: Contains the name of the Tcl program specified on the command line or the name that the Tcl shell was invoked under if no program was specified. argc Contains a count of the number of argv arguments (0 if none). argv A list containing the arguments passed in from the command line, excluding arguments used by the Tcl shell. The first element is the first passed argument, not the program name.
- **tcl_interactive**: Set to 1 if Tcl shell is invoked interactively, or 0 if the Tcl shell is directly executing a script. Normally checked by scripts so that they can function as a standalone application if specified on the command line, but merely load in and not execute if loaded during an interactive invocation of Tcl.

- **auto_path**: Path to search to locate Tcl scripts. Used by the auto_load command and the TclX unknown command handler. The path is a Tcl list of directory names.
- **tclx_library**: Path to the TclX runtime library. If your running the TclX shell or an application based on it (like /usr/lpp/ssp/bin/wish), this is the same value returned by ″info library″.
- **tcl_prompt1**: Contains code to run to output the prompt used when interactively prompting for commands.
- **tcl_prompt2**: Contains code to run to output the prompt used when interactively prompting for continuation of an incomplete command.
- **TCLXENV**: Array that contains information used internally by various Tcl procedures that are part of the TclX shell. Don′t change this array unless you know what your doing.

library standard library of Tcl procedures

SYNOPSIS

- auto_execok cmd
- auto_load cmd
- auto_mkindex dir pattern pattern ...
- auto_reset
- parray arrayName
- unknown cmd [arg arg ...]

DESCRIPTION

Tcl includes a library of Tcl procedures for commonly needed functions. The procedures defined in the Tcl library are generic ones suitable for use by many different applications. The location of the Tcl library is returned by the info library command. In addition to the Tcl library, each application will normally have its own library of support procedures as well; the location of this library is normally given by the value of the $app_library global variable, where app is the name of the application. For example, the location of the Tk library is kept in the variable $tk_library.

To access the procedures in the Tcl library, an application should source the file init.tcl in the library, for example with the Tcl command:

```
source [info library]/init.tcl
```

This will define the unknown procedure and arrange for the other procedures to be loaded on-demand using the autoload mechanism defined below.

COMMAND PROCEDURES

The following procedures are provided in the Tcl library:

- **auto_execok cmd**: Determines whether there is an executable file by the name cmd. This command examines the directories in the current search path (given by the PATH environment variable) to see if there is an executable file named cmd in any of those directories. If so, it returns 1; if not it returns 0. Auto_exec remembers information about previous searches in an array named auto_execs; this avoids the path search in future calls for the same cmd. The command auto_reset may be used to force auto_execok to forget its cached information.

- **auto_load cmd**: This command attempts to load the definition for a Tcl command named cmd. To do this, it searches an auto-load path, which is a list of one or more directories. The auto-load path is given by the global variable $auto_path if it exists. If there is no $auto_path variable, then the TCLLIBPATH environment variable is used, if it exists. Otherwise the auto-load path consists of just the Tcl library directory. Within each directory in the auto-load path there must be a file tclIndex that describes one or more commands defined in that directory and a script to evaluate to load each of the commands. The tclIndex file should be generated with the auto_mkindex command. If cmd is found in an index file, then the appropriate script is evaluated to create the command. The auto_load command returns 1 if cmd was successfully created. The command returns 0 if there was no index entry for cmd or if the script did not actually define cmd (for example because index information is out of date). If an error occurs while processing the script, then that error is returned. Auto_load only reads the index information once and saves it in the array auto_index; future calls to auto_load check for cmd in the array rather than re-reading the index files. The cached index information may be deleted with the command auto_reset. This will force the next auto_load command to reload the index database from disk.

- **auto_mkindex dir pattern pattern ...**: Generates an index suitable for use by auto_load. The command searches dir for all files whose names match any of the pattern arguments (matching is done with the glob command), generates an index of all the Tcl command procedures defined in all the matching files, and stores the index information in a file named tclIndex in dir. For example, the command:

  ```
  auto_mkindex foo *.tcl
  ```

  will read all the .tcl files in subdirectory foo and generate a new index file foo/tclIndex.

  Auto_mkindex parses the Tcl scripts in a relatively unsophisticated way: if any line contains the word proc as its first characters then it is assumed to be a procedure definition and the next word of the line is taken as the procedure′s name. Procedure definitions that do not appear in this way (for example they have spaces before the proc) will not be indexed.

- **auto_reset**: Destroys all the information cached by auto_execok and auto_load. This information will be re-read from disk the next time it is needed. Auto_reset also deletes any procedures listed in the auto-load index, so that fresh copies of them will be loaded the next time that they are used.

- **parray arrayName**: Prints on standard output the names and values of all the elements in the array arrayName. ArrayName must be an array accessible to the caller of parray. It may be either local or global.

- **unknown cmd** [**arg arg ...**]: This procedure is invoked automatically by the Tcl interpreter whenever the name of a command does not exist. The unknown procedure receives as its arguments the name and arguments of the missing command. Unknown first calls auto_load to load the command. If this succeeds, then it executes the original command with its original arguments. If the auto-load fails then unknown calls auto_execok to see if there is an executable file by the name cmd. If so, it invokes the Tcl exec command with cmd and all the args as arguments. If cmd cannot be auto-executed, unknown checks to see if the command was invoked at top-level and outside of any script. If so, then unknown takes takes two additional steps. First, it sees if cmd has one of the following three forms: !!, !event, or ¬old¬new[¬]. If so, then unknown carries out history substitution in the same way that csh would for these constructs. Second, and last, unknown checks to see if cmd is a unique abbreviation for an existing Tcl command. If so, it expands the command name and executes the command with the original arguments. If none of the above efforts has been able to execute the command, unknown generates an error return. If the global variable auto_noload is defined, then the auto-load step is skipped. If the global variable auto_noexec is defined then the auto-exec step is skipped. Under normal circumstances the return value from unknown is the return value from the command that was eventually executed.

VARIABLES

The following global variables are defined or used by the procedures in the Tcl library:

- **auto_execs**: Used by auto_execok to record information about whether particular commands exist as executable files.
- **auto_index**: Used by auto_load to save the index information read from disk.
- **auto_noexec**: If set to any value, then unknown will not attempt to auto-exec any commands.
- **auto_noload**: If set to any value, then unknown will not attempt to auto-load any commands.
- **auto_path**: If set, then it must contain a valid Tcl list giving directories to search during auto-load operations.
- **env**(**TCL_LIBRARY**): If set, then it specifies the location of the directory containing library scripts (the value of this variable will be returned by the command info library). If this variable is not set then a default value is used.
- **env**(**TCLLIBPATH**): If set, then it must contain a valid Tcl list giving directories to search during auto-load operations. This variable is only used if auto_path is not defined.
- **unknown_active**: This variable is set by unknown to indicate that it is active. It is used to detect errors where unknown recurses on itself infinitely. The variable is unset before unknown returns.

TCL LOADABLE LIBRARIES AND PACKAGES

Extended Tcl supports standard Tcl tclIndex libraries and package libraries. A package library file can contain multiple independent Tcl packages. A package is a named collection of related Tcl procedures and initialization code.

The package library file is just a regular Unix text file, editable with your favorite text editor, containing packages of Tcl source code. The package library file name must have the suffix .tlib. An index file with the suffix .tndx, corresponding to the package library. The .tndx will be automatically created by Tcl whenever it is out of date or missing (provided there is write access to the directory.

The variable auto_path contains a list of directories that are searched for libraries. The first time an unknown command trap is take, the indexes for the libraries are loaded into memory. If the auto_path variable is changed during execution of a program, it will be re-searched. Only the first package of a given name found during the execution of a program is loaded. This can be overridden with loadlibindex command.

The start of a package is delimited by:

```
#@package: package_name proc1 [..procN]
```

These lines must start in column one. Everything between the #@package: keyword and the next #@package: keyword or a #@packend keyword, or the end of the file, becomes part of the named package. The specified procedures, proc1..procN, are the entry points of the package. When a command named in a package specification is executed and detected as an unknown command, all code in the specified package will be sourced. This package should define all of the procedures named on the package line, define any support procedures required by the package and do any package-specific initialization. Packages declarations maybe continued on subsequent lines using standard Tcl backslash line continuations. The #@packend keyword is useful to make sure only the minimum required section of code is sourced. Thus for example a large comment block at the beginning of the next file will not be loaded.

Care should be taken in defining package_name, as the first package found in the path by with a given name is loaded. This can be useful in developing new version of packages installed on the system.

For example, in a package source file, the presence of the following line:

```
#@package: directory_stack pushd popd dirs
```

says that the text lines following that line in the package file up to the next package line or the end of the file is a package named directory_stack and that an attempt to execute either pushd, popd or dirs when the routine is not already defined will cause the directory_stack portion of the package file to be loaded.

This functionality is provided by Extended Tcl.

KEYWORDS

auto-exec, auto-load, library, unknown

KEYED LISTS Extended Tcl defines a special type of list referred to as keyed lists. These lists provided a structured data type built upon standard Tcl lists. This provides a functionality similar to structs in the C programming language.

A keyed list is a list in which each element contains a key and value pair. These element pairs are stored as lists themselves, where the key is the first element of the list, and the value is the second. The key-value pairs are refered to as fields. This is an example of a keyed list:

```
{{NAME {Frank Zappa}} {JOB {musician and composer}}}
```

If the variable person contained the above list, then keylget person NAME would return {Frank Zappa}. Executing the command:

```
keylset person ID 106
```

would make person contain

```
{{ID 106} {NAME {Frank Zappa}} {JOB {musician and composer}}
```

Fields may contain subfields; ″.″ is the separator character. Subfields are actually fields where the value is another keyed list. Thus the following list has the top level fields ID and NAME, and subfields NAME.FIRST and NAME.LAST:

```
{ID 106} {NAME {{FIRST Frank} {LAST Zappa}}}
```

There is no limit to the recursive depth of subfields, allowing one to build complex data structures.

Keyed lists are constructed and accessed via a number of commands. All keyed list management commands take the name of the variable containing the keyed list as an argument (that means passed by reference), rather than passing the list directly.

This functionality is provided by Extended Tcl.

ACL Returns ″Authorization OK″, if in ACL, otherwise

sysctl: 2501-122 ACL: Insufficient Authorization.

AUTH AUTH

SYNOPSIS

ACL [fileName]

DESCRIPTION

The ACL command is typically used in the authorization callback for a procedure or variable declaration. A successful (non-error) code is returned if one of the following is true:

- The remote user is authenticated as the local host principal.
- The remote user appears in the ACL file passed as an argument. If no ACL file is passed, the file defined by the $ACL variable is searched.

Examples:

```
create proc foo {} ACL {
        ....
}

create proc bar {} {ACL /tmp/bar.acl} {
        ....
}
```

AUTH Returns ″Authorization OK″, if Kerberos ticket found, otherwise sysctl: 2501-122 AUTH: Insufficient Authorization.

AUTH AUTH

SYNOPSIS

DESCRIPTION

The AUTH command is typically used in the authorization callback for a procedure or variable declaration. A successful (non-error) code is returned if the identity of the remote user has been authenticated.

Examples:

```
create proc foo {} AUTH {
        ....
}
```

NONE Returns ″Authorization OK″

AUTH AUTH

SYNOPSIS

NONE

DESCRIPTION

The NONE command is typically used in the authorization callback for a procedure or variable declaration. A successful (non-error) code is always returned. Thus, any command with NONE as its authorization callback can be executed by any user.

Examples:

```
create proc foo {} NONE {
         ....
}
```

SYSTEM Returns the following in an interactive session:

```
sysctl:  2501-140 Authorization Denied.
```

AUTH AUTH

SYNOPSIS

SYSTEM

DESCRIPTION

The SYSTEM command is typically used in the authorization callback for a procedure or variable declaration. A successful (non-error) code is never returned. Thus, any command with SYSTEM as its authorization callback cannot be executed by any user directly. This is typically used to define commands that can only be executed within the context of a pre-defined procedure (authorization callbacks are bypassed while a procedure is executing).

acladd **AUTH** ACL

SYNOPSIS

acladd [-f aclFile] [-p principal ...] [-a aclFile ...]

DESCRIPTION

The acladd command inserts new entries into an existing ACL file. If the -f option is not specified, the new entries are added to the default system ACL file. The -p option is used to add one or more principal names to the ACL file. The principal names are canonicalized into the "user.instance@REALM" format before being added to the ACL. The -a option is used to add one or more ACL files to the ACL. All path names used in conjunction with the -a flag should be fully-qualified path names.

The acladd command returns the canonicalized entries that it adds to the ACL file.

Example:

```
sysctl> acladd -p vatore -a /acls/this.acl
_PRINCIPAL vatore.@WATSON.IBM.COM
_ACL_FILE /acls/this.acl
sysctl>
```

RELATED INFORMATION

aclcheck, aclcreate, acldelete, acldestroy, acllist, aclrecreate

aclcheck **AUTH** ACL

SYNOPSIS

aclcheck [-f aclFile] principal

DESCRIPTION

The aclcheck command searches an ACL file looking for the principal given on the command line. If an ACL file is not specified via the -f option, the default system ACL is checked. If the target ACL file includes additional ACL files, those files are also checked.

The aclcheck command returns 1 if the principal was found, 0 if not.

Example:

```
sysctl> acllist -f /acls/this.acl
_PRINCIPAL vatore.@WATSON.IBM.COM
sysctl> aclcheck -f /acls/this.acl vatore
1
sysctl> aclcheck -f /acls/this.acl foo
0
sysctl>
```

RELATED INFORMATION

acladd, aclcreate, acldelete, acldestroy, acllist, aclrecreate

aclcreate **AUTH** ACL

SYNOPSIS

aclcreate [-f aclFile] [-p principal ...] [-a aclFile ...]

DESCRIPTION

The aclcreate command creates a new ACL file and optionally adds principals and/or ACL files to it. If an ACL file is not specified using the -f option, the default system ACL file is used. Any principals or ACL files specified using the -a or -p flags are added to the ACl after it is created. All principal names are canonicalized into the ″user.instance@REALM″ format before being added.

The aclcreate command returns a list of the records added to the file. If the target ACL file already exists, an error is returned.

Example:

```
sysctl> aclcreate -f /acls/this.acl -a /acls/another.acl
_ACL_FILE /acls/another.acl
sysctl>
```

RELATED INFORMATION

acladd, aclcheck, acldelete, acldestroy, acllist, aclrecreate

acldestroy **AUTH** ACL

SYNOPSIS

acldestroy [-f aclFile]

DESCRIPTION

The acldestroy command erases an existing ACL file. If no ACl file is specified with the -f option, the default system ACL is destroyed.

Example:

```
sysctl> acllist -f /acls/this.acl
_PRINCIPAL vatore.@WATSON.IBM.COM
sysctl> acldestroy -f /acls/this.acl
sysctl> acllist -f /acls/this.acl
/acls/this.acl: No such file or directory
sysctl>
```

RELATED INFORMATION

acladd, aclcheck, aclcreate, acldelete, acllist, aclrecreate

acllist **AUTH** ACL

SYNOPSIS

acllist [-f aclFile]

DESCRIPTION

The acllist command returns the contents of an ACL file as a newline-separated string. If no ACL file is specified using the -f option, the default system ACL is used.

Example:

```
sysctl> acllist
_PRINCIPAL vatore.@WATSON.IBM.COM
_ACL_FILE /acls/this.acl
sysctl>
```

RELATED INFORMATION

acladd, aclcheck, aclcreate, acldelete, acldestroy, aclrecreate

aclrecreate AUTH ACL

SYNOPSIS

aclcreate [-f aclFile] [-p principal ...] [-a aclFile ...]

DESCRIPTION

The aclcreate command creates a new ACL file and optionally adds principals and/or ACL files to it. If an ACL file is not specified using the -f option, the default system ACL file is used. Any principals or ACL files specified using the -p or -a flags are added to the ACl after it is created. All principal names are canonicalized into the ″user.instance@REALM″ format before being added.

The aclcreate command returns a list of the records added to the file. If the target ACL file already exists, an error is returned.

Example:

```
sysctl> aclcreate -f /acls/this.acl -a /acls/another.acl
_ACL_FILE /acls/another.acl
sysctl>
```

RELATED INFORMATION

acladd, aclcheck, acldelete, acldestroy, acllist, aclrecreate

acldelete **AUTH** ACL

SYNOPSIS

acldelete [-f aclFile] [-p principal ...] [-a aclFile ...]

DESCRIPTION

The acldelete command deletes a principal name or ACL file entry from an existing ACL file. If an aclFile is not specified via the -f option, the default system ACL is used. Principal names specified with the -p flag are canonicalized into the ″user.instance@REALM″ format. If a matching principal or ACL file entry is found in the ACL, it is removed. The acldelete command does not recursively delete entries. That is, if the ACL file being modified includes additional ACL files, acldelete does not attempt to delete records from the included files.

The acldelete command returns all records that were deleted from the ACL file.

Example:

```
sysctl> acllist
_PRINCIPAL vatore.@WATSON.IBM.COM
_PRINCIPAL ctl.root@WATSON.IBM.COM
sysctl> acldelete -p vatore
_PRINCIPAL vatore.@WATSON.IBM.COM
sysctl> acllist
_PRINCIPAL ctl.root@WATSON.IBM.COM
sysctl>
```

RELATED INFORMATION

acladd, aclcheck, aclcreate, acldestroy, acllist, aclrecreate

alarm Sets the number of seconds to exit sysctl (forced)

AUTH ACL

SYNOPSIS

alarm number

DESCRIPTION

Instructs the system to send a SIGALRM signal in the specified number of seconds. This is a floating point number, so fractions of a section may be specified. If seconds is 0.0, any previous alarm request is canceled. Only one alarm at a time may be active; the command returns the number of seconds left in the previous alarm. On systems without the setitimer system call, seconds is rounded up to an even number of seconds.

This command is provided by Extended Tcl.

append Append to variable

AUTH AUTH

SYNOPSIS

append varName value [value ...]

DESCRIPTION

Append all of the value arguments to the current value of variable varName. If varName does not exist, it is given a value equal to the concatenation of all the value arguments. This command provides an efficient way to build up long variables incrementally. For example, ″append a $b″ is much more efficient than ″set a ab″ if $a is long.

RELATED INFORMATION

append, variable

array **AUTH** AUTH

SYNOPSIS

array option arrayName [arg ...]

DESCRIPTION

This command performs one of several operations on the variable given by arrayName. Unless otherwise specified for individual commands below, arrayName must be the name of an existing array variable. The option argument determines what action is carried out by the command. The legal options (which may be abbreviated) are:

- **array anymore arrayName searchId**: Returns 1 if there are any more elements left to be processed in an array search, 0 if all elements have already been returned. SearchId indicates which search on arrayName to check, and must have been the return value from a previous invocation of array startsearch. This option is particularly useful if an array has an element with an empty name, since the return value from array nextelement will not indicate whether the search has been completed.
- **array donesearch arrayName searchId**: This command terminates an array search and destroys all the state associated with that search. SearchId indicates which search on arrayName to

destroy, and must have been the return value from a previous invocation of array startsearch. Returns an empty string.

- **array exists arrayName**: Returns 1 if arrayName is an array variable, 0 if there is no variable by that name or if it is a scalar variable.
- **array get arrayName**: Returns a list containing pairs of elements. The first element in each pair is the name of an element in arrayName and the second element of each pair is the value of the array element. The order of the pairs is undefined. If arrayName is not the name of an array variable, or if the array contains no elements, then an empty list is returned.
- **array names arrayName** [**pattern**]: Returns a list containing the names of all of the elements in the array that match pattern (using the glob-style matching rules of string match). If pattern is omitted then the command returns all of the element names in the array. If there are no (matching) elements in the array, or if arrayName is not the name of an array variable, then an empty string is returned.
- **array nextelement arrayName searchId**: Returns the name of the next element in arrayName, or an empty string if all elements of arrayName have already been returned in this search. The searchId argument identifies the search, and must have been the return value of an array startsearch command.

  Note: If elements are added to or deleted from the array, then all searches are automatically terminated just as if array donesearch had been invoked; this will cause array nextelement operations to fail for those searches.
- **array set arrayName list**: Sets the values of one or more elements in arrayName. The list must have a form like that returned by array get, consisting of an even number of elements. Each odd-numbered element in list is treated as an element name within arrayName, and the following element in list is used as a new value for that array element.
- **array size arrayName**: Returns a decimal string giving the number of elements in the array. If arrayName is not the name of an array then 0 is returned.
- **array startsearch arrayName**: This command initializes an element-by-element search through the array given by arrayName, such that invocations of the array nextelement command will return the names of the individual elements in the array. When the search has been completed, the array donesearch command should be invoked. The return value is a search identifier that must be used in array nextelement and array donesearch commands; it allows multiple searches to be underway simultaneously for the same array.

RELATED INFORMATION

array, element names, search for_array_keys var array_name code

This procedure performs a foreach-style loop for each key in the named array. The break and con- tinue statements work as with foreach.

This procedure is provided by Extended Tcl.

break Breaks the continuation of a loop

AUTH AUTH

SYNOPSIS

break

DESCRIPTION

This command is typically invoked inside the body of a looping command such as for or foreach or while. It returns a TCL_BREAK code, which causes a break exception to occur. The exception causes the current script to be aborted out to the the innermost containing loop command, which then aborts its execution and returns normally. Break exceptions are also handled in a few other situations, such as the catch command, Tk event bindings, and the outermost scripts of procedure bodies.

