

Solaris™ 8 Operating Environment System Administration II

SA-288

Student Guide



Sun Microsystems, Inc.
MS BRM01-209
500 Eldorado Boulevard
Broomfield, Colorado 80021
U.S.A.

Revision A, July 2000

Copyright 2000 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun Logo, JumpStart, Solaris, Solaris Management Console, Solstice AdminSuite, Solaris WebStart, Solstice DiskSuite, StorEdge Volume Manager, SunRay, Java, JDK, AnswerBook, Jumpstart, JavaStation, Sun-4, and SunInstall are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

U.S. Government approval required when exporting the product.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g) (2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015 (b)(6/95) and DFAR 227.7202-3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.



Please
Recycle



Adobe PostScript

Contents

About This Course	xiii
Course Goal	xiii
Course Overview	xiv
Course Map	xv
Module-by-Module Overview	xvi
Course Objectives	xix
Skills Gained by Module	xx
Guidelines for Module Pacing	xxii
Topics Not Covered	xxiii
How Prepared Are You?	xxv
Introductions	xxvi
How to Use the Course Materials	xxvii
Course Icons and Typographical Conventions	xxviii
Icons	xxviii
Typographical Conventions	xxix
Introducing the Client-Server Relationship	1-1
Objectives	1-1
Additional Resources	1-1
The Client-Server Model for Network Workstations	1-3
Servers	1-3
Clients	1-4
Check Your Progress	1-6
Introducing the Solaris Network Environment	2-1
Objectives	2-1
Additional Resources	2-1
Overview	2-2
The Function of the Layers	2-3
Peer-to-Peer Communication	2-6
Encapsulation and De-encapsulation	2-6
Common Protocols and Applications in the Solaris Operating Environment	2-7
TCP/IP Protocol Descriptions	2-8

Network Interface Layer Protocols	2-8
Internet Layer Protocols.....	2-8
Transport Layer Protocols	2-9
Application Layer Protocols.....	2-9
Network Files and Commands	2-11
Displaying the MAC Address.....	2-12
The <code>ifconfig -a</code> Command.....	2-12
The <code>banner</code> Command.....	2-12
Configuring Interfaces at Boot Time	2-13
The <code>/etc/rcS.d/S30network.sh</code> File.....	2-13
The <code>/etc/hostname.xxx</code> File	2-13
The <code>/etc/hosts</code> File	2-14
Important Files and Utilities.....	2-15
The <code>/etc/nodename</code> File.....	2-15
Determining the Current Network Configuration.....	2-15
Network Troubleshooting Utilities	2-16
Network Services	2-18
The Internet Service Daemon (<code>inetd</code>)	2-18
Port Numbers	2-18
Remote Procedure Call (RPC).....	2-19
Checking for Registered Services	2-19
Stopping a Network Service.....	2-20
Starting a Network Service.....	2-20
Check Your Progress	2-21
Solaris Operating Environment <code>syslog</code>	3-1
Objectives	3-1
Additional Resources	3-1
The <code>syslog</code> Facility.....	3-2
Controlling the Behavior of <code>syslogd</code>	3-4
Configuring the <code>/etc/syslog.conf</code> File.....	3-5
Selector Field.....	3-5
Action Field.....	3-8
The <code>/etc/syslog.conf</code> File	3-9
Starting and Stopping <code>syslogd</code>	3-10
<code>syslogd</code> and the <code>m4</code> Macro Processor	3-11
Detailed Operation	3-12
Modifying <code>inetd</code> to Use <code>syslog</code>	3-14
<code>inetd</code> Manual Page Excerpt.....	3-14
The <code>inetd</code> Startup File	3-15
Example of <code>syslog</code> Logged Entry	3-15
The <code>logger</code> Utility	3-17
Command Format.....	3-17
Command Options	3-17
Examples	3-18
Exercise: Using <code>syslog</code> and Auditing Utilities	3-19