RELATED INFORMATION

abort, break, loop

bsearch Search an opened file

AUTH ACL

SYNOPSIS

bsearch fileId key [retvar] [compare_proc]

DESCRIPTION

Search an opened file fileId containing lines of text sorted into ascending order for a match. Key contains the string to match. If retvar is specified, then the line from the file is returned in retvar, and the command returns 1 if key was found, and 0 if it was not. If retvar is not specified or is a null name, then the command returns the line that was found, or an empty string if key was not found.

By default, the key is matched against the first white-space separated field in each line. The field is treated as an ASCII string. If compare_proc is specified, then it defines the name of a Tcl procedure to evaluate against each line read from the sorted file during the execution of the bsearch command. Compare_proc takes two arguments, the key and a line extracted from the file. The compare routine should return a number less than zero if the key is less than the line, zero if the key matches the line, or greater than zero if the key is greater than the line. The file must be sorted in ascending order according to the same criteria compare_proc uses to compare the key with the line, or erroneous results will occur.

This command is provided by Extended Tcl.

case Evaluate one of several scripts, depending on a given value.

AUTH AUTH

SYNOPSIS

case string [in] patList body ... [default body]

DESCRIPTION

Note: The case command is obsolete and is supported only for backward compatibility. At some point in the future it may be removed entirely. You should use the switch command instead.

The case command matches string against each of the patList arguments in order. Each patList argument is a list of one or more patterns. If any of these patterns matches string then case evaluates the following body argument by passing it recursively to the Tcl interpreter and returns the result of that evaluation. Each patList argument consists of a single pattern or list of patterns. Each pattern may contain any of the wild-cards described under string match. If a patList argument is default, the corresponding body will be evaluated if no patList matches string. If no patList argument matches string and no default is given, then the case command returns an empty string.

Two syntaxes are provided for the patList and body arguments. The first uses a separate argument for each of the patterns and commands; this form is convenient if substitutions are desired on some of the patterns or commands. The second form places all of the patterns and commands together into a single argument; the argument must have proper list structure, with the elements of the list being the patterns and commands. The second form makes it easy to construct multi-line case commands, since the braces around the whole list make it unnecessary to include a backslash at the end of each line. Since the patList arguments are in braces in the second form, no command or variable substitutions are performed on them; this makes the behavior of the second form different than the first form in some cases.

RELATED INFORMATION

case, match, regular expression

catch Evaluate script and trap exceptional returns

AUTH AUTH

SYNOPSIS

catch command [varName]

DESCRIPTION

The catch command may be used to prevent errors from aborting command interpretation. Catch calls the Tcl interpreter recursively to execute script, and always returns a TCL_OK code, regardless of any errors that might occur while executing script. The return value from catch is a decimal string giving the code returned by the Tcl interpreter after executing script. This will be 0 (TCL_OK) if there were no errors in script; otherwise it will have a non-zero value corresponding to one of the exceptional return codes (see tcl.h for the definitions of code values). If the varName argument is given, then it gives the name of a variable; catch will set the variable to the string returned from script (either a result or an error message).

Note that catch catches all exceptions, including those generated by break and continue as well as errors.

RELATED INFORMATION

catch, error

catclose **AUTH** ACL

SYNOPSIS

catclose [-fail|-nofail] catHandle

DESCRIPTION ACL

Close the message catalog specified by cathandle. Normally, errors are ignored. If -fail is specified, any errors closing the message catalog file are returned. The option -nofail specifies the default behavior of not returning an error. The use of -fail only makes sense if it was also specified in the call to catopen.

This command is provided by Extended Tcl.

catgets **AUTH** ACL

SYNOPSIS

catgets catHandle setnum msgnum defaultstr

DESCRIPTION

Retrieve a message form a message catalog. CatHandle should be a Tcl message catalog handle that was returned by catopen. Setnum is the message set number, and msgnum is the message number. If the message catalog was not opened, or the message set or message number cannot be found, then the default string, defaultstr, is returned.

This command is provided by Extended Tcl.

catopen **AUTH** ACL

SYNOPSIS

catopen [-fail | -nofail] catname

DESCRIPTION

Open the message catalog catname. This may be a relative path name, in which case the NLSPATH environment variable is searched to find an absolute path to the message catalog. A handle in the form msgcatN is returned. Normally, errors are ignored, and in the case of a failed call to catopen, a handle is returned to an unopened message catalog. (This handle may still be passed to catgets and catclose, causing catgets to simply return the default string, as described above.) If the -fail option is specified, an error is returned if the open fails. The option -nofail specifies the default behavior of not returning an error when catopen fails to open a specified message catalog. If the handle from a failed catopen is passed to catgets, the default string is returned.

This command is provided by Extended Tcl.

cd Change working directory

AUTH ACL

SYNOPSIS

cd [dirName]

DESCRIPTION

Change the current working directory to dirName, or to the home directory (as specified in the HOME environment variable) if dirName is not given. If dirName starts with a tilde, then tilde-expansion is done as described for Tcl_TildeSubst. Returns an empty string.

RELATED INFORMATION

working directory

This procedure is provided by Extended Tcl.

cequal **AUTH** AUTH

SYNOPSIS

cequal string1 string2

DESCRIPTION

This command compares two strings for equality. It returns 1 if string1 and string2 are the identical and 0 if they are not. This command is a short-cut for string compare and avoids the problems with string expressions being treated unintentionally as numbers.

This command is provided by Extended Tcl.

cexpand **AUTH** AUTH

SYNOPSIS

cexpand string

DESCRIPTION

Expand backslash sequences in string to their actual characters. No other substitution takes place.

This command is provided by Extended Tcl.

checkauth Determines if you are authorized to access an object

AUTH ACL

SYNOPSIS

checkauth -cmd | -var | -class objName

DESCRIPTION

The checkauth command executes the authorization callbacks for the given object and determines whether or not access would be granted. It is typically used to debug new procedures, variables, or class files.

RELATED INFORMATION

setauth, getauth, create

chgrp **AUTH** ACL

SYNOPSIS

chgrp group filelist

DESCRIPTION

Set the group id of each file in the list filelist to group, which can be either a group name or a numeric group id.

This command is provided by Extended Tcl.

chmod **AUTH** ACL

SYNOPSIS

chmod mode filelist

DESCRIPTION

Set permissions of each of the files in the list filelist to mode, where mode is an absolute numeric mode or symbolic permissions as in the UNIX chmod(1) command. To specify a mode as octal, it should be prefixed with a ″0″ (that is 0622).

This command is provided by Extended Tcl.

chown **AUTH** ACL

SYNOPSIS

chown user|{user group} filelist

DESCRIPTION

Set owner of each file in the list filelist to owner, which can be a user name or numeric user id. If the first parameter is a list, then the owner is set to the first element of the list and the group is set to the second element. Group can be a group name or numeric group id. If group is {}, then the file group will be set to the login group of the specified user.

This command is provided by Extended Tcl.

chroot **AUTH** SYSTEM

SYNOPSIS

chroot dirname

DESCRIPTION

Change root directory to dirname, by invoking the POSIX chroot(2) system call. This command only succeeds if running as root.

This command is provided by Extended Tcl.

cindex **AUTH** AUTH

SYNOPSIS

cindex string indexExpr

DESCRIPTION

Returns the character indexed by the expression indexExpr (zero based) from string.

If the expression indexExpr starts with the string end, then end is replaced with the index of the last character in the string. If the expression starts with len, then len is replaced with the length of the string.

This command is provided by Extended Tcl.

clength **AUTH** AUTH

SYNOPSIS

clength string

DESCRIPTION

Returns the length of string in characters. This command is a shortcut for:

```
string length string
```

This command is provided by Extended Tcl.

close **AUTH** ACL

SYNOPSIS

close fileId

DESCRIPTION

Closes the file given by fileId. FileId must be the return value from a previous invocation of the open command; after this command, it should not be used anymore. If fileId refers to a command pipeline instead of a file, then close waits for the children to complete. The normal result of this command is an empty string, but errors are returned if there are problems in closing the file or waiting for children to complete.

RELATED INFORMATION

close, file

cmdtrace **AUTH** ACL

SYNOPSIS

cmdtrace level | on [noeval] [notruncate] [procs][fileid] | off | depth

DESCRIPTION

Print a trace statement for all commands executed at depth of level or below (1 is the top level). If on is specified, all commands at any level are traced. The following options are available:

- **noeval**: Causes arguments to be printed unevaluated. If noeval is specified, the arguments are printed before evaluation. Otherwise, they are printed afterwards.

  If the command line is longer than 60 characters, it is truncated to 60 and a ″...″ is postpended to indicate that there was more output than was displayed. If an evaluated argument contains a space, the entire argument will be enclosed inside of braces ({}′) to allow the reader to visually separate the arguments from each other.

- **notruncate**: Disables the truncation of commands and evaluated arguments.

- **procs**: Enables the tracing of procedure calls only. Commands that are not procedure calls (that means calls to commands that are written in C, C++ or some object-compatible language) are not traced if the procs option is specified. This option is particularly useful for greatly reducing the output of cmdtrace while debugging.

- **fileid**: This is a file id as returned by the open command. If specified, then the trace output will be written to the file rather than stdout. A stdio buffer flush is done after every line is written so that the trace may be monitored externally or provide useful information for debugging problems that cause core dumps.

The most common use of this command is to enable tracing to a file during the development. If a failure occurs, a trace is then available when needed. Command tracing will slow down the execution of code, so it should be removed when code is debugged. The following command will enable tracing to a file for the remainder of the program:

```
cmdtrace on [open cmd.log w]
```

- **cmdtrace off**: Turn off all tracing.
- **cmdtrace depth**: Returns the current maximum trace level, or zero if trace is disabled.

This command is provided by Extended Tcl.

commandloop AUTH ACL

SYNOPSIS

commandloop

DESCRIPTION

Create an interactive command loop for the current TCL interpreter. This command receives commands from stdin and executes them. It is useful TCL scripts that do not normally converse interactively with a user through a Tcl command interpreter, but which sometimes want to enter this mode.

Prompt1 is a Tcl command that is evaluated to output a prompt string. The old value of tcl_prompt1 is saved and it is set to this value for the duration of the command loop. Prompt2 is a command that is evaluated to output the ″downlevel prompt″, which is the prompt which is issued for continuation input. The old value of tcl_prompt2 is saved and it is set to this value for the duration of the command loop.

When the command terminates, the variables for the prompt hooks will be set to their old value. If these arguments are not specified, the prompt hooks use their current value.

This command is provided by Extended Tcl.

concat Join lists together

AUTH AUTH

SYNOPSIS

concat [arg arg ...]

DESCRIPTION

This command treats each argument as a list and concatenates them into a single list. It also eliminates leading and trailing spaces in the arg′s and adds a single separator space between arg′s. It permits any number of arguments. For example, the command:

```
concat a b {c d e} {f {g h}}
```

will return:

```
a b c d e f {g h}
```

as its result.

If no args are supplied, the result is an empty string.

RELATED INFORMATION

concatenate, join, lists

confadd Add items to a server′s configuration file

AUTH ACL

SYNOPSIS

```
confadd [-f fileName] var varName varValue varAuth
confadd [-f fileName] proc procName procArgs procAuth procBody
confadd [-f fileName] class className classFile classAuth
confadd [-f fileName] include fileName
```

DESCRIPTION

The confadd command adds items to the server′s configuration files. It is used in cases where automated additions to the configuration files are required. For instance, if a software package requires the definition of some sysctl commands, the package installation process can be modified to run a confadd to define the new commands and then issue an svcrestart to activate them.

The syntax of the confadd subcommands match those for the ″create proc″, ″create var″, ″create class″, and ″include″ commands. The confadd command does not support the addition of arbitrary Tcl constructs to the configuration files.

If the confadd adds a name that already appears in the configuration file, the existing definition is replaced with the new definition.

If no -f fileName argument is given, the modifications are made to the server′s default configuration file. Otherwise, the modifications are made to the file fileName. If fileName does not exist, an error is returned.

RELATED INFORMATION

create, include

confdelete Remove items from a server′s configuration file

AUTH ACL

SYNOPSIS

```
confdelete [-f fileName] var varName
confdelete [-f fileName] proc procName
confdelete [-f fileName] class className
confdelete [-f fileName] include fileName
```

DESCRIPTION

The confdelete command is the counterpart to the confadd command. It automates the process of removing definitions from a server′s configuration files. The object definitions removed by the confdelete command need not have been entered into the configuration file via the confadd command.

If the -f fileName argument is given, modifications are made to the configuration file fileName. Otherwise, changes are made to the default configuration file.

An error code is returned if the fileName argument (or the default configuration file) does not exist. If no matching object is found in the configuration file, no modification is made to the file.

RELATED INFORMATION

create, include

continue Skip to the next iteration of a loop

AUTH AUTH

SYNOPSIS

continue

DESCRIPTION

This command is typically invoked inside the body of a looping command such as for or foreach or while. It returns a TCL_CONTINUE code, which causes a continue exception to occur. The exception causes the current script to be aborted out to the the innermost containing loop command, which then continues with the next iteration of the loop. Catch exceptions are also handled in a few other situations, such as the catch command and the outermost scripts of procedure bodies.

RELATED INFORMATION

continue, iteration, loop

convertclock AUTH AUTH

SYNOPSIS

convertclock dateString [GMT | {}] [baseclock]

DESCRIPTION

Convert dateString to an integer clock value (see getclock). This command can parse and convert virtually any standard date and/or time string, which can include standard time zone mnemonics. If only a time is specified, the current date is assumed. If the string does not contain a time zone mnemonic, the local time zone is assumed, unless the GMT argument is specified, in which case the clock value is calculated assuming that the specified time is relative to Greenwich Mean Time.

If baseClock is specified, it should contain an integer clock value. Only the date in this value is used, not the time. This is useful for determining the time on a specific day or doing other date-relative conversions.

The character string consists of zero or more specifications of the following form:

- **time** : A time of day, which is of the form hh[:mm[:ss]] [meridian] [zone] or hhmm [meridian] [zone]. If no meridian is specified, hh is interpreted on a 24-hour clock.
- **date**: A specific month and day with optional year. The acceptable formats are mm/dd[/yy], yyyy/mm/dd, monthname dd[, yy], dd monthname [yy], and day, dd monthname yy. The default year is the current year. If the year is less then 100, then 1900 is added to it.
- **relative time**: A specification relative to the current time. The format is number unit; acceptable units are year, fortnight, month, week, day, hour, minute (or min), and second (or sec). The unit can be specified as a singular or plural, as in 3 weeks. These

modifiers may also be specified: tomorrow, yesterday, today, now, last, this, next, ago.

The actual date is calculated according to the following steps. First, any absolute date and/or time is processed and converted. Using that time as the base, day-of-week specifications are added. Next, relative specifications are used. If a date or day is specified, and no absolute or relative time is given, midnight is used. Finally, a correction is applied so that the correct hour of the day is produced after allowing for daylight savings time differences.

convertclock ignores case when parsing all words. The names of the months and days of the week can be abbreviated to their first three letters, with optional trailing period. Periods are ignored in any timezone or meridian values.

Note that convertclock will convert symbolic timezone names, but these are not standardized and there are conflicts with various parts of the world. Use GMT when trying to produce a portable time that can then be converted back to a numeric value.

The only dates in the range 1902 and 2037 may be converted. Some examples are:

```
convertclock "14 Feb 92"
convertclock "Feb 14, 1992 12:20 PM PST"
convertclock "12:20 PM Feb 14, 1992"
```

This command is provided by Extended Tcl.

copyfile **AUTH** ACL

SYNOPSIS

copyfile [-bytes num|-maxbytes num] fromFileId toFileId

DESCRIPTION

Copies the rest of the file specified by fromFileId, starting from its current position, to the file specified by toFileId, starting from its current position.

If -bytes is specified, then num bytes are copied. If less than num bytes are available, an error is returned. If -maxbytes is specified, then num bytes are copied but no error is returned if less are available.

The command returns the number of bytes that were copied.

The -bytes option is particularly useful for mixing binary data in with ASCII commands or data in a data stream.

This command is provided by Extended Tcl.

crange **AUTH** AUTH

SYNOPSIS

crange string firstExpr lastExpr

DESCRIPTION

Returns a range of characters from string starting at the character indexed by the expression firstExpr (zero-based) until the character indexed by the expression lastExpr.

If the expression firstExpr or lastExpr starts with the string end, then end is replaced with the index of the last character in the string. If the expression starts with len, then len is replaced with the length of the string.

This command is provided by Extended Tcl.

create Define new objects in the server

AUTH SYSTEM

SYNOPSIS

```
create proc procName procArgs procAuth procBody
create class className classFile [classAuth]
create var varName varValue [varAuth]
```

DESCRIPTION

The create command adds new object definitions to the server′s interpreter. It is normally executed from within the server′s configuration files.

CREATING PROCEDURES

The ″create proc″ command is similar to the Tcl proc command, except that an additional parameter is included which defines the procedure′s authorization callback. The authorization callback for the procedure is evaluated after the procedure′s variable frame is constructed, but before the body is evaluated. If the callback returns a Tcl error, access to the procedure is denied. Note that since the callback is executed after the procedure′s variable frame is constructed, arguments defined in the procedure are accessible from within its authorization callback. Conversely, global variables accessed from within the callback need to be declared as such using the global command. If the proc command is used in lieu of the ″create proc″ command, the procedure is assigned a default authorization level of SYSTEM. The server automatically disables all authorization checks while executing the body of a procedure for which a user is authorized to run. Thus, a procedure can evalute commands and/or access variables that the user is not able to do directly.

CREATING VARIABLES

The ″create var″ command defines read-only variables in the server. The varAuth parameter defines the authorization callback used to determine which users are able to read the value of the variable. The authorization callback is evaluated before the value of the variable is returned to the user. If the callback returns a Tcl error, access to the variable′s value is denied. If no varAuth parameter is provided, a default authorization of NONE is assigned meaning that anyone can read the contents of the variable (this is also the case with variables defined in ways other than ″create var″, e.g. with the set command). If a user attempts to modify the value of the variable via the set command or any other command, an error is returned. Note that the variable′s value can be modified within the context of a pre-defined procedure since all authorization checking is bypassed during the life of the procedure.

CREATING CLASSES

The ″create class″ command defines a new class within the server′s interpreter. A class is a mechanism for organizing a set of commands and variables into logical groups for clarity. The functionality of ″create class″ is similar to the include command, except that any procedures or variables defined in the class file via the ″create proc″ or ″create var″ commands are automatically prefixed by ″className:″. The prefixing allows easy identification of commands with other members of the same class and is a measure of avoiding name conflicts with existing commands. In addition, the use of a class allows for a single authorization callback to determine access to all variables and procedures defined in the class file. If the classAuth callback parameter is given, then that callback is logically-ORed with the object-specific callback when determining authorization to execute a command or read a variable. That is, authorization is granted if either the object callback or the class callback returns a normal Tcl result. Access to the object is denied only if both the class callback and the object callback return a Tcl error. The ″create class″ command is not recursive. That is, if a class is defined within a class file, they are treated as separate classes (and not sub-classes).

If a procedure or variable value definition requires access to another procedure or variable defined in the class file, the string ″CLASS::″ can be used in lieu of the actual class name. When reading the class file, the server automatically substitutes the current class name for all occurrences of ″CLASS:″. For example, the following class file defines a procedure which references a variable also defined in the class file:

```
create var testvar ″This is a test variable″

create proc show_test {} NONE {
        # Reference the class variable defined above
        puts ${CLASS::testvar}
}
```

RELATED INFORMATION

include, proc, rename, setauth, confadd, confdelete

csubstr **AUTH** AUTH

SYNOPSIS

csubstr string firstExpr lengthExpr

DESCRIPTION

Returns a range of characters from string starting at the character indexed by the expression firstExpr (zero-based) for lengthExpr characters.

If the expression firstExpr or lengthExpr starts with the string end, then end is replaced with the index of the last character in the string. If the expression starts with len, then len is replaced with the length of the string.

This command is provided by Extended Tcl.

ctoken **AUTH** AUTH

SYNOPSIS

ctoken strvar separators

DESCRIPTION

Parse a token out of a character string. The string to parse is contained in the variable named strvar. The string separators contains all of the valid separator characters for tokens in the string. All leading separators are skipped and the first token is returned. The variable strvar will be modified to contain the remainder of the string following the token.

This command is provided by Extended Tcl.

ctype **AUTH** AUTH

SYNOPSIS

ctype [-failindex var] class string

DESCRIPTION

ctype determines whether all characters in string are of the specified class. It returns 1 if they are all of class, and 0 if they are not, or if the string is empty. This command also provides another method (besides format and scan) of converting between an ASCII character and its numeric value. The following ctype commands are available:

- **ctype** [**-failindex var**] **alnum string**: Tests that all characters are alphabetic or numeric characters as defined by the character set.
- **ctype** [**-failindex var**] **alpha string**: Tests that all characters are alphabetic characters as defined by the character set.
- **ctype** [**-failindex var**] **ascii string**: Tests that all characters are an ASCII character (a non-negative number less than 0200).
- **ctype char number**: Converts the numeric value, string, to an ASCII character. Number must be in the range 0 through 255.
- **ctype** [**-failindex var**] **cntrl string**: Tests that all characters are ″control characters″ as defined by the character set.
- **ctype** [**-failindex var**] **digit string**: Tests that all characters are valid decimal digits, that meanse 0 through 9.
- **ctype** [**-failindex var**] **graph string**: Tests that all characters within are any character for which ctype print is true, except for space characters.
- **ctype** [**-failindex var**] **lower string**: Tests that all characters are lowercase letters as defined by the character set.
- **ctype ord character**: Convert a character into its decimal numeric value. The first character of the string is converted.
- **ctype** [**-failindex var**] **space string**: Tests that all characters are either a space, horizontal-tab, carriage return, newline, vertical-tab, or form-feed.
- **ctype** [**-failindex var**] **print string**: Tests that all characters are a space or any character for which ctype alnum or ctype punct is true or other ″printing character″ as defined by the character set.

- **ctype** [**-failindex var**] **punct string**: Tests that all characters are made up of any of the characters other than the ones for which alnum, cntrl, or space is true.
- **ctype** [**-failindex var**] **upper string**: Tests that all characters are uppercase letters as defined by the character set.
- **ctype** [**-failindex var**] **xdigit string**: Tests that all characters are valid hexadecimal digits, that is 0 through 9, a through f or A through F.

If -failindex is specified, then the index into string of the first character that did not match the class is returned in var.

This command is provided by Extended Tcl.

df Returns a list of filesystems

AUTH AUTH

SYNOPSIS

df [filesystem]

dup **AUTH** ACL

SYNOPSIS

dup fileId [targetFileId]

DESCRIPTION

Duplicate an open file. A new file id is opened that addresses the same file as fileId.

If targetFileId is specified, the the file is dup to this specified file id. Normally this is stdin, stdout, or stderr. The dup command will handle flushing output and closing this file. The target file should be open if its one of stdin, stdout, or stderr and the process is not going to do an execl. Otherwise internal C code that uses one of these files via direct access to stdio FILE struct may behave strangely or fail.

This command is provided by Extended Tcl.

echo Echoes a string

AUTH AUTH

SYNOPSIS

echo [str ...]

DESCRIPTION

Writes zero or more strings to standard output, followed by a newline.

This command is provided by Extended Tcl.

eof **AUTH** ACL

SYNOPSIS

eof fileId

DESCRIPTION

Returns 1 if an end-of-file condition has occurred on fileId, 0 otherwise. FileId must have been the return value from a previous call to open, or it may be stdin, stdout, or stderr to refer to one of the standard I/O channels.

RELATED INFORMATION

end, file

error **AUTH** AUTH

SYNOPSIS

error message [errorInfo] [errorCode]

DESCRIPTION

Returns a TCL_ERROR code, which causes command interpretation to be unwound. Message is a string that is returned to the application to indicate what went wrong.

If the info argument is provided and is non-empty, it is used to initialize the global variable errorInfo. errorInfo is used to accumulate a stack trace of what was in progress when an error occurred; as nested commands unwind, the Tcl interpreter adds information to errorInfo. If the info argument is present, it is used to initialize errorInfo and the first increment of unwind information will not be added by the Tcl interpreter. In other words, the command containing the error command will not appear in errorInfo; in its place will be info. This feature is most useful in conjunction with the catch command: if a caught error cannot be handled successfully, info can be used to return a stack trace reflecting the original point of occurrence of the error:

```
catch {...} errMsg
set savedInfo $errorInfo
   ...
error $errMsg $savedInfo
```

If the code argument is present, then its value is stored in the errorCode global variable. This variable is intended to hold a machine-readable description of the error in cases where such information is available; see the tclvars manual page for information on the proper format for the variable. If the code argument is not present, then errorCode is automatically reset to ″NONE″ by the Tcl interpreter as part of processing the error generated by the command.

RELALTED INFORMATION

error, errorCode, errorInfo

eval Evaluate a Tcl script

AUTH ACL

SYNOPSIS

eval arg [arg ...]

DESCRIPTION

Eval takes one or more arguments, which together comprise a Tcl script containing one or more commands. Eval concatenates all its arguments in the same fashion as the concat command, passes the concatenated string to the Tcl interpreter recursively, and returns the result of that evaluation (or any error generated by it).

RELATED INFORMATION

concatenate, evaluate, script

exec Invoke subprocess(es)

AUTH ACL

SYNOPSIS

exec [switches] arg [arg ...]

DESCRIPTION

This command treats its arguments as the specification of one or more subprocesses to execute. The arguments take the form of a standard shell pipeline where each arg becomes one word of a command, and each distinct command becomes a subprocess.

If the initial arguments to exec start with - then they are treated as command-line switches and are not part of the pipeline specification. The following switches are currently supported:

- **-keepnewline**: Retains a trailing newline in the pipeline′s output. Normally a trailing newline will be deleted.
- **--**: Marks the end of switches. The argument following this one will be treated as the first arg even if it starts with a -.

If an arg (or pair of arg′s) has one of the forms described below then it is used by exec to control the flow of input and output among the subprocess(es). Such arguments will not be passed to the subprocess(es). In forms such as ″< fileName″. fileName may either be in a separate argument from ″<″ or in the same argument with no intervening space (that means ″<fileName″).

- |: Separates distinct commands in the pipeline. The standard output of the preceding command will be piped into the standard input of the next command.
- |**&**: Separates distinct commands in the pipeline. Both standard output and standard error of the preceding command will be piped into the standard input of the next command. This form of redirection overrides forms such as 2> and >&.
- < **fileName**: The file named by fileName is opened and used as the standard input for the first command in the pipeline.
- <**@ fileId**: FileId must be the identifier for an open file, such as the return value from a previous call to open. It is used as the standard input for the first command in the pipeline. FileId must have been opened for reading.
- << **value**: Value is passed to the first command as its standard input.
- > **fileName**: Standard output from the last command is redirected to the file named fileName, overwriting its previous contents.
- **2**> **fileName**: Standard error from all commands in the pipeline is redirected to the file named fileName, overwriting its previous contents.
- >**& fileName**: Both standard output from the last command and standard error from all commands are redirected to the file named fileName, overwriting its previous contents.

- >> **fileName**: Standard output from the last command is redirected to the file named fileName, appending to it rather than overwriting it.
- **2**>> **fileName**: Standard error from all commands in the pipeline is redirected to the file named fileName, appending to it rather than overwriting it.
- >>**& fileName**: Both standard output from the last command and standard error from all commands are redirected to the file named fileName, appending to it rather than overwriting it.
- >@ **fileId**: FileId must be the identifier for an open file, such as the return value from a previous call to open. Standard output from the last command is redirected to fileId′s file, which must have been opened for writing.
- **2**>@ **fileId**: FileId must be the identifier for an open file, such as the return value from a previous call to open. Standard error from all commands in the pipeline is redirected to fileId′s file. The file must have been opened for writing.
- >**&@ fileId**: FileId must be the identifier for an open file, such as the return value from a previous call to open. Both standard output from the last command and standard error from all commands are redirected to fileId′s file. The file must have been opened for writing.

If standard output has not been redirected then the exec command returns the standard output from the last command in the pipeline. If any of the commands in the pipeline exit abnormally or are killed or suspended, then exec will return an error and the error message will include the pipeline′s output followed by error messages describing the abnormal terminations; the errorCode variable will contain additional information about the last abnormal termination encountered. If any of the commands writes to its standard error file and that standard error is not redirected, then exec will return an error; the error message will include the pipeline′s standard output, followed by messages about abnormal terminations (if any), followed by the standard error output.

If the last character of the result or error message is a newline then that character is normally deleted from the result or error message. This is consistent with other Tcl return values, which do not normally end with newlines. However, if -keepnewline is specified then the trailing newline is retained.

If standard input is not redirected with < or << or <@ then the standard input for the first command in the pipeline is taken from the application′s current standard input.

If the last arg is & then the pipeline will be executed in background. In this case the exec command will return a list whose elements are the process identifiers for all of the subprocesses in the pipeline. The standard output from the last command in the pipeline will go to the application′s standard output if it has not been redirected, and error output from all of the commands in the pipeline will go to the application′s standard error file unless redirected.

The first word in each command is taken as the command name; tilde-substitution is performed on it, and if the result contains no slashes then the directories in the PATH environment variable are searched for an executable by the given name. If the name contains a slash then it must refer to an executable reachable from the current directory. No ″glob″ expansion or other shell-like substitutions are performed on the arguments to commands.

RELATED INFORMATION

execute, pipeline, redirection, subprocess

execl

AUTH ACL

SYNOPSIS

execl [-argv0 argv0] prog [argList]

DESCRIPTION

Do an execl, replacing the current program (either Extended Tcl or an application with Extended Tcl embedded into it) with prog and passing the arguments in the list arglist.

The -argv0 options specifies that argv0 is to be passed to the program as argv [0] rather than prog.

If you are using execl in a Tk application and it fails, you may not do anything that accesses the X server or you will receive a BadWindow error from the X server. This includes executing the Tk version of the exit command. We suggest using the following command to abort Tk applications after an execl failure:

```
kill [id process]
```

This command is provided by Extended Tcl.

exit

Exits the interactive session or the procedure

AUTH AUTH

SYNOPSIS

exit [returnCode]

DESCRIPTION

Terminate the process, returning returnCode to the system as the exit status. If returnCode is not specified then it defaults to 0.

expr

AUTH AUTH

SYNOPSIS

expr arg [arg ...]

flush

Flush buffered output for a file

AUTH ACL

SYNOPSIS

flush fileId

DESCRIPTION

Flushes any output that has been buffered for fileId. FileId must have been the return value from a previous call to open, or it may be stdout or stderr to access one of the standard I/O streams; it must refer to a file that was opened for writing. The command returns an empty string.

RELATED INFORMATION

buffer, file, flush, output

fmtclock **AUTH** AUT

SYNOPSIS

fmtclock clockval [format] [GMT | {}]

DESCRIPTION

Converts a Unix integer time value, typically returned by getclock, convertclock, or the atime, mtime, or ctime options of the file command, to human-readable form. The format argument is a string that describes how the date and time are to be formatted. Field descriptors consist of a "%" followed by a field descriptor character. All other characters are copied into the result. Valid field descriptors are:

```
%% - Insert a %.
%a - Abbreviated weekday name.
%A - Full weekday name
%b - Abbreviated month name.
%B - Full month name.
%d - Day of month (01 - 31).
%D - Date as %m/%d/%y.
%e - Day of month (1-31), no leading zeros.
%h - Abbreviated month name.
%H - Hour (00 - 23).
%I - Hour (00 - 12).
%j - Day number of year (001 - 366).
%m - Month number (01 - 12).
%M - Minute (00 - 59).
%n - Insert a new line.
%p - AM or PM.
%r - Time as %I:%M:%S %p.
%R - Time as %H:%M.
%S - Seconds (00 - 59).
%t - Insert a tab.
%T - Time as %H:%M:%S.
%U - Week number of year (01 - 52), Sunday is the first
            day of the week.
%w - Weekday number (Sunday = 0).
%W - Week number of year (01 - 52), Monday is the first
            day of the week.
%x - Local specific date format.
%X - Local specific time format.
%y - Year within century (00 - 99).
%Y - Year as ccyy (e.g. 1990)
%Z - Time zone name.
```

If format is not specified, "%a %b %d %H:%M:%S %Z %Y" is used. If GMT is specified, the time will be formated as Greenwich Mean Time. If the argument is not specified or is empty, then the local timezone will be used as defined by the TIMEZONE environment variable.

This command is provided by Extended Tcl.

fcntl

AUTH ACL

SYNOPSIS

fcntl handle attribute [value]

DESCRIPTION

This command either sets or clears a file option or returns its current value. If value are not specified, then the current value of attribute is returned. The following attributes may be specified:

- **RDONLY**: The file is opened for reading only. (Get only)
- **WRONLY**: The file is opened for writing only. (Get only)
- **RDWR**: The file is opened for reading and writing. (Get only)
- **READ**: If the file is readable. (Get only).
- **WRITE**: If the file is writable. (Get only).
- **APPEND**: The file is opened for append-only writes. All writes will be forced to the end of the file.
- **NONBLOCK**: The file is to be accessed with nonblocking I/O. See the read system call for a description of how it affects the behavior of file reads.
- **CLOEXEC**: Close the file on an process exec. If the execl command or some other mechanism causes the process to do an exec, the file will be closed if this option is set.
- **NOBUF**: The file is not buffered. If set, then there no stdio buffering for the file.
- **LINEBUF**: Output the file will be line buffered. The buffer will be flushed when a newline is written, when the buffer is full, or when input is requested.

The APPEND, NONBLOCK, and CLOEXEC attributes may be set or cleared by specifying the attribute name and a value 1 to set the attribute and 0 to clear it.

The NOBUF and LINEBUF attributes may only be set (a value of 1) and only one of the options may be selected. Once set, it may not be changed. These options should be set before any I/O operations have been done on the file or data may be lost.

This command is provided by Extended Tcl.

file

Manipulate file names and attributes

AUTH ACL

SYNOPSIS

file option name [arg ...]

DESCRIPTION

This command provides several operations on a file′s name or attributes. Name is the name of a file; if it starts with a tilde, then tilde substitution is done before executing the command (see the manual entry for Tcl_TildeSubst for details). Option indicates what to do with the file name. Any unique abbreviation for option is acceptable. The valid options are:

- **file atime name**: Returns a decimal string giving the time at which file name was last accessed. The time is measured in the standard POSIX fashion as seconds from a fixed starting time (often January 1, 1970). If the file does not exist or its access time cannot be queried then an error is generated.
- **file dirname name**: Returns all of the characters in name up to but not including the last slash character. If there are no slashes in name then returns ″.″. If the last slash in name is its first character, then return ″/″.
- **file executable name**: Returns 1 if file name is executable by the current user, 0 otherwise.
- **file exists name**: Returns 1 if file name exists and the current user has search privileges for the directories leading to it, 0 otherwise.
- **file extension name**: Returns all of the characters in name after and including the last dot in the last element of name. If there is no dot in the last element of name then returns the empty string.
- **file isdirectory name**: Returns 1 if file name is a directory, 0 otherwise.
- **file isfile name**: Returns 1 if file name is a regular file, 0 otherwise.
- **file lstat name varName**: Same as stat option (see below) except uses the lstat kernel call instead of stat. This means that if name refers to a symbolic link the information returned in varName is for the link rather than the file it refers to. On systems that do not support symbolic links this option behaves exactly the same as the stat option.
- **file mtime name**: Returns a decimal string giving the time at which file name was last modified. The time is measured in the standard POSIX fashion as seconds from a fixed starting time (often January 1, 1970). If the file does not exist or its modified time cannot be queried then an error is generated.
- **file owned name**: Returns 1 if file name is owned by the current user, 0 otherwise.
- **file readable name**: Returns 1 if file name is readable by the current user, 0 otherwise.
- **file readlink name**: Returns the value of the symbolic link given by name (thta means the name of the file it points to). If name is not a symbolic link or its value cannot be read, then an error is returned. On systems that do not support symbolic links this option is undefined.
- **file rootname name**: Returns all of the characters in name up to but not including the last ″.″ character in the name. If name does not contain a dot, then returns name.
- **file size name**: Returns a decimal string giving the size of file name in bytes. If the file does not exist or its size cannot be queried then an error is generated.
- **file stat name varName**: Invokes the stat kernel call on name, and uses the variable given by varName to hold information returned

from the kernel call. VarName is treated as an array variable, and the following elements of that variable are set: atime, ctime, dev, gid, ino, mode, mtime, nlink, size, type, uid. Each element except type is a decimal string with the value of the corresponding field from the stat return structure; see the manual entry for stat for details on the meanings of the values. The type element gives the type of the file in the same form returned by the command file type. This command returns an empty string.

- **file tail name**: Returns all of the characters in name after the last slash. If name contains no slashes then returns name.
- **file type name**: Returns a string giving the type of file name, which will be one of file, directory, characterSpecial, blockSpecial, fifo, link, or socket.
- **file writable name**: Returns 1 if file name is writable by the current user, 0 otherwise.

RELATED INFORMATION

attributes, directory, file, name, stat

flock **AUTH** ACL

SYNOPSIS

flock [-read | -write] [-nowait] fileId [start] [length] [origin]

DESCRIPTION

This command places a lock on all or part of the file specified by fileId. The lock is either advisory or mandatory, depending on the mode bits of the file. The lock is placed beginning at relative byte offset start for length bytes. If start or length is omitted or empty, zero is assumed. If length is zero, then the lock always extents to end of file, even if the file grows. If origin is ″start″, then the offset is relative to the beginning of the file. If it is ″current″, it is relative to the current access position in the file. If it is ″end″, then it is relative to the end-of-file (a negative is before the EOF, positive is after). If origin is omitted, start is assumed.

The following options are recognized:

- **-read**: Place a read lock on the file. Multiple processes may be accessing the file with readlocks.
- **-write**: Place a write lock on the file. Only one process may be accessing a file if there is a write lock.
- **-nowait**: If specified, then the process will not block if the lock can not be obtained. With this option, the command returns 1 if the lock is obtained and 0 if it is not.

See your system′s fcntl system call documentation for full details of the behavior of file locking. If locking is being done on ranges of a file, it is best to use unbuffered file access (see the fcntl command).

This command is provided by Extended Tcl.

for ″For″ loop

AUTH ACL

SYNOPSIS

for start test next body

DESCRIPTION

For is a looping command, similar in structure to the C for statement. The start, next, and body arguments must be Tcl command strings, and test is an expression string. The for command first invokes the Tcl interpreter to execute start. Then it repeatedly evaluates test as an expression; if the result is non-zero it invokes the Tcl interpreter on body, then invokes the Tcl interpreter on next, then repeats the loop. The command terminates when test evaluates to 0. If a continue command is invoked within body then any remaining commands in the current execution of body are skipped; processing continues by invoking the Tcl interpreter on next, then evaluating test, and so on. If a break command is invoked within body or next, then the for command will return immediately. The operation of break and continue are similar to the corresponding statements in C. For returns an empty string.

RELATED INFORMATION

for, iteration, looping

foreach Iterate over all elements in a list

SYNOPSIS

foreach varName list body

DESCRIPTION

In this command varname is the name of a variable, list is a list of values to assign to varname, and body is a Tcl script. For each element of list (in order from left to right), foreach assigns the contents of the field to varname as if the lindex command had been used to extract the field, then calls the Tcl interpreter to execute body. The break and continue statements may be invoked inside body, with the same effect as in the for command. Foreach returns an empty string.

RELATED INFORMATION

foreach, iteration, list, looping

fork Forks the current Tcl process

AUTH ACL

SYNOPSIS

fork

DESCRIPTION

Fork the current Tcl process. Fork returns zero to the child process and the process number of the child to the parent process. If the fork fails, a Tcl error is generated.

If an execl is not going to be performed before the child process does output, or if a close and dup sequence is going to be performed on stdout or stderr, then a flush should be issued against stdout, stderr and any other open output file before doing the fork. Otherwise characters from the parent process pending in the buffers will be output by both the parent and child processes.

Note: If you are forking in a Tk based application you must execl before doing any window operations in the child or you will receive a BadWindow error from the X server.

This command is provided by Extended Tcl.

format Format a string in the style of sprintf

AUTH AUTH

SYNOPSIS

format formatString [arg arg ...]

DESCRIPTION

This command generates a formatted string in the same way as the ANSI C sprintf procedure (it uses sprintf in its implementation). FormatString indicates how to format the result, using % conversion specifiers as in sprintf and the additional arguments, if any, provide values to be substituted into the result. The return value from format is the formatted string.

DETAILS ON FORMATTING

The command operates by scanning formatString from left to right. Each character from the format string is appended to the result string unless it is a percent sign. If the character is a % then it is not copied to the result string. Instead, the characters following the % character are treated as a conversion specifier. The conversion specifier controls the conversion of the next successive arg to a particular format and the result is appended to the result string in place of the conversion specifier. If there are multiple conversion specifiers in the format string, then each one controls the conversion of one additional arg. The format command must be given enough args to meet the needs of all of the conversion specifiers in formatString.

Each conversion specifier may contain up to six different parts: an XPG3 position specifier, a set of flags, a minimum field width, a precision, a length modifier, and a conversion character. Any of these fields may be omitted except for the conversion character. The fields that are present must appear in the order given above. The paragraphs below discuss each of these fields in turn.

If the % is followed by a decimal number and a ″$″, as in ″%2$d″, then the value to convert is not taken from the next sequential argument. Instead, it is taken from the argument indicated by the number, where 1 corresponds to the first arg. If the conversion specifier requires multiple arguments because of * characters in the specifier then successive arguments are used, starting with the argument given by the number. This follows the XPG3 conventions for positional specifiers. If there are any positional specifiers in formatString then all of the specifiers must be positional.

The second portion of a conversion specifier may contain any of the following flag characters, in any order:

- **-**: Specifies that the converted argument should be left-justified in its field (numbers are normally right-justified with leading spaces if needed).

- **+**: Specifies that a number should always be printed with a sign, even if positive.
- **space**: Specifies that a space should be added to the beginning of the number if the first character is not a sign.
- **0**: Specifies that the number should be padded on the left with zeroes instead of spaces.
- #: Requests an alternate output form. For o and O conversions it guarantees that the first digit is always 0. For x or X conversions, 0x or 0X (respectively) will be added to the beginning of the result unless it is zero. For all floatingpoint conversions (e, E, f, g, and G) it guarantees that the result always has a decimal point. For g and G conversions it specifies that trailing zeroes should not be removed.

The third portion of a conversion specifier is a number giving a minimum field width for this conversion. It is typically used to make columns line up in tabular printouts. If the converted argument contains fewer characters than the minimum field width then it will be padded so that it is as wide as the minimum field width. Padding normally occurs by adding extra spaces on the left of the converted argument, but the 0 and - flags may be used to specify padding with zeroes on the left or with spaces on the right, respectively. If the minimum field width is specified as * rather than a number, then the next argument to the format command determines the minimum field width; it must be a numeric string.