Preparation.....	3-19
Task Summary.....	3-19
Tasks	3-19
Exercise Summary.....	3-21
Check Your Progress	3-22
Introducing Disk Management.....	4-1
Objectives	4-1
Additional Resources	4-2
Physical Disks.....	4-3
Typical Physical Disk Drivers	4-3
Access Paths.....	4-3
Virtual Disk Access Paths	4-5
Virtual Volume Management.....	4-6
Solstice DiskSuite	4-6
Sun StorEdge Volume Manager.....	4-6
Concatenated Volumes	4-7
Adding a Disk.....	4-9
Reconfiguration Boot.....	4-9
The devfsadmd Daemon	4-9
Installing the Solstice DiskSuite Software	4-11
Solaris Product Registry	4-18
Starting the DiskSuite Tool.....	4-20
Creating the State-Database Replicas.....	4-23
Concatenating File Systems	4-28
Exercise: Managing Disks	4-37
Preparation.....	4-37
Task Summary.....	4-37
Tasks	4-38
Exercise Summary.....	4-44
Check Your Progress	4-45
Further Study.....	4-45
Solaris Pseudo File Systems and Swap Space.....	5-1
Objectives	5-1
Additional Resources	5-1
Solaris Pseudo File Systems.....	5-2
The /proc File System.....	5-3
The tmpfs File System.....	5-4
The fdfs File System.....	5-5
The swapfs File System	5-6
Virtual and Physical Addresses	5-7
Anonymous Memory Pages	5-7
Reserving Swap Space.....	5-7
Criteria for Swap Space.....	5-8
Swap Space	5-9
Using the swap Command.....	5-9

Command Format.....	5-9
Options	5-9
Adding a Swap File	5-10
Removing a Swap File.....	5-11
Adding a Swap Slice.....	5-12
Adding a Permanent Swap File Using the /etc/vfstab File.....	5-12
The dumpadm Command.....	5-13
Command Format.....	5-15
The coreadm Command.....	5-17
Command Format.....	5-17
Default coreadm Command Without Options	5-18
Patterns	5-19
Examples	5-20
Options Supported by coreadm.....	5-22
Exercise: Managing Pseudo File Systems and Swap Space	5-24
Preparation.....	5-24
Task Summary.....	5-24
Tasks	5-25
Exercise Summary.....	5-30
Task Solutions.....	5-31
Check Your Progress	5-32
NFS.....	6-1
Objectives	6-1
Additional Resources	6-1
The NFS Distributed File System.....	6-2
The Benefits of a Network File System	6-3
NFS Distributed File System Components.....	6-4
The NFS Daemons	6-5
The Mount Daemon.....	6-5
NFS Server Daemons.....	6-5
NFS Daemons on the Client and Server	6-6
NFS File Handles.....	6-6
The NFS Server.....	6-7
The share Command	6-7
The /etc/dfs/dfstab File	6-8
NFS Access Management.....	6-9
The unshare Command.....	6-11
The shareall and unshareall Commands	6-11
Configuring the NFS File Server.....	6-13
NFS Informational Commands.....	6-14
The dfshares Command	6-14
The dfmounts Command	6-15
The NFS Client	6-16
The mount Command	6-16

The /etc/vfstab File.....	6-17
Recommended Mounting Options.....	6-19
A Read-Only Directory	6-20
A Read-Write Directory	6-20
A Read-Only Application Directory	6-21
The umount Command.....	6-21
The mountall and umountall Commands	6-21
The mountall Command.....	6-21
The umountall Command.....	6-22
The NFS Client Setup	6-23
Mounting Using the /etc/vfstab File.....	6-23
NFS Server Logging.....	6-24
Enabling NFS Server Logging.....	6-25
The /etc/nfs/nfslog.conf File	6-26
The /etc/default/nfslogd File	6-28
Summary of NFS Commands, Files, and Daemons.....	6-29
Troubleshooting NFS Errors	6-30
rpcbind Failure Error	6-30
Server Not Responding Error.....	6-31
NFS Client Fails a Reboot Error	6-31
Stopped Server Error	6-32
Program Not Registered Error	6-33
Stale File Handle Error	6-34
Unknown Host Error.....	6-34
Mount Point Error.....	6-35
No Such File Error.....	6-35
Exercise: Configuring the NFS Environment.....	6-36
Preparation.....	6-36
Task Summary.....	6-36
Tasks	6-37
Exercise Summary.....	6-40
Task Solutions.....	6-41
Check Your Progress	6-44
AutoFS.....	7-1
Objectives	7-1
Additional Resources	7-1
AutoFS Overview.....	7-2
AutoFS Components	7-3
Automount Maps.....	7-5
Master Maps	7-6
Direct Maps.....	7-8
Indirect Maps.....	7-9
The automount Command	7-11
Command Format.....	7-11
The Client autofs File System.....	7-12