The fourth portion of a conversion specifier is a precision, which consists of a period followed by a number. The number is used in different ways for different conversions. For e, E, and f conversions it specifies the number of digits to appear to the right of the decimal point. For g and G conversions it specifies the total number of digits to appear, including those on both sides of the decimal point (however, trailing zeroes after the decimal point will still be omitted unless the # flag has been specified). For integer conversions, it specifies a minimum number of digits to print (leading zeroes will be added if necessary). For s conversions it specifies the maximum number of characters to be printed; if the string is longer than this then the trailing characters will be dropped. If the precision is specified with * rather than a number then the next argument to the format command determines the precision; it must be a numeric string.

The fifth part of a conversion specifier is a length modifier, which must be h or l. If it is h it specifies that the numeric value should be truncated to a 16-bit value before converting. This option is rarely useful. The l modifier is ignored.

The last thing in a conversion specifier is an alphabetic character that determines what kind of conversion to perform. The following conversion characters are currently supported:

- **d**: Convert integer to signed decimal string.
- **u**: Convert integer to unsigned decimal string.

- **i**: Convert integer to signed decimal string; the integer may either be in decimal, in octal (with a leading 0) or in hexadecimal (with a leading 0x).
- **o**: Convert integer to unsigned octal string.
- **x** or **X**: Convert integer to unsigned hexadecimal string, using digits ″0123456789abcdef″ for x and ″0123456789ABCDEF″ for X).
- **c**: Convert integer to the 8-bit character it represents.
- **s**: No conversion; just insert string.
- **f**: Convert floating-point number to signed decimal string of the form xx.yyy, where the number of y′s is determined by the precision (default:6). If the precision is 0 then no decimal point is output.
- **e** or **e**: Convert floating-point number to scientific notation in the form x.yyye+-zz, where the number of y′s is determined by the precision (default:6). If the precision is 0 then no decimal point is output. If the E form is used then E is printed instead of e.
- **g** or **G**: If the exponent is less than -4 or greater than or equal to the precision, then convert floating-point number as for %e or %E. Otherwise convert as for %f. Trailing zeroes and a trailing decimal point are omitted.
- **%**: No conversion: just insert %.

For the numerical conversions the argument being converted must be an integer or floating-point string; format converts the argument to binary and then converts it back to a string according to the conversion specifier.

DIFFERENCES FROM ANSI SPRINTF The behavior of the format command is the same as the ANSI C sprintf procedure except for the following differences:

1. %p and %n specifiers are not currently supported.
2. For %c conversions the argument must be a decimal string, which will then be converted to the corresponding character value.
3. The l modifier is ignored; integer values are always converted as if there were no modifier present and real values are always converted as if the l modifier were present (i.e. type double is used for the internal representation). If the h modifier is specified then integer values are truncated to short before conversion.

RELATED INFORMATION

conversion specifier, format, sprintf, string, substitution

frename **AUTH** ACL

SYNOPSIS

frename oldPath newPath

DESCRIPTION

Renames oldPath to newPath. This command does not support renaming across file systems unless the rename system call supports them. If renaming accross file systems is desired, exec the mv command.

This command is provided by Extended Tcl.

fscheck Returns percentage of usage of filesystems

SYNOPSIS

fscheck number

```
fscheck 80
/usr ==> 99.73% Used, 3108 KB Free
/home ==> 93.84% Used, 2272 KB Free
/home/ftp ==> 82.41% Used, 73508 KB Free
```

AUTH AUTH

fstat Obtain status information about an open file.

AUTH ACL

SYNOPSIS

fstat fileId [item]|]stat arrayVar]

DESCRIPTION Obtain status information about an open file.

The following keys are used to identify data items:

- **atime**: The time of last access.
- **ctime**: The time of last file status change
- **dev**: The device containing a directory for the file. This value uniquely identifies the file system that contains the file.
- **gid**: The group ID of the file′s group.
- ino: The inode number. This field uniquely identifies the file in a given file system.
- **mode**: The mode of the file (see the mknod system call).
- **mtime**: Time when the data in the file was last modified.
- **nlink**: The number of links to the file.
- **size**: The file size in bytes.
- **tty**: If the file is associated with a terminal, then 1 otherwise 0.
- **type**: The type of the file in symbolic form, which is one of the following values: file, directory, characterSpecial, blockSpecial, fifo, link, or socket.
- **uid**: The user ID of the file′s owner.

If one of these keys is specified as item, then that data item is returned.

If stat arrayvar is specified, then the information is returned in the array arrrayvar. Each of the above keys indexes an element of the array containing the data.

If only fileId is specified, the command returns the data as a keyed list.

The following values may be returned only if explicitly asked for, it will not be returned with the array or keyed list forms:

- **remotehost**: If fileId is a TCP/IP socket connection, then a list is returned with the first element being the remote host IP address. If the remote host name can be found, it is returned as the second element of the list. The remote host IP port number is returned as the this element.
- **localhost**: If fileId is a TCP/IP socket connection, then a list is returned with the first element being the local host IP address. If the local host name can be found, it is returned as the second element of the list. The local host IP port number is returned as the this element.

This command is provided by Extended Tcl.

funlock **AUTH** ACL

SYNOPSIS

funlock fileId [start] [length] [origin]

DESCRIPTION

Remove a locked from a file that was previously placed with the flock command. The arguments are the same as for the flock command, see that command for more details.

This command is provided by Extended Tcl.

getauth Display the authorization callbacks for an object

AUTH ACL

SYNOPSIS

getauth -cmd | -var | -class objName

DESCRIPTION

The getauth command retrieves the authorization callback for the given object. If the object is part of a class, both the object′s callback and the class callback are displayed.

RELATED INFORMATION

setauth, checkauth, create

gets Read a line from a file

AUTH ACL

SYNOPSIS

gets fileId [varName]

DESCRIPTION

This command reads the next line from the file given by fileId and discards the terminating newline character. If varName is specified then the line is placed in the variable by that name and the return value is a count of the number of characters read (not including the newline). If the end of the file is reached before reading any characters then -1 is returned and varName is set to an empty string. If varName is not specified then the return value will be the line (minus the newline character) or an empty string if the end of the file is reached before reading any characters. An empty string will also be returned if a line contains no characters except the newline, so eof may have to be used to determine what really happened. If the last character in the file is not a newline character then gets behaves as if there were an additional newline character at the end of the file. FileId must be stdin or the return value from a previous call to open; it must refer to a file that was opened for reading. Any existing end-of-file or error condition on the file is cleared at the beginning of the gets command.

RELATED INFORMATION

file, line, read

getclock Returns number of seconds, from date 1-1-70 to now.

AUTH AUTH

SYNOPSIS

getclock

DESCRIPTION

Return the current date and time as a system-dependent integer value. The unit of the value is seconds, allowing it to be used for relative time calculations.

This command is provided by Extended Tcl.

glob Return names of files that match patterns

AUTH ACL

SYNOPSIS

glob [switches] name [name ...]

DESCRIPTION

This command performs file name ″globbing″ in a fashion similar to the csh shell. It returns a list of the files whose names match any of the pattern arguments.

If the initial arguments to glob start with - then they are treated as switches. The following switches are currently supported:

- **-nocomplain**: Allows an empty list to be returned without error; without this switch an error is returned if the result list would be empty.
- **--**: Marks the end of switches. The argument following this one will be treated as a pattern even if it starts with a -.

The pattern arguments may contain any of the following special characters:

- **?**: Matches any single character.
- *****: Matches any sequence of zero or more characters.
- [**chars**]: Matches any single character in chars. If chars contains a sequence of the form a-b then any character between a and b (inclusive) will match.
- **x**: Matches the character x.
- {**a,b,...**}: Matches any of the strings a, b, etc.

As with csh, a ″.″ at the beginning of a file′s name or just after a ″/″ must be matched explicitly or with a {} construct. In addition, all ″/″ characters must be matched explicitly.

If the first character in a pattern is ″~″ then it refers to the home directory for the user whose name follows the ″~″. If the ″~″ is followed immediately by ″/″ then the value of the HOME environment variable is used.

The glob command differs from csh globbing in two ways. First, it does not sort its result list (use the lsort command if you want the list sorted). Second, glob only returns the names of files that actually exist; in csh no check for existence is made unless a pattern contains a [, *, or [] construct.

RELATED INFORMATION

exist, file, glob, pattern

global Access global variables

AUTH AUTH

SYNOPSIS

global varName [varName ...]

DESCRIPTION

This command is ignored unless a Tcl procedure is being interpreted. If so then it declares the given varname′s to be global variables rather than local ones. For the duration of the current procedure (and only while executing in the current procedure), any reference to any of the varnames will refer to the global variable by the same name.

RELATED INFORMATION

global, procedure, variable

help Sysctld help facility

AUTH AUTH

SYNOPSIS

```
help [-commands] [cmdName]

help:setPath dir [dir ...]
help:addPath dir [dir ...]
help:delPath dir [dir ...]
help:getPath

help:checkCommands
```

DESCRIPTION

The help facility is a mechanism for displaying information about command usage. The facility consists of a main help command, called help, and a set of utility functions for modifying the location where help files are available.

If the -commands argument is given, a list of all commands the user is authorized to execute is returned. Otherwise, the help page for the command name cmdName is displayed.

The help:* utility routines are used to add additional help directories to the standard help search path. The help search path is used by the server to locate help pages. The help path is a list of directories. When searching for the help page for a topic, the server starts at the first directory of the help path. If a file named by the help page exists in that directory, it is displayed. If no file is found and the help directory contains subdirectories, each subdirectory is also searched. If no matching file is found, the next directory in the help path is searched. If no page can be found in any help directory, an error is returned.

The help:getPath routine displays the current help path. The help:addPath, help:delPath, and help:setPath procedures all are used to modify the help path. The addPath command adds additional directories to the help path. delPath removes the given directories from the help path. The setPath command re-initializes the help path to the given value.

Typically, the help:addPath command is used to expand the help search path to include application-specific directories. This prevents the administrator from having to manage one giant help directory. In most cases, a line such as this can be included in an application′s class definition file or include file:

help:addPath <app-help-directory>

history Manipulate the history list

AUTH AUTH

SYNOPSIS

history [option] [arg arg ...]

DESCRIPTION

The history command performs one of several operations related to recently-executed commands recorded in a history list. Each of these recorded commands is referred to as an ″event″. When specifying an event to the history command, the following forms may be used:

1. A number: if positive, it refers to the event with that number (all events are numbered starting at 1). If the number is negative, it selects an event relative to the current event (-1 refers to the previous event, -2 to the one before that, and so on).
2. A string: selects the most recent event that matches the string. An event is considered to match the string either if the string is the same as the first characters of the event, or if the string matches the event in the sense of the string match command.

The history command can take any of the following forms:

- **history**: Same as history info, described below.
- **history add command** [**exec**]: Adds the command argument to the history list as a new event. If exec is specified (or abbreviated) then the command is also executed and its result is returned. If exec is not specified then an empty string is returned as result.
- **history change newValue** [**event**]: Replaces the value recorded for an event with newValue. Event specifies the event to replace and defaults to the current event (not event -1). This command is intended for use in commands that implement new forms of history substitution and wish to replace the current event (which invokes the substitution) with the command created through substitution. The return value is an empty string.

- **history event** [**event**]: Returns the value of the event given by event. Event defaults to -1. This command causes history revision to occur: see below for details.
- **history info** [**count**]: Returns a formatted string (intended for humans to read) giving the event number and contents for each of the events in the history list except the current event. If count is specified then only the most recent count events are returned.
- **history keep count**: This command may be used to change the size of the history list to count events. Initially, 20 events are retained in the history list. This command returns an empty string.
- **history nextid**: Returns the number of the next event to be recorded in the history list. It is useful for things like printing the event number in command-line prompts.
- **history redo** [**event**]: Re-executes the command indicated by event and return its result. Event defaults to -1. This command results in history revision: see below for details.
- **history substitute old new** [**event**]: Retrieves the command given by event (-1 by default), replace any occurrences of old by new in the command (only simple character equality is supported; no wild cards), execute the resulting command, and return the result of that execution. This command results in history revision: see below for details.
- **history words selector** [**event**]: Retrieves from the command given by event (-1 by default) the words given by selector, and return those words in a string separated by spaces. The selector argument has three forms. If it is a single number then it selects the word given by that number (0 for the command name, 1 for its first argument, and so on). If it consists of two numbers separated by a dash, then it selects all the arguments between those two. Otherwise selector is treated as a pattern; all words matching that pattern (in the sense of string match) are returned. In the numeric forms $ may be used to select the last word of a command. For example, suppose the most recent command in the history list is:

  ```
  format  {%s is %d years old}
  Alice [expr $ageInMonths/12]
  ```

  Below are some history commands and the results they would produce:

  ```
  history words $ [expr $ageInMonths/12]
  history words 1-2{%s is %d years  old} Alice
  history words *a*o*{%s
    is %d years old} [expr $ageInMonths/12]
  ```

  History words results in history revision: see below for details.

HISTORY REVISION

The history options event, redo, substitute, and words result in ″history revision″. When one of these options is invoked then the current event is modified to eliminate the history command and replace it with the result of the history command. For example, suppose that the most recent command in the history list is:

```
set a [expr $b+2]
```

and suppose that the next command invoked is one of the ones on the left side of the table below. The command actually recorded in the history event will be the corresponding one on the right side of the table.

```
history redo    set a [expr $b+2]
history s a b   set b [expr $b+2]
set c [history w 2]set c [expr $b+2]
```

History revision is needed because event specifiers like -1 are only valid at a particular time: once more events have been added to the history list a different event specifier would be needed. History revision occurs even when history is invoked indirectly from the current event (for example a user types a command that invokes a Tcl procedure that invokes history): the top-level command whose execution eventually resulted in a history command is replaced. If you wish to invoke commands like history words without history revision, you can use history event to save the current history event and then use history change to restore it later.

RELATED INFORMATION

event, history, record, revision

id

AUTH ACL

SYNOPSIS

id arg [arg..]

DESCRIPTION

This command provides a means of getting, setting and converting user, group and process ids. The id command has the following options:

- **id user** [**name**]
- **id userid** [**uid**]: Set the real and effective user ID to name or uid, if the name (or uid) is valid and permissions allow it. If the name (or uid) is not specified, the current name (or uid) is returned.
- **id convert userid uid**
- **id convert user name**: Convert a user ID number to a user name, or vice versa.
- **id group** [**name**]
- **id groupid** [**gid**]: Set the real and effective group ID to name or gid, if the name (or gid) is valid and permissions allow it. If the group name (or gid) is not specified, the current group name (or gid) is returned.
- **id groups**
- **id groupids**: Return the current group access list of the process. The option groups returns group names and groupids returns id numbers.
- **id convert groupid gid**
- **id convert group name**: Convert a group ID number to a group name, or vice versa.

- **id effective user**
- **id effective userid**: Return the effective user name, or effective user ID number, respectively.
- **id effective group**
- **id effective groupid**: Return the effective group name, or effective group ID number, respectively.
- **id effective groupids**: Return all of the groupids the user is a member of.
- **id host**: Return the hostname of the system the program is running on.
- **id process**: Return the process ID of the current process.
- **id process parent**: Return the process ID of the parent of the current process.
- **id process group**: Return the process group ID of the current process.
- **id process group set**: Set the process group ID of the current process to its process ID.
- **id host**: Returns the standard host name of the machine the process is executing on.

This command is provided by Extended Tcl.

if Execute scripts conditionally

AUTH AUTH

SYNOPSIS

if expr1 [then] body1 elseif expr2 [then] body2 elseif ... [else] [bodyN]

DESCRIPTION

The if command evaluates expr1 as an expression (in the same way that expr evaluates its argument). The value of the expression must be a Boolean (a numeric value, where 0 is false and anything is true, or a string value such as true or yes for true and false or no for false); if it is true then body1 is executed by passing it to the Tcl interpreter. Otherwise expr2 is evaluated as an expression and if it is true then body2 is executed, and so on. If none of the expressions evaluates to true then bodyN is executed. The then and else arguments are optional ″noise words″ to make the command easier to read. There may be any number of elseif clauses, including zero. BodyN may also be omitted as long as else is omitted too. The return value from the command is the result of the body script that was executed, or an empty string if none of the expressions was non-zero and there was no bodyN.

RELATED INFORMATION

Boolean, conditional, else, false, if, true

include Include additional configuration files

AUTH SYSTEM

SYNOPSIS

include fileName

DESCRIPTION

Similar to the source Tcl command in that it reads file fileName and passes the contents to the Tcl interpreter as a script to evaluate in the normal fashion.

The only difference is that if an error occurs while evaluating the contents of the file during startup or restart, the result of the error is logged to the server log file and no error is returned. This allows the startup sequence to continue even in cases where one of the included files contains an error.

RELATED INFORMATION

source, confadd, confdelete

incr Increment the value of a variable

AUTH AUTH

SYNOPSIS

incr varName [increment]

info Return information about the state of the Tcl interpreter

SYNOPSIS

info option [arg arg ...]

DESCRIPTION

This command provides information about various internals of the Tcl interpreter. The legal option′s (which may be abbreviated) are:

- **info args procname**: Returns a list containing the names of the arguments to procedure procname, in order. Procname must be the name of a Tcl command procedure.
- **info body procname**: Returns the body of procedure procname. Procname must be the name of a Tcl command procedure.
- **info cmdcount**: Returns a count of the total number of commands that have been invoked in this interpreter.
- **info commands** [**pattern**]: If pattern is not specified, returns a list of names of all the Tcl commands, including both the built-in commands written in C and the command procedures defined using the proc command. If pattern is specified, only those names matching pattern are returned. Matching is determined using the same rules as for string match.
- **info complete command**: Returns 1 if command is a complete Tcl command in the sense of having no unclosed quotes, braces, brackets or array element names. If the command does not appear to be complete i then 0 is returned. This command is typically used in line-oriented input environments to allow users to type in commands that span multiple lines; if the command is not complete, the script can delay evaluating it until additional lines have been typed to complete the command.
- **info default procname arg varname**: Procname must be the name of a Tcl command procedure and arg must be the name of an argument to that procedure. If arg does not have a default value

then the command returns 0. Otherwise it returns 1 and places the default value of arg into variable varname.

- **info exists varName**: Returns 1 if the variable named varName exists in the current context (either as a global or local variable), returns 0 otherwise.
- **info globals** [**pattern**]: If pattern is not specified, returns a list of all the names of currently-defined global variables. If pattern is specified, only those names matching pattern are returned. Matching is determined using the same rules as for string match.
- **info level** [**number**]: If number is not specified, this command returns a number giving the stack level of the invoking procedure, or 0 if the command is invoked at toplevel. If number is specified, then the result is a list consisting of the name and arguments for the procedure call at level number on the stack. If number is positive then it selects a particular stack level (1 refers to the top-most active procedure, 2 to the procedure it called, and so on); otherwise it gives a level relative to the current level (0 refers to the current procedure, -1 to its caller, and so on). See the uplevel command for more information on what stack levels mean.
- **info library**: Returns the name of the library directory in which standard Tcl scripts are stored. The default value for the library is compiled into Tcl, but it may be overridden by setting the TCL_LIBRARY environment variable. If there is no TCL_LIBRARY variable and no compiled-in value then and error is generated. See the library manual entry for details of the facilities provided by the Tcl script library. Normally each application will have its own application-specific script library in addition to the Tcl script library; I suggest that each application set a global variable with a name like $app_library (where app is the application′s name) to hold the location of that application′s library directory.
- **info locals** [**pattern**]: If pattern is not specified, returns a list of all the names of currently-defined local variables, including arguments to the current procedure, if any. Variables defined with the global and upvar commands will not be returned. If pattern is specified, only those names matching pattern are returned. Matching is determined using the same rules as for string match.
- **info patchlevel**: Returns a decimal integer giving the current patch level for Tcl. The patch level is incremented for each new release or patch, and it uniquely identifies an official version of Tcl.
- **info procs** [**pattern**]: If pattern is not specified, returns a list of all the names of Tcl command procedures. If pattern is specified, only those names matching pattern are returned. Matching is determined using the same rules as for string match.
- **info script**: If a Tcl script file is currently being evaluated (that means there is a call to Tcl_EvalFile active or there is an active invocation of the source command), then this command returns the name of the innermost file being processed. Otherwise the command returns an empty string.

- **info tclversion**: Returns the version number for this version of Tcl in the form x.y, where changes to x represent major changes with probable incompatibilities and changes to y represent small enhancements and bug fixes that retain backward compatibility.
- **info vars** [**pattern**]: If pattern is not specified, returns a list of all the names of currently-visible variables, including both locals and currently-visible globals. If pattern is specified, only those names matching pattern are returned. Matching is determined using the same rules as for string match.

RELATED INFORMATION

command, information, interpreter, level, procedure, variable

infox

AUTH ACL

SYNOPSIS

infox option

DESCRIPTION

Return information about Extended Tcl, or the current application. The following infox command options are available:

- **version**: Return the version number of Extended Tcl. The version number for Extended Tcl is generated by combining the base version of the standard Tcl code with a letter indicating the version of Extended Tcl being used. This is the documentation for version 7.4a.
- **patchlevel**: Return the patchlevel for Extended Tcl.
- **have_flock**: Return 1 if the flock command defined, 0 if it is not available.
- **have_fsync**: Return 1 if the fsync system call is available and the sync command will sync individual files. 0 if it is not available and the sync command will always sync all file buffers.
- **have_msgcats**: Return 1 if XPG message catalogs are available, 0 if they are not. The catgets is designed to continue to function without message catalogs, always returning the default string.
- **have_posix_signals**: Return 1 if Posix signals are available (block and unblock options available for the signal command). 0 is returned if Posix signals are not available.
- **have_sockets**: Return 1 if sockets are available (server_* suite and fstat localhost and remotehost options). 0 is returned if sockets are not available.
- **appname**: Return the symbolic application name of the current application linked with the Extended Tcl library. The C variable tclAppName must be set by the application to return an application specific value for this variable.
- **applongname**: Return a natural language name for the current application. The C variable tclLongAppName must be set by the application to return an application specific value for this variable.
- **appversion**: Return the version number for the current application. The C variable tclAppVersion must be set by the application to return an application-specific value for this variable.

- **apppatchlevel**: Return the patchlevel for the current application. The C variable tclAppPatchlevel must be set by the application to return an application-specific value for this variable.

This command is provided by Extended Tcl.

join Create a string by joining together list elements

AUTH AUTH

SYNOPSIS

join list [joinString]

DESCRIPTION

The list argument must be a valid Tcl list. This command returns the string formed by joining all of the elements of list together with joinString separating each adjacent pair of elements. The joinString argument defaults to a space character.

RELATED INFORMATION

element, join, list, separator

keyldel **AUTH** AUTH

SYNOPSIS

keyldel listvar key

DESCRIPTION

Delete the field specified by key from the keyed list in the variable listvar. This removes both the key and the value from the keyed list.

This command is provided by Extended Tcl.

keylget **AUTH** AUTH

SYNOPSIS

keylget listvar [key] [retvar | {}]

DESCRIPTION

Return the value associated with key from the keyed list in the variable listvar. If retvar is not specified, then the value will be returned as the result of the command. In this case, if key is not found in the list, an error will result.

If retvar is specified and key is in the list, then the value is returned in the variable retvar and the command returns 1 if the key was present within the list. If key is not in the list, the command will return 0, and retvar will be left unchanged. If {} is specified for retvar, the value is not returned, allowing the Tcl programmer to determine if a key is present in a keyed list without setting a variable as a side-effect.

If key is omitted, then a list of all the keys in the keyed list is returned.

This command is provided by Extended Tcl.

keylkeys **AUTH** AUTH

SYNOPSIS

keylkeys listvar [key]

DESCRIPTION

Return the a list of the keyes in the keyed list in the variable listvar. If keys is specified, then it is the name of a key field who′s subfield keys are to be retrieve.

This command is provided by Extended Tcl.

keylset **AUTH** AUTH

SYNOPSIS

keylset listvar key value [key value...]

DESCRIPTION

Set the value associated with key, in the keyed list contained in the variable listvar, to value. If listvar does not exists, it is created. If key is not currently in the list, it will be added. If it already exists, value replaces the existing value. Multiple keywords and values may be specified, if desired.

This command is provided by Extended Tcl.

kill **AUTH** ACL

SYNOPSIS

kill [-pgroup] [signal] idlist

DESCRIPTION

Send a signal to the each process in the list idlist, if permitted. Signal, if present, is the signal number or the symbolic name of the signal, see the signal system call manual page. The leading ″SIG″ is optional when the signal is specified by its symbolic name. The default for signo is 15, SIGTERM.

If -pgroup is specified, the numbers in idlist are take as process group ids and the signal is sent to all of the process in that process group. A process group id of 0 specifies the current process group.

This command is provided by Extended Tcl.

lappend Append list elements onto a variable

AUTH AUTH

SYNOPSIS

lappend varName value [value ...]

DESCRIPTION

This command treats the variable given by varName as a list and appends each of the value arguments to that list as a separate element, with spaces between elements. If varName does not exist, it is created as a list with elements given by the value arguments. Lappend is similar to append except that the values are appended as list elements rather than raw text. This command provides a relatively efficient way to build up large lists. For example, ″lappend a $b″ is much more efficient than ″set a [concat $a [list $b]]″. when $a is long.

RELATED INFORMATION

append, element, list, variable

lassign **AUTH** AUTH

SYNOPSIS

lassign list varname [varname..]

DESCRIPTION

Assign successive elements of a list to specified variables. If there are more variable names than fields, the remaining variables are set to the empty string. If there are more elements than variables, a list of the unassigned elements is returned.

For example,

```
lassign {dave 100 200 {Dave Foo}} name uid gid longName
```

Assigns name to ″dave″, uid to ″100″, gid to ″200″, and longName to ″Dave Foo″.

This command is provided by Extended Tcl.

lempty **AUTH** ACL

SYNOPSIS

lempty list

DESCRIPTION

Determine if the specified list is empty. If empty, 1 is returned, otherwise, 0 is returned. This command is an alternative to comparing a list to an empty string.

This command is provided by Extended Tcl.

lgets **AUTH** ACL

SYNOPSIS

lgets fileId [varName]

DESCRIPTION

Reads the next Tcl list from the file given by fileId and discards the terminating newline character. This command differs from the gets command, in that it reads Tcl lists rather than lines. If the list contains a newline, then that newline will be returned as part of the result. Only a newline not quoted as part of the list indicates the end of the list. There is no corresponding command for outputting lists, as puts will do this correctly. If varName is specified, then the line is placed in the variable by that name and the return value is a count of the number of characters read (not including the newline). If the end of the file is reached before reading any characters then -1 is returned and varName is set to an empty string. If varName is not specified then the return value will be the line (minus the newline character) or an empty string if the end of the file is reached before reading any characters. An empty string will also be returned if a line contains no characters except the newline, so eof may have to be used to determine what really happened.

This command is provided by Extended Tcl.

lindex Retrieve an element from a list

AUTH AUTH

SYNOPSIS

lindex list index

DESCRIPTION

This command treats list as a Tcl list and returns the index′th element from it (0 refers to the first element of the list). In extracting the element, lindex observes the same rules concerning braces and quotes and backslashes as the Tcl command interpreter; however, variable substitution and command substitution do not occur. If index is negative or greater than or equal to the number of elements in value, then an empty string is returned. If index has the value end, it refers to the last element in the list.

RELATED INFORMATION

element, index, list

link **AUTH** ACL

SYNOPSIS

link [-sym] srcpath destpath

DESCRIPTION

Create a directory entry, destpath, linking it to the existing file, srcpath. If -sym is specified, a symbolic link, rather than a hard link, is created. (The -sym option is only available on systems that support symbolic links.)

This command is provided by Extended Tcl.

linsert Insert elements into a list

AUTH AUTH

SYNOPSIS

linsert list index element [element ...]

DESCRIPTION

This command produces a new list from list by inserting all of the element arguments just before the indexth element of list. Each element argument will become a separate element of the new list. If index is less than or equal to zero, then the new elements are inserted at the beginning of the list. If index has the value end, or if it is greater than or equal to the number of elements in the list, then the new elements are appended to the list.

RELATED INFORMATION

element, insert, list

list Create a list

AUTH AUTH

SYNOPSIS

list [arg arg ...]

DESCRIPTION

This command returns a list comprised of all the args, or an empty string if no args are specified. Braces and backslashes get added as necessary, so that the index command may be used on the result to re-extract the original arguments, and also so that eval may be used to execute the resulting list, with arg1 comprising the command's name and the other args comprising its arguments. List produces slightly different results than concat: concat removes one level of grouping before forming the list, while list works directly from the original arguments. For example, the command:

```
list a b {c d e} {f {g h}}
```

will return

```
a b {c d e} {f {g h}}
```

while concat with the same arguments will return

```
a b c d e f {g h}
```

RELATED INFORMATION

element, list

listfs Return a list of all local filesystems

AUTH ACL

SYNOPSIS

listfs

DESCRIPTION

The listfs command returns a list of all local filesystems currently mounted on the local host.

llength Count the number of elements in a list

AUTH AUTH

SYNOPSIS

llength list

DESCRIPTION

Treats list as a list and returns a decimal string giving the number of elements in it.

RELATED INFORMATION

element, list, length

lmatch **AUTH** AUTH

SYNOPSIS

lmatch [mode] list pattern

DESCRIPTION

Search the elements of list, returning a list of all elements matching pattern. If none match, an empty list is returned.

The mode argument indicates how the elements of the list are to be matched against pattern and it must have one of the following values:

- **-exact**: The list element must contain exactly the same string as pattern.
- **-glob Pattern**: is a glob-style pattern which is matched against each list element using the same rules as the string match command.

- **-regexp Pattern**: is treated as a regular expression and matched against each list element using the same rules as the regexp command.

If mode is omitted then it defaults to -glob.

This command is provided by Extended Tcl

load Load a shared library into the server

AUTH SYSTEM

SYNOPSIS

load libraryName [initProc]

DESCRIPTION

The load command is used to dynamically load a shared library into the server and call an initialization routine within the library. Typically, the library's initialization routine defines commands or creates variables used by the library. The libraryName argument is the path name to the shared library. If the initProc argument is given, the libraries symbol table is searched for a function of that name. If found, that function is invoked to perform the library initialization. If the initProc argument is not given, the name of the initialization function is derived from the libraryName argument using the following algorithm:

```
/usr/lib/libxxx.sl -> xxx_Init()
```

Unfortunately, the semantics of shared libraries are not consistent across all operating system platforms. Some features, such as the specification of the initialization procedure, are not available on all platforms. If an initialization procedure is passed to a load command on a platform that does not have that support, an error code is returned.

During a server restart, any dynamically loaded libraries are automatically unloaded before the internal interpreter is deleted.

RELATED INFORMATION

unload

loadlibindex Load a package library index

AUTH Not available

SYNOPSIS

loadlibindex libfile.tlib

DESCRIPTION

Load the package library index of the library file libfile (which must have the suffix .tlib). Package library indexes along the auto_path are loaded automatically on the first demand_load; this command is provided to explicitly load libraries that are not in the path. If the index file (with a .tndx suffix) does not exists or is out of date, it will be rebuilt if the user has directory permissions to create it. If a package with the same name as a package in libfile.tlib has already been loaded, its definition will be overridden by the new package. However, if any procedure has actually been used from the previously defined package, the procedures from libfile.tlib will not be loaded.

This command will also load an index built by mkindex.tcl program supplied with standard Tcl. This file must be named ″tclIndex″.

This command is provided by Extended Tcl.

loop **AUTH** ACL

SYNOPSIS

loop var first limit [incr] body

DESCRIPTION

Loop is a looping command, similar in behavior to the Tcl for statement, except that the loop statement achieves substantially higher performance and is easier to code when the beginning and ending values of a loop are known, and the loop variable is to be incremented by a known, fixed amount every time through the loop.

The var argument is the name of a Tcl variable that will contain the loop index. The loop index is set to the value specified by first. The Tcl interpreter is invoked upon body zero or more times, where var is incremented by increment every time through the loop, or by one if increment is not specified. Increment can be negative in which case the loop will count downwards.

When var reaches limit, the loop terminates without a subsequent execution of body. For instance, if the original loop parameters would cause loop to terminate, say first was one, limit was zero and increment was not specified or was non-negative, body is not executed at all and loop returns.

The first, limit and increment are integer expressions. They are only evaulated once at the beginning of the loop.

If a continue command is invoked within body then any remaining commands in the current execution of body are skipped, as in the for command. If a break command is invoked within body then the loop command will return immediately. Loop returns an empty string.

This command is provided by Extended Tcl.

lrange Return one or more adjacent elements from a list

AUTH AUTH

SYNOPSIS

lrange list first last

DESCRIPTION

List must be a valid Tcl list. This command will return a new list consisting of elements first through last, inclusive. First or last may be end (or any abbreviation of it) to refer to the last element of the list. If first is less than zero, it is treated as if it were zero. If last is greater than or equal to the number of elements in the list, then it is treated as if it were end. If first is greater than last then an empty string is returned.

Note: ″lrange list first first″ does not always produce the same result as ″lindex list first″ (although it often does for simple fields that are not enclosed in braces); it does, however, produce exactly the same results as ″list[lindex list first]″.

RELATED INFORMATION

element, list, range, sublist

lreplace Replace elements in a list with new elements

AUTH AUTH

SYNOPSIS

lreplace list first last [element element ...]

DESCRIPTION

Lreplace returns a new list formed by replacing one or more elements of list with the element arguments. First gives the index in list of the first element to be replaced (0 refers to the first element). If first is less than zero then it refers to the first element of list; the element indicated by first must exist in the list. Last gives the index in list of the last element to be replaced; it must be greater than or equal to first. First or last may be end (or any abbreviation of it) to refer to the last element of the list. The element arguments specify zero or more new arguments to be added to the list in place of those that were deleted. Each element argument will become a separate element of the list. If no element arguments are specified, then the elements between first and last are simply deleted.

RELATED INFORMATION

element, list, replace

lsearch See if a list contains a particular element

AUTH AUTH

SYNOPSIS

lsearch [mode] list pattern

DESCRIPTION

This command searches the elements of list to see if one of them matches pattern. If so, the command returns the index of the first matching element. If not, the command returns -1. The mode argument indicates how the elements of the list are to be matched against pattern and it must have one of the following values:

- **-exact**: The list element must contain exactly the same string as pattern.
- **-glob**: Pattern is a glob-style pattern which is matched against each list element using the same rules as the string match command.
- **-regexp**: Pattern is treated as a regular expression and matched against each list element using the same rules as the regexp command.

If mode is omitted then it defaults to -glob.

RELATED INFORMATION

list, match, pattern, regular expression, search, string

lsort Sort the elements of a list

AUTH AUTH

SYNOPSIS

lsort [-ascii] [-integer] [-real] [-increasing] [-decreasing] [-command string] list

DESCRIPTION

This command sorts the elements of list, returning a new list in sorted order. By default ASCII sorting is used with the result returned in increasing order. However, any of the following switches may be specified before list to control the sorting process (unique abbreviations are accepted):

- **-ascii**: Use string comparison with ASCII collation order. This is the default.
- **-integer**: Convert list elements to integers and use integer comparison.
- **-real**: Convert list elements to floatingpoint values and use floating comparison.
- **-command command**: Use command as a comparison command. To compare two elements, evaluate a Tcl script consisting of command with the two elements appended as additional arguments. The script should return an integer less than, equal to, or greater than zero if the first element is to be considered less than, equal to, or greater than the second, respectively.
- **-increasing**: Sort the list in increasing order (″smallest″ items first). This is the default.
- **-decreasing**: Sort the list in decreasing order (″largest″ items first).

RELATED INFORMATION

element, list, order, sort

lvarcat **AUTH** AUTH

SYNOPSIS

lvarcat var string [string...]

DESCRIPTION

This command treats each string argument as a list and concatenates them to the end of the contents of var, forming a a single list. The list is stored back into var and also returned as the result. If var does not exist, it is created.

This command is provided by Extended Tcl.

lvarpop **AUTH** AUTH

SYNOPSIS

lvarpop var [indexExpr] [string]

DESCRIPTION

The lvarpop command pops (deletes) the element indexed by the expression indexExpr from the list contained in the variable var. If index is omitted, then 0 is assumed. If string, is specified, then the deleted element is replaced by string. The replaced or deleted element is returned. Thus ″lvarpop argv 0″ returns the first element of argv, setting argv to contain the remainder of the string.

If the expression indexExpr starts with the string end, then end is replaced with the index of the last element in the list. If the expression starts with len, then len is replaced with the length of the list.

This command is provided by Extended Tcl.

lvarpush **AUTH** AUTH

SYNOPSIS

lvarpush var string [indexExpr]

DESCRIPTION

The lvarpush command pushes (inserts) string as an element in the list contained in the variable var. The element is inserted before position indexExpr in the list. If index is omitted, then 0 is assumed. If var does not exists, it is created.

If the expression indexExpr starts with the string end, then end is replaced with the index of the last element in the list. If the expression starts with len, then len is replaced with the length of the list. Note the a value of end means insert the string before the last element.

This command is provided by Extended Tcl.

max Calculates the maximum of specified values

AUTH ACL

SYNOPSIS

max num1 num2 [..numN]

min Calculates the minimum of specified values

AUTH ACL

SYNOPSIS

min num1 num2 [..numN]

mkdir Creates a directory

AUTH ACL

SYNOPSIS

mkdir [-path] dirList

DESCRIPTION

Create each of the directories in the list dirList. The mode on the new directories is 777, modified by the umask. If -path is specified, then any nonexistent parent directories in the specified path(s) are also created.

This command is provided by Extended Tcl.

nice Increases or decreases the ″nice″ buffer value

AUTH ACL

SYNOPSIS

nice [priorityincr]

DESCRIPTION

Change or return the process priority. If priorityincr is omitted, the current priority is returned. If priorityincr is positive, it is added to the current priority level, up to a system defined maximum (normally 19),

Negative priorityincr values cumulatively increase the program′s priority down to a system defined minimum (normally -19); increasing priority with negative niceness values will only work for the superuser.

The new priority is returned.

This command is provided by Extended Tcl.

open Open a file

AUTH ACL

SYNOPSIS

open filename [access] [permissions]

DESCRIPTION

This command opens a file and returns an identifier that may be used in future invocations of commands like read, puts and close. FileName gives the name of the file to open; if it starts with a tilde then tilde substitution is performed as described for Tcl_TildeSubst. If the first character of fileName is ″|″ then the remaining characters of fileName are treated as a command pipeline to invoke, in the same style as for exec. In this case, the identifier returned by open may be used to write to the command′s input pipe or read from its output pipe.

The access argument indicates the way in which the file (or command pipeline) is to be accessed. It may take two forms, either a string in the form that would be passed to the fopen library procedure or a list of POSIX access flags. It defaults to ″r″. In the first form access may have any of the following values:

- **r**: Open the file for reading only; the file must already exist.
- **r+**: Open the file for both reading and writing; the file must already exist.
- **w**: Open the file for writing only. Truncate it if it exists. If it does not exist, create a new file.
- **w+**: Open the file for reading and writing. Truncate it if it exists. If it does not exist, create a new file.
- **a**: Open the file for writing only. The file must already exist, and the file is positioned so that new data is appended to the file.
- **a+**: Open the file for reading and writing. If the file does not exist, create a new empty file. Set the initial access position to the end of the file.

In the second form, access consists of a list of any of the following flags, all of which have the standard POSIX meanings. One of the flags must be either RDONLY, WRONLY or RDWR.

- **RDONLY**: Open the file for reading only.
- **WRONLY**: Open the file for writing only.
- **RDWR**: Open the file for both reading and writing.
- **APPEND**: Set the file pointer to the end of the file prior to each write.
- **CREAT**: Create the file if it does not already exist (without this flag it is an error for the file not to exist).
- **EXCL**: If CREAT is also specified, an error is returned if the file already exists.
- **NOCTTY**: If the file is a terminal device, this flag prevents the file from becoming the controlling terminal of the process.
- **NONBLOCK**: Prevents the process from blocking while opening the file. For details refer to your system documentation on the open system call′s O_NONBLOCK flag.
- **TRUNC**: If the file exists it is truncated to zero length.

If a new file is created as part of opening it, permissions (an integer) is used to set the permissions for the new file in conjunction with the process′s file mode creation mask. Permissions defaults to 0666.

If a file is opened for both reading and writing then seek must be invoked between a read and a write, or vice versa (this restriction does not apply to command pipelines opened with open). When fileName specifies a command pipeline and a write-only access is used, then standard output from the pipeline is directed to the current standard output unless overridden by the command. When fileName specifies a command pipeline and a read-only access is used, then standard input from the pipeline is taken from the current standard input unless overridden by the command.

RELATED INFORMATION

access mode, append, controlling terminal, create, file, non-blocking, open, permissions, pipeline, process

pfck Parallel file systems check

AUTH AUTH

SYNOPSIS

```
pfck [-g] [-s n]  [-f n]  [-u n]  [-pf n]  [-pu n]
    [-is n] [-if n] [-iu n] [-pif n] [-piu n]
```

DESCRIPTION

The pfck command displays all file systems meeting a specified set of search criteria. The search criteria are specified in a series of one or more options. These o options are of the form of an attribute abbreviation followed by a number. Each abbreviation stands for a file system size attribute and the number stands for an amount to be compared against the corresponding actual value for each file system. The file system and its size attributes are displayed if any of the specified values are satisfied according to the following:

```
-s:   file system size is        > n (KB)
-f:   file system free space is   < n (KB).
-u:   file system used space is   > n (KB).
-pf:  file system free space % is < n %.
-pu:  file system used space % is > n %.
-is:  file system inodes is      > n.
-if:  file system free inode is   < n.
-iu:  file system used inodes is  > n.
-pif: file system free inode % is < n %.
-piu: file system used inode % is > n %.
```

If -any- of the specified criteria is satisfied then that file system is displayed (logical OR of all specified attributes). Attributes not specified on the command line are not included in the check.

Note: Checks for ″size″ and ″used″ are always ″>″; Checks for ″free″ are always ″<″.

The -g option causes the column heading to be suppressed (useful in combination with grep command).

Example:

```
sysctl> pfck -s 10000 -f 2000
tappan.kgn.ibm.com:--------------------------------------------------
Filesystem        Size-KB Used-KB Free-KB %Free  iUsed  iFree %iFree
/var                16384    4068   12316   76%    342   3754    92%
/usr               552960  464276   88684   17%  22777 116487    84%
/tmp                49152   32532   16620   34%    241  12047    99%
/home               65536   63128    2408    4%   5372  11012    68%
sysctl>
```

RELATED INFORMATION

pdf

pdf pdf - parallel display file systems

AUTH AUTH

SYNOPSIS

pdf [-g] [filsys [filsys] ...]

DESCRIPTION

The pdf command displays the named file systems and their usage statistics (e.g., size, inodes, free space, per cent free space, etc.). The Sysctl pdf command is similar to the standard ″df″ command. If no file systems are named then all file systems are displayed. The -g option causes the column heading to be suppressed (useful in combination with grep command).

Example:

```
sysctl> pdf /usr /tmp
tappan.kgn.ibm.com:--------------------------------------------------
Filesystem        Size-KB Used-KB Free-KB %Free  iUsed  iFree %iFree
/                    8192    5200    2992   37%    815   1233    61%
/var                16384    4064   12320   76%    342   3754    92%
/usr               552960  464276   88684   17%  22777 116487    84%
/tmp                49152   32532   16620   34%    241  12047    99%
/home               65536   63128    2408    4%   5372  11012    68%
```

```
sysctl>
```

RELATED INFORMATION

pfck

pfps Kill, renice or print statistics on processes

AUTH ACL

SYNOPSIS

pfps [-w hostlist | -] [-e hostlist]
    [-print | -kill signal | -nice val]
    [expression]

DESCRIPTION

The pfps commands allows processes to be killed, reniced, or printed out out with "ps guw" statistics. Processes can be selected by creating a logical expression from the following options and arguments:

```
-cpu    percent   true for processes %cpu > specified %cpu
-e      hostlist  comma separated list of hostnames to
                  exclude from hostlist
-kill   signal    kill process with specified signal if
                  expression evaluates to true
-mem    percent   true for processes %mem > specified
                  %mem
-n      name      full pathname of process
-nice   n         nice process with specified value n
-o      owner     owner of the process
-or               logically "or" options
-print            print all process that evaluate to true
                  in the "ps guw" format
-pty    name      process controlling terminal
-r      state     run state of process
-rtime  hh:mm     total execution time of process
-stime  dd:hh:mm  start time of process
-tn     name      name of proccess
-w      hostlist  list of comma separated hostnames on which
                  the pfps server code will run
```

AUTHORIZATION

The following authorization is required for the three possible options:

```
-print  A user must be authenticated.
-kill   An authenticated user can only kill a process which
        he owns. If a user is in the ACL file then -kill will
        work as if being run by root.
-nice   An authenticated user can only renice his own process
        to decrease its priority. If a user is in the ACL
        file then -nice will work as if being run by root.
```

RELATED INFORMATION

ps, kill, renice

pid Retrieve process id(s)

AUTH ACL

SYNOPSIS

pid [fileId]

DESCRIPTION

If the fileId argument is given then it should normally refer to a process pipeline created with the open command. In this case the pid command will return a list whose elements are the process identifiers of all the processes in the pipeline, in order. The list will be empty if fileId refers to an open file that is not a process pipeline. If no fileId argument is given then pid returns the process identifier of the current process. All process identifiers are returned as decimal strings.

RELATED INFORMATION

file, pipeline, process identifier

pipe **AUTH** ACL

SYNOPSIS

pipe [fileId_var_r fileId_var_w]

DESCRIPTION

Create a pipe. If fileId_var_r and fileId_var_r are specified, then pipe will set the a variable named fileId_var_r to contain the fileId of the side of the pipe that was opened for reading, and fileId_var_w will contain the fileId of the side of the pipe that was opened for writing.

If the fileId variables are not specified, then a list containing the read and write fileIdw is returned as the result of the command.

This command is provided by Extended Tcl.

pkill Kills a command

AUTH ACL

SYNOPSIS

pkill -signal <command or process-id>, kills specified command

proc Creates a Tcl procedure

AUTH SYSTEM

SYNOPSIS

proc name args body

DESCRIPTION

The proc command creates a new Tcl procedure named name, replacing any existing command or procedure there may have been by that name. Whenever the new command is invoked, the contents of body will be executed by the Tcl interpreter. Args specifies the formal arguments to the procedure. It consists of a list, possibly empty, each of whose elements specifies one argument. Each argument specifier is also a list with either one or two fields. If there is only a single field in the specifier then it is the name of the argument; if there are two fields, then the first is the argument name and the second is its default value.

When name is invoked a local variable will be created for each of the formal arguments to the procedure; its value will be the value of corresponding argument in the invoking command or the argument′s default value. Arguments with default values need not be specified in a procedure invocation. However, there must be enough actual arguments for all the formal arguments that do not have defaults, and there must not be any extra actual arguments. There is one special case to permit procedures with variable numbers of arguments. If the last formal argument has the name args, then a call to the procedure may contain more actual arguments than the procedure has formals. In this case, all of the actual arguments starting at the one that would be assigned to args are combined into a list (as if the list command had been used); this combined value is assigned to the local variable args.

When body is being executed, variable names normally refer to local variables, which are created automatically when referenced and deleted when the procedure returns. One local variable is automatically created for each of the procedure′s arguments. Global variables can only be accessed by invoking the global command or the upvar command.

The proc command returns an empty string. When a procedure is invoked, the procedure′s return value is the value specified in a return command. If the procedure does not execute an explicit return, then its return value is the value of the last command executed in the procedure′s body. If an error occurs while executing the procedure body, then the procedure-as-a-whole will return that same error.

RELATED INFORMATION

argument, procedure

procStat **AUTH** SYSTEM

SYNOPSIS

procStat ...?

profile **AUTH** ACL

SYNOPSIS

profile [-commands] on | off arrayVar

DESCRIPTION

This command is used to collect a performance profile of a Tcl script. It collects data at the Tcl procedure level. The number of calls to a procedure, and the amount of real and CPU time is collected. Time is also collected for the global context. The procedure data is collected by bucketing it based on the procedure call stack, this allows determination of how much time is spent in a particular procedure in each of it is calling contexts.

The on option enables profile data collection. If the -commands option is specified, data on all commands within a procedure is collected as well a procedures. Multiple occurrences of a command within a procedure are not distinguished, but this data may still be useful for analysis.

The off option turns off profiling and moves the data collected to the array arrayVar. The array is address by a list containing the procedure call stack. Element zero is the top of the stack, the procedure that the data is for. The data in each entry is a list consisting of the procedure call count and the real time and CPU time in milliseconds spent in the procedure (and all procedures it called). The list is in the form {count real cpu}. A Tcl procedure profrep is supplied for reducing the data and producing a report.

This command is provided by Extended Tcl.

pstat Lists AIX process status

AUTH AUTH

SYNOPSIS

pstat processname

Example:

```
pstat aixterm
```

Results:

```
CMD          PID   PPID USER       INVOCATION
aixterm    26038     1  root       aixterm
aixterm    28876     1  root       aixterm
```

puts Write to a file

AUTH ACL

SYNOPSIS

puts [-nonewline] [fileId] string

DESCRIPTION

Writes the characters given by string to the file given by fileId. FileId must have been the return value from a previous call to open, or it may be stdout or stderr to refer to one of the standard I/O channels; it must refer to a file that was opened for writing. If no fileId is specified then it defaults to stdout. Puts normally outputs a newline character after string, but this feature may be suppressed by specifying the -nonewline switch. Output to files is buffered internally by Tcl; the flush command may be used to force buffered characters to be output.

RELATED INFORMATION

file, newline, output, write

pwd Returns the pathname of the current working directory

AUTH ACL

SYNOPSIS

pwd

quit Similar to exit, End the current session

AUTH AUTH

SYNOPSIS

quit

DESCRIPTION

The quit command ends the current session with the sysctld server. It is typically used to end an interactive session with the server.

RELATED INFORMATION

exit

random **AUTH** AUTH

SYNOPSIS

random limit | seed [seedval]

Example:

```
random 10
```

Will generate a random rumber under 10.

read Read from a file

AUTH ACL

SYNOPSIS

read fileId [numBytes] or read [-nonewline] fileId

DESCRIPTION

In the first form, all of the remaining bytes are read from the file given by fileId; they are returned as the result of the command. If the -nonewline switch is specified then the last character of the file is discarded if it is a newline. In the second form, the extra argument specifies how many bytes to read; exactly this many bytes will be read and returned, unless there are fewer than numBytes bytes left in the file; in this case, all the remaining bytes are returned. FileId must be stdin or the return value from a previous call to open; it must refer to a file that was opened for reading. Any existing end-of-file or error condition on the file is cleared at the beginning of the read command.

RELATED INFORMATION

file, read

readdir **AUTH** ACL

SYNOPSIS

readdir dirPath

DESCRIPTION

Returns a list containing the contents of the directory dirPath. The directory entries ″.″ and ″..″ are not returned.

This command is provided by Extended Tcl.

regexp Match a regular expression against a string

AUTH AUTH

SYNOPSIS

regexp [switches] exp string [matchVar]
    [subMatchVar subMatchVar ...]

DESCRIPTION

Determines whether the regular expression exp matches part or all of string and returns 1 if it does, 0 if it does not.

If additional arguments are specified after string then they are treated as the names of variables in which to return information about which part(s) of string matched exp. MatchVar will be set to the range of string that matched all of exp. The first subMatchVar will contain the characters in string that matched the leftmost parenthesized subexpression within exp, the next subMatchVar will contain the characters that matched the next parenthesized subexpression to the right in exp, and so on.

If the initial arguments to regexp start with - then they are treated as switches. The following switches are currently supported:

- **-nocase**: Causes upper-case characters in string to be treated as lower case during the matching process.
- **-indices**: Changes what is stored in the subMatchVars. Instead of storing the matching characters from string, each variable will contain a list of two decimal strings giving the indices in string of the first and last characters in the matching range of characters.
- **--**: Marks the end of switches. The argument following this one will be treated as exp even if it starts with a -.

If there are more subMatchVar′s than parenthesized subexpressions within exp, or if a particular subexpression in exp does not match the string (e.g. because it was in a portion of the expression that wasn′t matched), then the corresponding subMatchVar will be set to ″-1 -1″ if -indices has been specified or to an empty string otherwise.

REGULAR EXPRESSIONS

Regular expressions are implemented using Henry Spencer′s package and much of the description of regular expressions below is copied verbatim from Henry Spencer′s manual entry.

A regular expression is zero or more branches, separated by ″|″. It matches anything that matches one of the branches.

A branch is zero or more pieces, concatenated. It matches a match for the first, followed by a match for the second, etc.

A piece is an atom possibly followed by ″*″, ″+″, or ″?″. An atom followed by ″*″ matches a sequence of 0 or more matches of the atom. An atom followed by ″+″ matches a sequence of 1 or more matches of the atom. An atom followed by ″?″ matches a match of the atom, or the null string.

An atom is a regular expression in parentheses (matching a match for the regular expression), a range (see below), ″.″ (matching any single character), ″¬″ (matching the null string at the beginning of the input string), ″$″ (matching the null string at the end of the input string), a ″″ followed by a single character (matching that character), or a single character with no other significance (matching that character).

A range is a sequence of characters enclosed in ″[]″. It normally matches any single character from the sequence. If the sequence begins with ¬, it matches any single character not from the rest of the sequence. If two characters in the sequence are separated by -, this is shorthand for the full list of ASCII characters between them (for example, ″[0-9]″ matches any decimal digit). To include a literal] in the sequence, make it the first character (following a possible ¬). To include a literal -, make it the first or last character.

CHOOSING AMONG ALTERNATIVE MATCHES

In general there may be more than one way to match a regular expression to an input string. For example, consider the command:

```
regexp  (a*)b*  aabaaabb  x  y
```

Considering only the rules given so far, x and y could end up with the values aabb and aa, aaab and aaa, ab and a, or any of several other combinations. To resolve this potential ambiguity regexp chooses among alternatives using the rule ″first then longest″. In other words, it considers the possible matches in order working from left to right across the input string and the pattern, and it attempts to match longer pieces of the input string before shorter ones. More specifically, the following rules apply in decreasing order of priority:

1. If a regular expression could match two different parts of an input string then it will match the one that begins earliest.
2. If a regular expression contains | operators then the leftmost matching sub-expression is chosen.
3. In *, + , and [constructs, longer matches are chosen in preference to shorter ones.
4. In sequences of expression components the components are considered from left to right.

In the example from above, (a*)b* matches aab: the (a*) portion of the pattern is matched first and it consumes the leading aa; then the b* portion of the pattern consumes the next b. Or, consider the following example:

```
regexp  (ab|a)(b*)c  abc  x  y  z
```

After this command x will be abc, y will be ab, and z will be an empty string. Rule 4 specifies that (ab|a) gets first shot at the input string and Rule 2 specifies that the ab sub-expression is checked before the a subexpression. Thus the b has already been claimed before the (b*) component is checked and (b*) must match an empty string.

RELATED INFORMATION

match, regular expression, string

regsub Perform substitutions based on regular expression pattern matching

AUTH AUTH

SYNOPSIS

regsub [switches] exp string subSpec varName

DESCRIPTION

This command matches the regular expression exp against string, and it copies string to the variable whose name is given by varName. If there is a match, then while copying string to varName the portion of string that matched exp is replaced with subSpec. If subSpec contains a ″&″ or ″0″, then it is replaced in the substitution with the portion of string that matched exp. If subSpec contains a ″n″, where n is a digit between 1 and 9, then it is replaced in the substitution with the portion of string that matched the n-th parenthesized subexpression of exp. Additional backslashes may be used in subSpec to prevent special interpretation of ″&″ or ″0″ or ″n″ or backslash. The use of backslashes in subSpec tends to interact badly with the Tcl parser′s use of backslashes, so it is generally safest to enclose subSpec in braces if it includes backslashes.

If the initial arguments to regexp start with - then they are treated as switches. The following switches are currently supported:

- **-all**: All ranges in string that match exp are found and substitution is performed for each of these ranges. Without this switch only the first matching range is found and substituted. If -all is specified, then ″&″ and ″n″ sequences are handled for each substitution using the information from the corresponding match.
- **-nocase**: Upper-case characters in string will be converted to lower-case before matching against exp; however, substitutions specified by subSpec use the original unconverted form of string.
- **--**: Marks the end of switches. The argument following this one will be treated as exp even if it starts with a -.

The command returns a count of the number of matching ranges that were found and replaced. See the manual entry for regexp for details on the interpretation of regular expressions.

RELATED INFORMATION

match, pattern, regular expression, substitute

rename Rename or delete a command

AUTH SYSTEM

SYNOPSIS

rename oldName newName

DESCRIPTION

Rename the command that used to be called oldName so that it is now called newName. If newName is an empty string then oldName is deleted. The rename command returns an empty string as result.

RELATED INFORMATION

command, delete, rename

replicate **AUTH** AUTH

SYNOPSIS

replicate string countExpr

DESCRIPTION

Returns string, replicated the number of times indicated by the expression countExpr.

This command is provided by Extended Tcl.

return Return from procedure

AUTH NONE

SYNOPSIS

return [-code code] [-errorinfo info] [-errorcode code] [string]

DESCRIPTION

Return immediately from the current procedure (or toplevel command or source command), with string as the return value. If string is not specified then an empty string will be returned as result.

EXCEPTIONAL RETURNS

In the usual case where the -code option is not specified the procedure will return normally (its completion code will be TCL_OK). However, the -code option may be used to generate an exceptional return from the procedure. Code may have any of the following values:

- **ok**: Normal return: same as if the option is omitted.
- **error**: Error return: same as if the error command were used to terminate the procedure, except for handling of errorInfo and errorCode variables (see below).
- **return**: The current procedure will return with a completion code of TCL_RETURN, so that the procedure that invoked it will return also.
- **break**: The current procedure will return with a completion code of TCL_BREAK, which will terminate the innermost nested loop in the code that invoked the current procedure. i
- **continue**: The current procedure will return with a completion code of TCL_CONTINUE, which will terminate the current iteration of the innermost nested loop in the code that invoked the current procedure.
- **value**: Value must be an integer; it will be returned as the completion code for the current procedure.

The -code option is rarely used. It is provided so that procedures that implement new control structures can reflect exceptional conditions back to their callers.

Two additional options, -errorinfo and -errorcode, may be used to provide additional information during error returns. These options are ignored unless code is error.

The -errorinfo option specifies an initial stack trace for the errorInfo variable; if it is not specified then the stack trace left in errorInfo will include the call to the procedure and higher levels on the stack but it will not include any information about the context of the error within the procedure. Typically the info value is supplied from the value left in errorInfo after a catch command trapped an error within the procedure.

If the -errorcode option is specified then code provides a value for the errorCode variable. If the option is not specified then errorCode will default to NONE.

RELATED INFORMATION

break, continue, error, procedure, return

rmdir **AUTH** ACL

SYNOPSIS

rmdir [-nocomplain] dirlist

DESCRIPTION

Remove each of the directories in the list dirList. If -nocomplain is specified, then errors will be ignored.

This command is provided by Extended Tcl.

safeargs Check arguments for special shell characters

AUTH ACL

SYNOPSIS

safeargs arg1 [arg2 ...]

DESCRIPTION

The safeargs command searches the arguments passed for occurrences of special shell characters (for example, $&*(){}[] ′″;|]<> n~ , etc) and returns an error result if it finds any. The character checking implemented by safeargs is the same as in the safeexec and safesystem commands. Typically, this is used in cases where arguments provided by the user are passed to a shell via mechanisms other than safeexec and safesystem, such as using the open command with a pipe:

```
create proc safe_pipe {args} {
     safeargs $args
     set f [open ″| /bin/ls $args″]
     return $f
}
```

RELATED INFORMATION

safesystem, safeexec

safeexec A safe version of the exec command

AUTH ACL

SYNOPSIS

safeexec [switches] arg [arg ...]

DESCRIPTION

The safeexec command is similar to the Tcl exec command except that the arguments are first checked for the occurrence of special shell characters. This command, along with the safesystem and safeargs commands, provides a mechanism for securely passing arguments from the remote user to a shell command. If safeexec is used in place of exec, a remote user is unable to pass special shell characters directly to a shell invocation.

RELATED INFORMATION

safeargs, safesystem, exec

safesystem A safe version of the system command

AUTH ACL

SYNOPSIS

safesystem command

DESCRIPTION

The safesystem command is similar to the Tcl system command except that the arguments are first checked for the occurrence of special shell characters. This command, along with the safeexec and safeargs commands, provides a mechanism for securely passing arguments from the remote user to a shell command. If safesystem is used in place of system, a remote user is unable to pass special shell characters directly to a shell invocation.

RELATED INFORMATION

safeargs, safeexec, system

scan Parse string using conversion specifiers in the style of sscanf

AUTH AUTH

SYNOPSIS

scan string format [varName varName ...]

DESCRIPTION

This command parses fields from an input string in the same fashion as the ANSI C sscanf procedure and returns a count of the number of conversions performed, or -1 if the end of the input string is reached before any conversions have been performed. String gives the input to be parsed and format indicates how to parse it, using % conversion specifiers as in sscanf. Each varName gives the name of a variable; when a field is scanned from string the result is converted back into a string and assigned to the corresponding variable.

DETAILS ON SCANNING

Scan operates by scanning string and formatString together. If the next character in formatString is a blank or tab then it matches any number of white space characters in string (including zero). Otherwise, if it is not a % character then it must match the next character of string. When a % is encountered in formatString, it indicates the start of a conversion specifier. A conversion specifier contains three fields after the %: a *, which indicates that the converted value is to be discarded instead of assigned to a variable; a number indicating a maximum field width; and a conversion character. All of these fields are optional except for the conversion character.

When scan finds a conversion specifier in formatString, it first skips any white-space characters in string. Then it converts the next input characters according to the conversion specifier and stores the result in the variable given by the next argument to scan. The following conversion characters are supported:

- **d**: The input field must be a decimal integer. It is read in and the value is stored in the variable as a decimal string.
- **o**: The input field must be an octal integer. It is read in and the value is stored in the variable as a decimal string.
- **x**: The input field must be a hexadecimal integer. It is read in and the value is stored in the variable as a decimal string.
- **c**: A single character is read in and its binary value is stored in the variable as a decimal string. Initial white space is not skipped in this case, so the input field may be a whitespace character. This conversion is different from the ANSI standard in that the input field always consists of a single character and no field width may be specified.

- **s**: The input field consists of all the characters up to the next white-space character; the characters are copied to the variable.
- **e** or **f** or **g**: The input field must be a floating-point number consisting of an optional sign, a string of decimal digits possibly containing a decimal point, and an optional exponent consisting of an e or E followed by an optional sign and a string of decimal digits. It is read in and stored in the variable as a floating-point string.
- [**chars**]: The input field consists of any number of characters in chars. The matching string is stored in the variable. If the first character between the brackets is a] then it is treated as part of chars rather than the closing bracket for the set.
- [¬ **chars**]: The input field consists of any number of characters not in chars. The matching string is stored in the variable. If the character immediately following the ¬ is a] then it is treated as part of the set rather than the closing bracket for the set.

The number of characters read from the input for a conversion is the largest number that makes sense for that particular conversion (e.g. as many decimal digits as possible for %d, as many octal digits as possible for %o, and so on). The input field for a given conversion terminates either when a white-space character is encountered or when the maximum field width has been reached, whichever comes first. If a * is present in the conversion specifier then no variable is assigned and the next scan argument is not consumed.

DIFFERENCES FROM ANSI SSCANF

The behavior of the scan command is the same as the behavior of the ANSI C sscanf procedure except for the following differences:

1. %p and %n conversion specifiers are not currently supported.
2. For %c conversions a single character value is converted to a decimal string, which is then assigned to the corresponding varName; no field width may be specified for this conversion.
3. The l, h, and L modifiers are ignored; integer values are always converted as if there were no modifier present and real values are always converted as if the l modifier were present (i.e. type double is used for the internal representation).

RELATED INFORMATION

conversion specifier, parse, scan

scancontext AUTH ACL

SYNOPSIS

scancontext option

DESCRIPTION

This command manages file scan contexts. A scan context is a collection of regular expressions and commands to execute when that regular expression matches a line of the file. A context may also have a single default match, to be applied against lines that do not match any of the regular expressions. Multiple scan contexts may be defined and they may be reused on multiple files. A scan context is identified by a context handle. The scancontext command takes the following forms:

- **scancontext create**: Create a new scan context. The scanmatch command is used to define patterns in the context. A contexthandle is returned, which the Tcl programmer uses to refer to the newly created scan context in calls to the Tcl file scanning commands.
- **scancontext delete contexthandle**: Delete the scan context identified by contexthandle, and free all of the match statements and compiled regular expressions associated with the specified context.
- **scancontext copyfile contexthandle** [**filehandle**]: Set or return the file handle that unmatched lines are copied to (See scanfile). If filehandle is omitted, the copy file handle is returned. If no copy file is associated with the context, {} is returned. If a file handle is specified, it becomes the copy file for this context. If filehandle is {}, then it removes any copy file specification for the context.

This command is provided by Extended Tcl.

scanfile **AUTH** ACL

SYNOPSIS

scanfile [-copyfile filehandle] contexthandle filehandle

DESCRIPTION

Scan the file specified by fileId, starting from the current file position. Check all patterns in the scan context specified by contexthandle against it, executing the match commands corresponding to patterns matched.

If the optional -copyfile argument is specified, the next argument is a file ID to which all lines not matched by any pattern (excluding the default pattern) are to be written. If the copy file is specified with this flag, instead of using the scancontext copyfile command, the file is disassociated from the scan context at the end of the scan.

This command is provided by Extended Tcl.

scanmatch **AUTH** ACL

SYNOPSIS

scanmatch [-nocase] contexthandle [regexp] command

DESCRIPTION

Specify Tcl commands, to be evaluated when regexp is matched by a scanfile command. The match is added to the scan context specified by contexthandle. Any number of match statements may be specified for a give context. Regexp is a regular expression (see the regexp command). If -nocase is specified as the first argument, the pattern is matched regardless of alphabetic case.

If regexp is not specified, then a default match is specified for the scan context. The default match will be executed when a line of the file does not match any of the regular expressions in the current scancontext.

The array matchInfo is available to the Tcl code that is executed when an expression matches (or defaults). It contains information about the file being scanned and where within it the expression was matched.

matchInfo is local to the top level of the match command unless declared global at that level by the Tcl global command. If it is to be used as a global, it must be declared global before scanfile is called (since scanfile sets the matchInfo before the match code is executed, a subsequent global will override the local variable). The following array entries are available:

- **matchInfo**(**line**): Contains the text of the line of the file that was matched.
- **matchInfo**(**offset**): The byte offset into the file of the first character of the line that was matched.
- **matchInfo**(**linenum**): The line number of the line that was matched. This is relative to the first line scanned, which is usually, but not necessarily, the first line of the file. The first line is line number one.
- **matchInfo**(**context**): The context handle of the context that this scan is associated with.
- **matchInfo**(**handle**): The file id (handle) of the file currently being scanned.
- **matchInfo**(**copyHandle**): The file id (handle) of the file specified by the -copyfile option. The element does not exist if -copyfile was not specified.
- **matchInfo**(**submatch0**): Will contain the characters matching the first parenthesized subexpression. The second will be contained in submatch1, etc.
- **matchInfo**(**subindex0**): Will contain the a list of the starting and ending indices of the string matching the first parenthesized subexpression. The second will be contained in subindex1, etc.

All scanmatch patterns that match a line will be processed in the order in which their specifications were added to the scan context. The remainder of the scanmatch patterncommand pairs may be skipped for a file line if a continue is executed by the Tcl code of a preceding, matched pattern.

If a return is executed in the body of the match command, the scanfile command currently in progress returns, with the value passed to return as its return value.

This command is provided by Extended Tcl.

seek Change the access position for an open file

AUTH ACL

SYNOPSIS

seek fileId offset [origin]

DESCRIPTION

Change the current access position for fileId. FileId must have been the return value from a previous call to open, or it may be stdin, stdout, or stderr to refer to one of the standard I/O channels. The offset and origin arguments specify the position at which the next read or write will occur for fileId. Offset must be an integer (which may be negative) and origin must be one of the following:

- **start** The new access position will be offset bytes from the start of the file.
- **current** The new access position will be offset bytes from the current access position; a negative offset moves the access position backwards in the file.
- **end** The new access position will be offset bytes from the end of the file. A negative offset places the access position before the end-of-file, and a positive offset places the access position after the end-of-file.

The origin argument defaults to start. This command returns an empty string.

RELATED INFORMATION

access position, file, seek

select **AUTH** ACL

SYNOPSIS

select readFileIds [writeFileIds] [exceptFileIds] [timeout]

DESCRIPTION

This command allows an Extended Tcl program to wait on zero or more files being ready for for reading, writing, have an exceptional condition pending, or for a timeout period to expire. readFileIds, writeFileIds, exceptFileIds are each lists of fileIds, as returned from open, to query. An empty list ({}) may be specified if a category is not used.

The files specified by the readFileIds list are checked to see if data is available for reading. The writeFileIds are checked if the specified files are clear for writing. The exceptFileIds are checked to see if an exceptional condition has occurred (typically, an error). The write and exception checking is most useful on devices, however, the read checking is very useful when communicating with multiple processes through pipes. Select considers data pending in the stdio input buffer for read files as being ready for reading, the files do not have to be unbuffered.

Timeout is a floating point timeout value, in seconds. If an empty list is supplied (or the parameter is omitted), then no timeout is set. If the value is zero, then the select command functions as a poll of the files, returning immediately even if none are ready.

If the timeout period expires with none of the files becoming ready, then the command returns an empty list. Otherwise the command returns a list of three elements, each of those elements is a list of the fileIds that are ready in the read, write and exception classes. If none are ready in a class, then that element will be the null list. For example:

```
select {file3 file4 file5} {file6 file7} {} 10.5
```

could return

```
{file3 file4} {file6} {}
```

or perhaps

```
file3 {} {}
```

This command is provided by Extended Tcl.

server_open **AUTH** ACL

SYNOPSIS

server_open [option] host service|port

set Read and write variables

AUTH AUTH

SYNOPSIS

set varName [newValue]

DESCRIPTION

Returns the value of variable varName. If value is specified, then set the value of varName to value, creating a new variable if one does not already exist, and return its value. If varName contains an open parenthesis and ends with a close parenthesis, then it refers to an array element: the characters before the first open parenthesis are the name of the array, and the characters between the parentheses are the index within the array. Otherwise varName refers to a scalar variable. If no procedure is active, then varName refers to a global variable. If a procedure is active, then varName refers to a parameter or local variable of the procedure unless the global command has been invoked to declare varName to be global.

RELATED INFORMATION

read, write, variable

setauth Set the authorization callback for an object

AUTH SYSTEM

SYNOPSIS

setauth -cmd | -var | -class objName callBack

DESCRIPTION

The setauth command attaches a new authorization callback to the given object. It is typically used in configuration files to override the default authorization specifier for a variable, procedure, or class.

The callBack argument is a Tcl expression that replaces the existing authorization callBack for the object objName. If a new authorization callback is attached to a variable that was not defined using the ″create var″ command, then the variable′s definition is modified so that it now behaves as if it was defined using ″create var″. That is, write access is no longer permitted and read access is granted via the authorization callback.

RELATED INFORMATION

getauth, checkauth, create

signal **AUTH** ACL

SYNOPSIS

signal action signalList [command]

DESCRIPTION

Specify the action to take when a Unix signal is received by Extended Tcl, or a program that embeds it. Siglist is a list of either the symbolic or numeric Unix signal (the SIG prefix is optional). Action is one of the following actions to be performed on receipt of the signal. To specify all modifiable signals, use ″*″. (this will not include SIGKILL and SIGSTOP, as they can not be modified).

- **default** Perform system default action when signal is received (see signal system call documentation).
- **ignore** Ignore the signal.
- **error** Generate a catchable Tcl error. It will be as if the command that was running returned an error. The error code will be in the form:

  ```
  POSIX SIG signame
  ```

  For the death of child signal, signame will always be SIGCHLD, rather than SIGCLD, to allow writing portable code.
- **trap** When the signal occurs, execute command and continue execution if an error is not returned by command. The command will be executed in the global context. The command will be edited before execution, replacing occurrences of ″%S″ with the signal name. Occurrences of ″%%″ result in a single ″%″. This editing occurs just before the trap command is evaluated. If an error is returned, then follow the standard Tcl error mechanism. Often command will just do an exit.
- **get** Retrieve the current settings of the specified signals. A keyed list will be returned were the keys are one of the specified signals and the values are a list consisting of the action associated with the signal, a 0 if the signal may be delivered (not block) and a 1 if it is blocked. The actions maybe one of ″default″, ″ignore″, ″error″ or ″trap″. If the action is trap, the third element is the command associated with the action. The action ″unknown″ is returned if a non-Tcl signal handler has been associated with the signal.
- **set** Set signals from a keyed list in the format returned by the get. For this action, siglist is the keyed list of signal state. Signals with an action of ″unknown″ are not modified.
- **block** Block the specified signals from being received. (Posix systems only).
- **unblock** Allow the specified signal to be received. Pending signals will not occur. (Posix systems only).

The signal action will remain enabled after the specified signal has occurred. The exception to this is SIGCHLD on systems without Posix signals. For these systems, SIGCHLD is not be automatically reenabled. After a SIGCHLD signal is received, a call to wait must be performed to retrieve the exit status of the child process before issuing another signal SIGCHLD ... command. For code that is to be portable between both types of systems, use this approach.

Signals are not processed until after the completion of the Tcl command that is executing when the signal is received. If an interactive Tcl shell is running, then the SIGINT will be set to error, noninteractive Tcl sessions leave SIGINT unchanged from when the process started (normally default for foreground processes and ignore for processes in the background).

This command is provided by Extended Tcl.

sleep **AUTH** ACL

SYNOPSIS

sleep seconds

DESCRIPTION

Sleep the Extended Tcl process for seconds seconds.

This command is provided by Extended Tcl.

source Evaluate a file as a Tcl script

AUTH SYSTEM

SYNOPSIS

source fileName

DESCRIPTION

Read file fileName and pass the contents to the Tcl interpreter as a script to evaluate in the normal fashion. The return value from source is the return value of the last command executed from the file. If an error occurs in evaluating the contents of the file then the source command will return that error. If a return command is invoked from within the file then the remainder of the file will be skipped and the source command will return normally with the result from the return command. If fileName starts with a tilde, then it is tilde-substituted as described in the Tcl_TildeSubst manual entry.

RELATED INFORMATION

file, script

split Split a string into a proper Tcl list

AUTH AUTH

SYNOPSIS

split string [splitChars]

DESCRIPTION

Returns a list created by splitting string at each character that is in the splitChars argument. Each element of the result list will consist of the characters from string that lie between instances of the characters in splitChars. Empty list elements will be generated if string contains adjacent characters in splitChars, or if the first or last character of string is in splitChars. If splitChars is an empty string then each character of string becomes a separate element of the result list. SplitChars defaults to the standard white-space characters. For example,

```
split ″comp.unix.misc″ .
```

returns ″comp unix misc″ and

```
split ″Hello world″ {}
```

returns ″H e l l o { } w o r l d″.

RELATED INFORMATION

list, split, string

statfs Obtain status about mounted filesystems

AUTH ACL

SYNOPSIS

statfs pathName arrayvar

DESCRIPTION

The statfs command returns status information about a mounted file system. The filesystem containing the pathName path is queried. The arrayvar array variable is filled in with the following keys:

- size - Size of the filesystem in KB
- used - Amount of space used in KB
- free - Amount of space free in KB
- isize - Total number of inodes
- iused - Number of inodes used
- ifree - Number of inodes free

For example, a sysctl proc could issue

```
statfs /usr fsbuf
```

and access the amount of free space in that filesystem with

```
$fsbuf(free)
```

The statfs command works for all mounted filesystems, not just local filesystems.

RELATED INFORMATION

listfs

string Manipulate strings

AUTH AUTH

SYNOPSIS

string option arg [arg ...]

DESCRIPTION

Performs one of several string operations, depending on option. The legal options (which may be abbreviated) are:

- **string compare string1 string2**: Perform a character-by-character comparison of strings string1 and string2 in the same way as the C strcmp procedure. Return -1, 0, or 1, depending on whether string1 is lexicographically less than, equal to, or greater than string2.
- **string first string1 string2**: Search string2 for a sequence of characters that exactly match the characters in string1. If found, return the index of the first character in the first such match within string2. If not found, return -1.

- **string index string charIndex**: Returns the charIndex′th character of the string argument. A charIndex of 0 corresponds to the first character of the string. If charIndex is less than 0 or greater than or equal to the length of the string then an empty string is returned.
- **string last string1 string2**: Search string2 for a sequence of characters that exactly match the characters in string1. If found, return the index of the first character in the last such match within string2. If there is no match, then return -1.
- **string length string**: Returns a decimal string giving the number of characters in string.
- **string match pattern string**: See if pattern matches string; return 1 if it does, 0 if it does not. Matching is done in a fashion similar to that used by the C-shell. For the two strings to match, their contents must be identical except that the following special sequences may appear in pattern:
  - *****: Matches any sequence of characters in string, including a null string.
  - **?**: Matches any single character in string.
  - [**chars**]: Matches any character in the set given by chars. If a sequence of the form x-y appears in chars, then any character between x and y, inclusive, will match.
  - **x**: Matches the single character x. This provides a way of avoiding the special interpretation of the characters *?[] in pattern.
- **string range string first last**: Returns a range of consecutive characters from string, starting with the character whose index is first and ending with the character whose index is last. An index of 0 refers to the first character of the string. Last may be end (or any abbreviation of it) to refer to the last character of the string. If first is less than zero then it is treated as if it were zero, and if last is greater than or equal to the length of the string then it is treated as if it were end. If first is greater than last then an empty string is returned.
- **string tolower string**: Returns a value equal to string except that all upper case letters have been converted to lower case.
- **string toupper string**: Returns a value equal to string except that all lower case letters have been converted to upper case.
- **string trim string** [**chars**]: Returns a value equal to string except that any leading or trailing characters from the set given by chars are removed. If chars is not specified then white space is removed (spaces, tabs, newlines, and carriage returns).
- **string trimleft string** [**chars**]: Returns a value equal to string except that any leading characters from the set given by chars are removed. If chars is not specified then white space is removed (spaces, tabs, newlines, and carriage returns).
- **string trimright string** [**chars**]: Returns a value equal to string except that any trailing characters from the set given by chars are

removed. If chars is not specified then white space is removed (spaces, tabs, newlines, and carriage returns).

- **string wordend string index**: Returns the index of the character just after the last one in the word containing character index of string. A word is considered to be any contiguous range of alphanumeric or underscore characters, or any single character other than these.
- **string wordstart string index**: Returns the index of the first character in the word containing character index of string. A word is considered to be any contiguous range of alphanumeric or underscore characters, or any single character other than these.

RELATED INFORMATION

case conversion, compare, index, match, pattern, string, word

svcconnect Connection authorization callback

AUTH AUTH

SYNOPSIS

svcconnect

DESCRIPTION

The svcconnect command determines the connection authorization policy for the sysctld server. When a user connects to the server, the svcconnect command is evaluated before evaluating the commands sent by the user (or before entering an interactive command loop). The result of the svcconnect command determines whether or not the server will allow the connection to proceed.

The result of the svcconnect callback is interpreted in the following manner:

- If the user is not authorized to execute the svcconnect command, the connection is terminated.
- If the svcconnect commands returns a Tcl error (via the error Tcl command), the connection is terminated.
- Otherwise, the session is allowed to continue.

The default svcconnect command allows any user to connect and is equivalent to the following procedure:

```
create proc svcconnect NONE {} {
     return "Authorization OK"
}
```

By modifying the authorization callback of svcconnect, or by redefining the procedure body, a site can implement whatever connection authorization policy it desires.

Some examples:

1. Only allow authenticated users to connect:

   ```
   setauth -cmd svcconnect AUTH
   ```

2. Only allow users listed in a certain ACL to connect:

   ```
   setauth -cmd svcconnect {ACL /acls/connect.acl}
   ```

3. Only allow hosts in the current DNS domain to connect:

```
create proc svcconnect NONE {} {
     global SCHOST SCLHOST

     set ldomain $SCLHOST
     set rdomain $SCHOST

     ctoken ldomain ″.″
     ctoken rdomain ″.″

     if [cequal $ldomain $rdomain] {
        return ″Authorization OK″
     } else {
        error ″Authorization Denied″
     }
}
```

RELATED INFORMATION

svclogevent, setauth, create

svcdetach Detach the current process from the remote user

AUTH ACL

SYNOPSIS

svcdetach

DESCRIPTION

The svcdetach command is used to break the connection with the remote user. This is typically used when starting long-running daemons from within sysctld. The svcdetach command will close the connection with the remote user and point stdin/stdout/stderr at /dev/null. The svcredirect command can be used after svcdetach to redirect stdout and stderr.

Example: Use svcdetach to start a background server and let all output go to /dev/null:

```
create proc start_server ACL {} {
    svcdetach
    svclog ″Starting a background daemon ...″
    execl <server-path-name>
}
```

RELATED INFORMATION

svcredirect

svclog Write a line to the server′s log file

AUTH ACL

SYNOPSIS

svclog message

DESCRIPTION

The svclog command writes a time-stamped message to the server′s log file containing the string passed as the ″message″ parameter. If the server is running in audit mode, the connection ID of the current session is included along with the timestamp.

RELATED INFORMATION

svclogevent

svclogevent Connection-logging callback

AUTH SYSTEM

SYNOPSIS

svclogevent

DESCRIPTION

The svclogevent command is executed whenever a user connects to the server. Its purpose is to write a line to the log file indicating the connection.

The default svclogevent command is equivalent to the following procedure:

```
create proc svclogevent SYSTEM {} {
     global SCPRINCIPAL SCHOST
     if {$SCPRINCIPAL == ""} {
               set principal unknown
     } else {
               set principal $SCPRINCIPAL
     }
     svclog "$principal on $SCHOST"
}
```

By redefining the svclogevent procedure via the server's configuration files, a site can modify the format of the connection information written to the log file.

RELATED INFORMATION

svcconnect, svclog

svcpid Get the process ID of the parent sysctld server

AUTH ACL

SYNOPSIS

svcpid

DESCRIPTION This command returns the process ID of the parent sysctld process. If the process ID of the current process is desired, the "pid" command can be used.

RELATED INFORMATION

pid, id

svcredirect Redirect stdout/stderr to new file handles

AUTH ACL

SYNOPSIS

svcredirect fileId [fileId2]

DESCRIPTION

The svcredirect command is used to redirect stdout and stderr of the current process to alternate file handles. It is typically used in conjunction with the "svcdetach" command to start a back-end process and capture the output for later use.

If only one fileId is given, both stdout and stderr are redirected to that fileId. If two are given, stdout is redirected to the first and stderr to the second. All file handles passed must have been opened for writing.

Example: Use svcdetach and svcredirect to start a background server and capture all stdout/stderr in a file:

```
create proc start_server ACL {} {
     set f [open /var/adm/server.log w]
     svcdetach
     svclog "Starting a background daemon ..."
     svcredirect $f
     execl <server-path-name>
}
```

RELATED INFORMATION

svcdetach

svcrestart Restart the server

AUTH ACL

SYNOPSIS

svcrestart

DESCRIPTION

The svcrestart command initiates a restart sequence within the server. At restart, the server unloads all dynamically-loaded shared libraries, deletes its internal interpreter, re-initializes the log file, re-reads all configuration files and recreates its internal interpreter. A restart is normally issued when a configuration file (or dynamic library) has been modified and the changes need to be realized by the server.

Note that any session active when the restart is initiated (including the session that invokes the svcrestart command) is not affected by the restart. Only sessions which begin after the restart has been issued will see the changes made by the restart.

RELATED INFORMATION unload

svcversion Get the current version level of the server

AUTH NONE

SYNOPSIS

svcversion

DESCRIPTION

The svcversion command returns the current version level of the server.

RELATED INFORMATION

switch Evaluate one of several scripts, depending on a given value

AUTH AUTH

SYNOPSIS

switch [switches] string pattern body ... [default body]

DESCRIPTION

The switch command matches its string argument against each of the pattern arguments in order. As soon as it finds a pattern that matches string it evaluates the following body argument by passing it recursively to the Tcl interpreter and returns the result of that evaluation. If the last pattern argument is default then it matches anything. If no pattern argument matches string and no default is given, then the switch command returns an empty string.

If the initial arguments to switch start with - then they are treated as options. The following options are currently supported:

- **-exact**: Use exact matching when comparing string to a pattern. This is the default.
- **-glob**: When matching string to the patterns, use globstyle matching (i.e. the same as implemented by the string match command).
- **-regexp**: When matching string to the patterns, use regular expression matching (i.e. the same as implemented by the regexp command).
- **--**: Marks the end of options. The argument following this one will be treated as string even if it starts with a -.

Two syntaxes are provided for the pattern and body arguments. The first uses a separate argument for each of the patterns and commands; this form is convenient if substitutions are desired on some of the patterns or commands. The second form places all of the patterns and commands together into a single argument; the argument must have proper list structure, with the elements of the list being the patterns and commands. The second form makes it easy to construct multi-line switch commands, since the braces around the whole list make it unnecessary to include a backslash at the end of each line. Since the pattern arguments are in braces in the second form, no command or variable substitutions are performed on them; this makes the behavior of the second form different than the first form in some cases. If a body is specified as ″-″ it means that the body for the next pattern should also be used as the body for this pattern (if the next pattern also has a body of ″-″ then the body after that is used, and so on). This feature makes it possible to share a single body among several patterns.

Below are some examples of switch commands:

```
switch abc a - b {format 1} abc {format 2} default {format 3}
```

will return 2,

```
switch -regexp aaab {
    ¬a.*b$ -
    b {format 1}
    a* {format 2}
    default {format 3}
}
```

will return 1, and

```
switch xyz {
    a
      -
    b
      {format 1}
    a*
      {format 2}
    default
      {format 3}
}
```

will return 3.

RELATED INFORMATION

switch, match, regular expression

sync **AUTH** ACL

SYNOPSIS

synchronize [fileId]

DESCRIPTION

If fileId is not specified, or if it is and this system does not support the fsync system call, issues a sync system call to flush all pending disk output. If fileId is specified and the system does support the fsync system call, issues an fsync on the file corresponding to the specified Tcl fileId to force all pending output to that file out to the disk.

If fileId is specified, the file must be writable. A flush will be issued against the fileId before the sync.

The infox have_fsync command can be used to determine if ″sync fileId″ will do a sync or a fsync.

This command is provided by Extended Tcl.

system **AUTH** ACL

SYNOPSIS

system ″command″

DESCRIPTION

Executes command via the system(3) call. Differs from exec in that system does not return the executed command′s standard output as the result string, and system goes through the Unix shell to provide wildcard expansion, redirection, etc, as is normal from an sh command line. The exit code of the command is returned.

This command is provided by Extended Tcl.

tclvars Variables used by Tcl

AUTH No Authorization Callback

DESCRIPTION

The following global variables are created and managed automatically by the Tcl library. Except where noted below, these variables should normally be treated as read-only by application-specific code and by users.

- **env**: This variable is maintained by Tcl as an array whose elements are the environment variables for the process. Reading an element will return the value of the corresponding environment variable. Setting an element of the array will modify the corresponding environment variable or create a new one if it does not already exist. Unsetting an element of env will remove the corresponding environment variable. Changes to the env array will affect the environment passed to children by commands like exec. If the entire env array is unset then Tcl will stop monitoring env accesses and will not update environment variables.

- **errorCode**: After an error has occurred, this variable will be set to hold additional information about the error in a form that is easy to process with programs. errorCode consists of a Tcl list with one or more elements. The first element of the list identifies a general class of errors, and determines the format of the rest of the list. The following formats for errorCode are used by the Tcl core; individual applications may define additional formats.
  - **ARITH code msg**: This format is used when an arithmetic error occurs (e.g. an attempt to divide by zero in the expr command). Code identifies the precise error and msg provides a human-readable description of the error. Code will be either DIVZERO (for an attempt to divide by zero), DOMAIN (if an argument is outside the domain of a function, such as acos(-3)), IOVERFLOW (for integer overflow), OVERFLOW (for a floating-point overflow), or UNKNOWN (if the cause of the error cannot be determined).
  - **CHILDKILLED pid sigName msg**: This format is used when a child process has been killed because of a signal. The second element of errorCode will be the process′s identifier (in decimal). The third element will be the symbolic name of the signal that caused the process to terminate; it will be one of the names from the include file signal.h, such as SIGPIPE. The fourth element will be a short human-readable message describing the signal, such as ″write on pipe with no readers″ for SIGPIPE.
  - **CHILDSTATUS pid code**: This format is used when a child process has exited with a non-zero exit status. The second element of errorCode will be the process′s identifier (in decimal) and the third element will be the exit code returned.

tell Return current access position for an open file

AUTH ACL

SYNOPSIS

tell fileId

DESCRIPTION

Returns a decimal string giving the current access position in fileId. FileId must have been the return value from a previous call to open, or it may be stdin, stdout, or stderr to refer to one of the standard I/O channels.

RELATED INFORMATION

access position, file

time Time the execution of a script

AUTH ACL

SYNOPSIS

time command [count]

DESCRIPTION

This command will call the Tcl interpreter count times to evaluate script (or once if count is not specified). It will then return a string of the form:

```
503 microseconds per iteration
```

This indicates the average amount of time required per iteration, in microseconds. Time is measured in elapsed time, not CPU time.

RELATED INFORMATION

script, time

times **AUTH** ACL

SYNOPSIS

times

DESCRIPTION

Return a list containing the process and child execution times in the form:

```
utime stime cutime cstime
```

Also see the times(2) system call manual page. The values are in milliseconds.

This command is provided by Extended Tcl.

trace Monitor variable accesses

AUTH ACL

SYNOPSIS

trace option [arg arg ...]

DESCRIPTION

This command causes Tcl commands to be executed whenever certain operations are invoked. At present, only variable tracing is implemented. The legal option′s (which may be abbreviated) are:

- **trace variable name ops command**: Arrange for command to be executed whenever variable name is accessed in one of the ways given by ops. Name may refer to a normal variable, an element of an array, or to an array as a whole (that means, name may be just the name of an array, with no parenthesized index). If name refers to a whole array, then command is invoked whenever any element of the array is manipulated.

  Ops indicates which operations are of interest, and consists of one or more of the following letters:

  – **r**: Invoke command whenever the variable is read.

  – **w**: Invoke command whenever the variable is written.

– **u**: Invoke command whenever the variable is unset. Variables can be unset explicitly with the unset command, or implicitly when procedures return (all of their local variables are unset). Variables are also unset when interpreters are deleted, but traces will not be invoked because there is no interpreter in which to execute them.

When the trace triggers, three arguments are appended to command so that the actual command is as follows:

```
command name1 name2 op
```

Name1 and name2 give the name(s) for the variable being accessed: if the variable is a scalar then name1 gives the variable′s name and name2 is an empty string; if the variable is an array element then name1 gives the name of the array and name2 gives the index into the array; if an entire array is being deleted and the trace was registered on the overall array, rather than a single element, then name1 gives the array name and name2 is an empty string. Op indicates what operation is being performed on the variable, and is one of r, w, or u as defined above.

Command executes in the same context as the code that invoked the traced operation: if the variable was accessed as part of a Tcl procedure, then command will have access to the same local variables as code in the procedure. This context may be different than the context in which the trace was created. If command invokes a procedure (which it normally does) then the procedure will have to use upvar or uplevel if it wishes to access the traced variable. Note also that name1 may not necessarily be the same as the name used to set the trace on the variable; differences can occur if the access is made through a variable defined with the upvar command.

For read and write traces, command can modify the variable to affect the result of the traced operation. If command modifies the value of a variable during a read or write trace, then the new value will be returned as the result of the traced operation. The return value from command is ignored except that if it returns an error of any sort then the traced operation also returns an error with the same error message returned by the trace command (this mechanism can be used to implement read-only variables, for example). For write traces, command is invoked after the variable′s value has been changed; it can write a new value into the variable to override the original value specified in the write operation. To implement read-only variables, command will have to restore the old value of the variable.

While command is executing during a read or write trace, traces on the variable are temporarily disabled. This means that reads and writes invoked by command will occur directly, without invoking command (or any other traces) again. However, if command unsets the variable then unset traces will be invoked.

When an unset trace is invoked, the variable has already been deleted: it will appear to be undefined with no traces. If an unset occurs because of a procedure return, then the trace will be invoked in the variable context of the procedure being returned to:

the stack frame of the returning procedure will no longer exist. Traces are not disabled during unset traces, so if an unset trace command creates a new trace and accesses the variable, the trace will be invoked. Any errors in unset traces are ignored.

If there are multiple traces on a variable they are invoked in order of creation, most-recent first. If one trace returns an error, then no further traces are invoked for the variable. If an array element has a trace set, and there is also a trace set on the array as a whole, the trace on the overall array is invoked before the one on the element.

Once created, the trace remains in effect either until the trace is removed with the trace vdelete command described below, until the variable is unset, or until the interpreter is deleted. Unsetting an element of array will remove any traces on that element, but will not remove traces on the overall array.

This command returns an empty string.

- **trace vdelete name ops command** If there is a trace set on variable name with the operations and command given by ops and command, then the trace is removed, so that command will never again be invoked. Returns an empty string.
- **trace vinfo name** Returns a list containing one element for each trace currently set on variable name. Each element of the list is itself a list containing two elements, which are the ops and command associated with the trace. If name does not exist or does not have any traces set, then the result of the command will be an empty string.

RELATED INFORMATION

read, variable, write, trace, unset

translit **AUTH** AUTH

SYNOPSIS

translit inrange outrange string

DESCRIPTION

Translate characters in string, changing characters occuring in inrange to the corresponding character in outrange. Inrange and outrange may be list of characters or a range in the form ″A-M″. For example:

```
translit a-z A-Z foobar
```

This command is provided by Extended Tcl.

umask **AUTH** ACL

SYNOPSIS

umask [octalmask]

DESCRIPTION

Sets file-creation mode mask to the octal value of octalmask. If octalmask is omitted, the current mask is returned.

This command is provided by Extended Tcl.

unknown Handle attempts to use non-existent commands

AUTH Not available with default Tcl installation on SP

SYNOPSIS

unknown cmdName ?arg arg ...?

DESCRIPTION

This command does not actually exist as part of Tcl, but Tcl will invoke it if it does exist. If the Tcl interpreter encounters a command name for which there is not a defined command, then Tcl checks for the existence of a command named unknown. If there is no such command, then the interpreter returns an error. If the unknown command exists, then it is invoked with arguments consisting of the fully-substituted name and arguments for the original non-existent command. The unknown command typically does things like searching through library directories for a command procedure with the name cmdName, or expanding abbreviated command names to full-length, or automatically executing unknown commands as sub-processes. In some cases (such as expanding abbreviations) unknown will change the original command slightly and then (re-)execute it. The result of the unknown command is used as the result for the original non-existent command.

RELATED INFORMATION

error, non-existent command

unset Delete variables

AUTH AUTH

SYNOPSIS

unset varName [varName ...]

DESCRIPTION

This command removes one or more variables. Each name is a variable name, specified in any of the ways acceptable to the set command. If a name refers to an element of an array then that element is removed without affecting the rest of the array. If a name consists of an array name with no parenthesized index, then the entire array is deleted. The unset command returns an empty string as result. An error occurs if any of the variables does not exist, and any variables after the non-existent one are not deleted.

RELATED INFORMATION

remove, variable

unlink **AUTH** ACL

SYNOPSIS

unlink [-nocomplain] filelist

DESCRIPTION

Delete (unlink) the files whose names are in the list filelist. If -nocomplain is specified, then errors will be ignored.

This command is provided by Extended Tcl.

unload Unload a previously-loaded shared library

AUTH SYSTEM

SYNOPSIS

unload all | libraryName

DESCRIPTION

The unload command is used to unload a previously loaded shared library that was loaded via the load command. The libraryName argument must exactly match the path name of the library supplied in the load command. When a library is unloaded, the server deletes all Tcl commands and variables created when the library was initially loaded, thus avoiding dangling references.

If the argument passed is ″all″, the server unloads all libraries that have been loaded to this point.

The server automatically unloads all libraries when restarting or terminating operation.

RELATED INFORMATION

load, svcrestart

uplevel Execute a script in a different stack frame

AUTH SYSTEM

SYNOPSIS

uplevel [level] command [arg ...]

DESCRIPTION

All of the arg arguments are concatenated as if they had been passed to concat; the result is then evaluated in the variable context indicated by level. Uplevel returns the result of that evaluation.

If level is an integer then it gives a distance (up the procedure calling stack) to move before executing the command. If level consists of # followed by a number then the number gives an absolute level number. If level is omitted then it defaults to 1. Level cannot be defaulted if the first command argument starts with a digit or #.

For example, suppose that procedure a was invoked from top-level, and that it called b, and that b called c. Suppose that c invokes the uplevel command. If level is 1 or #2 or omitted, then the command will be executed in the variable context of b. If level is 2 or #1 then the command will be executed in the variable context of a. If level is 3 or #0 then the command will be executed at toplevel (only global variables will be visible).

The uplevel command causes the invoking procedure to disappear from the procedure calling stack while the command is being executed. In the above example, suppose c invokes the command:

```
uplevel 1 {set x 43; d}
```

Where d is another Tcl procedure. The set command will modify the variable x in b′s context, and d will execute at level 3, as if called from b. If it in turn executes the command:

```
uplevel {set x 42}
```

Then the set command will modify the same variable x in b′s context: the procedure c does not appear to be on the call stack when d is executing. The command ″info level″ may be used to obtain the level of the current procedure.

Uplevel makes it possible to implement new control constructs as Tcl procedures (for example, uplevel could be used to implement the while construct as a Tcl procedure).

RELATED INFORMATION

context, stack frame, variables

upvar Create link to variable in a different stack frame

AUTH ACL

SYNOPSIS

upvar [level] otherVar localVar [otherVar localVar ...]

DESCRIPTION

This command arranges for one or more local variables in the current procedure to refer to variables in an enclosing procedure call or to global variables. Level may have any of the forms permitted for the uplevel command, and may be omitted if the first letter of the first otherVar is not # or a digit (it defaults to 1). For each otherVar argument, upvar makes the variable by that name in the procedure frame given by level (or at global level, if level is #0) accessible in the current procedure by the name given in the corresponding myVar argument. The variable named by otherVar need not exist at the time of the call; it will be created the first time myVar is referenced, just like an ordinary variable. MyVar may not refer to an element of an array, but otherVar may refer to an array element. Upvar returns an empty string.

The upvar command simplifies the implementation of call-by-name procedure calling and also makes it easier to build new control constructs as Tcl procedures. For example, consider the following procedure:

```
proc add2 name {
    upvar $name x
    set x [expr $x+2]
}
```

Add2 is invoked with an argument giving the name of a variable, and it adds two to the value of that variable. Although add2 could have been implemented using uplevel instead of upvar, upvar makes it simpler for add2 to access the variable in the caller′s procedure frame.

If an upvar variable is unset (e.g. x in add2 above), the unset operation affects the variable it is linked to, not the upvar variable. There is no way to unset an upvar variable except by exiting the procedure in which it is defined. However, it is possible to retarget an upvar variable by executing another upvar command.

RELATED INFORMATION

context, frame, global, level, procedure, variable

wait **AUTH** ACL

SYNOPSIS

wait [-nohang] [-untraced] [-pgroup] [pid]

DESCRIPTION

Waits for a process created with the execl command to terminate, either due to an untrapped signal or call to exit system call. If the process id pid is specified, they wait on that process, otherwise wait on any child process to terminate.

If -nohang is specified, then do not block waiting on a process to terminate. If no process is immediately available, return an empty list. If -untraced is specified then the status of child processes that are stopped, and whose status has not yet been reported since they stopped, are also returned. If -pgroup is specfied and pid is not specified, then wait on any child process whose process groupd ID is they same as the calling process. If pid is specified with -pgroup, then it is take as a process group ID, waiting on any process in that process group to terminate.

Wait returns a list containing three elements: The first element is the process id of the process that terminated. If the process exited normally, the second element is ″EXIT″, and the third contains the numeric exit code. If the process terminated due to a signal, the second element is ″SIG″, and the third contains the signal name. If the process is currently stopped (on systems that support SIGSTP), the second element is ″STOP″, followed by the signal name.

Note that it is possible to wait on processes to terminate that were create in the background with the exec command. However, if any other exec command is executed after the process terminates, then the process status will be reaped by the exec command and will not be available to the wait command.

This command is provided by Extended Tcl.

while Execute script repeatedly as long as a condition is met

AUTH ACL

SYNOPSIS

while test body

DESCRIPTION

The while command evaluates test as an expression (in the same way that expr evaluates its argument). The value of the expression must a proper Boolean value; if it is a true value then body is executed by passing it to the Tcl interpreter. Once body has been executed then test is evaluated again, and the process repeats until eventually test evaluates to a false Boolean value. Continue commands may be executed inside body to terminate the current iteration of the loop, and break commands may be executed inside body to cause immediate termination of the while command. The while command always returns an empty string.

RELATED INFORMATION

Boolean value, loop, test, while

whoami Return the authenticated identity of the user

AUTH NONE

SYNOPSIS

whoami

DESCRIPTION

Returns the authenticated identity of the remote user in the user.instance@REALM format. If the user is unauthenticated, the string ″unknown″ is returned.

RELATED INFORMATION

Appendix G. S/370 Channel Emulator (RPQ 8K1922) Hardware Characteristics

Following are the specifics for ordering a S/370 Channel Emulator:

```
 REVDATE =950224
 REFNO   =8K1922
 RPQTITL =S/370 CHAN EMUL/A ON
RPQDATA        8K1922    S/370 CHAN EMUL/A ON SP2        NON-STANDARD
               PLANT OR FIELD INSTALLATION               LAB OF CONTROL: KGN
REVDATE        02/24/95
MACH_MOD         MACHINE:  9076   FI HRS: 0000.0      SYS INSTALL HRS: 0000.0
               MODELS:     201  202  203  204  3A2  3A3  3A4  3B2
                           3B3  3B4  301  302  303  304  401  402
                           403  404
               MODIFICATION/FEATURE RPQ
               LEAD TIME:   IS
               NON STANDARD RPQ (I) - SUBMIT REQUEST FOR APPROVAL

PREREQ    PRE-REQUISITES:     NONE

          LIMITATIONS:    LIMIT OF 4 PER SP2 THIN NODE.
                          LIMIT OF 4 PER SP2 WIDE NODE.
                          LIMIT PER SP2 TALL FRAME IS 12.
                          LIMIT PER SP2 LOW FRAME (MODELS 3AX) IS 7.
                             Note: The volume of cables from large numbers of
                                   these adapters may limit the ability to
                                   install other I/O adapters.
                          MAY BE INSTALLED IN THE LOWER 4 DRAWERS ONLY.

          SPECIFY:    NONE

          ----------------------------------------------------------------------
          REVISION NOTES: 12/06/94 - Changed Frame Limit to 12.
          ----------------------------------------------------------------------
DESCRIPTI DESCRIPTION:

          SUPPORT FOR FC#2759 IN A SP2 THIN OR WIDE NODE.

          THIS RPQ PROVIDES FOR THE SUPPORT OF THIS FEATURE. THIS FEATURE
             STILL NEEDS TO BE PURCHASED AT CURRENT SALES MANUAL PRICES.
             THE PRICE QUOTED IS FOR EACH ADAPTER.
          THIS ADAPTER REQUIRES AN ADDITIONAL, SEPARATELY PRICED, RS/6000
             PRPQ FOR THE TAPE SUPPORT (PRPQ NUMBER P83033). TWO PTF'S ARE ALSO
             REQUIRED TO AIX 3.2. THEY ARE PTF NUMBERS U405589 AND U403146.

          IT IS IMPORTANT THAT THE CUSTOMER UNDERSTANDS THAT THIS ADAPTER DOES
             NOT HAVE PARITY CHECKING ON CRITICAL PARTS OF THE DATA PATH, AND
             CAN, UPON HARDWARE FAILURE, EXPOSE THE CUSTOMER DATA TO CORRUPTION.

PRODNO    9076
          9076-201
          9076-202
          9076-203
```

```
DATA BASE : SPMD  -  IBM INTERNAL USE ONLY                               PAGE:    2

          9076-204
          9076-3A2
          9076-3A3
          9076-3A4
          9076-3B2
          9076-3B3
          9076-3B4
          9076-301
          9076-302
          9076-303
          9076-304
          9076-401
          9076-402
          9076-403
          9076-404
```

List of Abbreviations

ADSM	ADSTAR Distributed Storage Manager
AFS	Andrew File System
ARP	Address Resolution Protocol
ASCII	American National Standard for Information Interchange
BNF	Backus Naur/normal form (a metalanguage)
BOS	Base Operation System
DBCS	Double-Byte Character Set
DFS	Distributed File System
EBCDIC	Extended Binary Coded Decimal Interchange Code
FDDI	Fiber Distributed Data Interface
GUI	Graphical User Interface
HACMP	High-Availability Cluster Multi-Processing
HCON	IBM 3270 Host Connection Program
HPS	High Performance Switch
HSD	Hashed Shared Disk or Virtual Shared Disk Data-Striping Device
ITSO	International Technical Support Organization
LAN	Local Area Network
LPP	Licensed Program Product
LVM	Logical Volume Manager
NFS	Network File System
NIS	Network Information System
NTP	Network Time Protocol
PSF	Print Services Facility
PSSP	Parallel System Support Program
PERL	Practical Extraction and Report Language
PTX	Performance Toolbox
RAID	Redundant Array of Independent Disks
RDBMS	Relational Database Management System
RPC	Remote Procedure Call
RPQ	Request for Price Quotation (IBM custom-built machines or features)
RVSD	Recoverable Virtual Shared Disk
SCSI	Small Computer System Interface
SDR	System Data Repository
SMIT	System Management Interface Tool
SNMP	Simple Network Management Protocol (a TCP/IP protocol)
SP2	IBM Scalable POWERparallel System 2
SRC	System Resource Controller
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP/IP	User Datagram Protocol/Internet Protocol
VSD	Virtual Shared Disk

Index

ITSO Technical Bulletin Evaluation RED000

International Technical Support Organization
RS/6000 SP System Management: Easy, Lean and Mean
June 1995

Publication No. GG24-2563-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____

Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

____

____

What other topics would you like to see in this Bulletin?

____

____

What other Technical Bulletins would you like to see published?

____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

______________________________ ______________________________
Name Address

______________________________ ______________________________
Company or Organization

______________________________ ______________________________
Phone No.

ITSO Technical Bulletin Evaluation **RED000**
GG24-2563-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Mail Station P099
522 SOUTH ROAD
POUGHKEEPSIE NY
USA 12601-5400

Fold and Tape **Please do not staple** Fold and Tape

GG24-2563-00

Cut or Fold
Along Line

Printed in U.S.A.

GG24-2563-00