Multi-threaded autofs	7-13
Automount Administration	7-14
Setting up a Direct Map	7-14
Setting up an Indirect Map	7-15
Exercise: Using the Automounter.....	7-16
Preparation.....	7-16
Task Summary	7-16
Tasks	7-17
Exercise Summary.....	7-23
Task Solutions.....	7-24
Check Your Progress	7-29
CacheFS	8-1
Objectives	8-1
Additional Resources	8-1
CacheFS File System.....	8-2
Using CacheFS Terminology.....	8-3
Using CacheFS File System Commands	8-3
Creating a CacheFS File System.....	8-4
CacheFS Cache Directory Details	8-6
CacheFS Statistics and Consistency Checking.....	8-7
The cachefsstat Command.....	8-7
The cfsadmin Command	8-8
Enhancing CacheFS File System Caching.....	8-9
Sizing the Cache	8-11
CacheFS File System Integrity.....	8-13
Dismantling a CacheFS File System	8-14
Exercise: Configuring the CacheFS File System	8-16
Preparation.....	8-16
Task Summary	8-16
Tasks	8-17
Exercise Summary.....	8-20
Task Solutions.....	8-21
Check Your Progress	8-26
Role-Based Access Control	9-1
Objectives	9-1
Additional Resources	9-1
Role-Based Access Control	9-2
Components.....	9-3
Delimiters	9-4
Extended User Attributes Database (user_attr).....	9-6
Authorizations.....	9-9
Execution Profiles	9-12
Execution Attributes.....	9-15
Assuming Role-Based Access Control	9-19
Tools for Managing Role-Based Access Control	9-20

The roleadd Command.....	9-20
The rolemod Command.....	9-21
The useradd Command.....	9-22
Additional Commands.....	9-23
Creating a User and a Role	9-24
Testing the Configuration.....	9-24
Exercise: Implementing System Security.....	9-25
Preparation.....	9-25
Task Summary.....	9-25
Tasks	9-26
Exercise Summary.....	9-36
Check Your Progress	9-37
Solaris Management Console™ and Solaris AdminSuite	10-1
Objectives	10-1
Additional Resources	10-2
The Solaris Management Console	10-3
The Benefits of Using the Console.....	10-3
Installation Requirements.....	10-4
Download Procedure.....	10-5
Installing SMC	10-6
Running the SMC Application.....	10-12
Solaris AdminSuite	10-15
User Manager	10-15
Group Manager	10-15
Host Manager	10-16
Mount/Share Manager	10-16
Serial Port Manager	10-16
Installation Procedure	10-17
Selecting a Name Service	10-25
Solaris AdminSuite Components	10-28
Viewing Users	10-30
Adding Users.....	10-34
Viewing Groups	10-43
Adding Groups	10-47
Modifying Groups	10-50
Adding a Host	10-55
Renaming a Host.....	10-58
File System Usage	10-66
Configuring Serial Ports.....	10-71
Check Your Progress	10-74

Naming Services Overview	11-1
Objectives	11-1
Additional Resources	11-1
Name Services Overview.....	11-2
Available Name Services	11-3
DNS Overview	11-5
The DNS nsswitch Template	11-6
Top-Level Domains	11-6
Network Information Service Overview	11-7
NIS Domains.....	11-7
Client-Server Arrangement	11-7
NIS Maps.....	11-7
The NIS nsswitch Template.....	11-8
The NIS+ Environment	11-9
NIS+ Namespace.....	11-9
An Example of the NIS+ Hierarchical Namespace	11-10
NIS+ Tables.....	11-11
The NIS+ nsswitch Template	11-11
Lightweight Directory Access Protocol (LDAP) Overview	11-12
Common Uses of LDAP	11-13
The LDAP nsswitch Template	11-14
The Name Service Switch	11-15
The nsswitch.conf Configuration Files.....	11-15
The /etc/nsswitch.nis Template.....	11-16
Modification of the /etc/nsswitch.conf File.....	11-18
Name Service Switch Status and Action Values	11-19
Exercise: Reviewing Naming Services	11-21
Preparation.....	11-21
Tasks	11-21
Exercise Summary.....	11-23
Task Solutions.....	11-24
Check Your Progress	11-26
NIS	12-1
Objectives	12-1
Additional Resources	12-1
Introduction to NIS Concepts	12-2
NIS Master Server	12-2
NIS Slave Servers	12-3
NIS Clients	12-3
NIS Processes.....	12-4
The ypserv Daemon.....	12-5
The ypbind Daemon.....	12-5
The rpc.yppasswdd Daemon	12-5
The ypxfrd Daemon.....	12-6
The rpc.yupdated Daemon	12-6

The Structure of NIS Maps	12-7
NIS Maps Filenames	12-7
Map Contents and Sort Keys	12-8
Commands to Read Maps	12-8
Generating NIS Maps	12-9
The ypinit Command and the NIS Makefile	12-11
Password File	12-11
Configuring the NIS Master Server	12-13
Accessing and Testing the NIS Service	12-17
Configuring the NIS Client	12-19
Configuring the NIS Slave Server	12-20
Updating the NIS Map	12-22
Updating the Hosts Map and Propagating to Slave Servers	12-22
Updating the NIS Password Map	12-23
Updating the NIS Slave Server Map	12-25
Updating Other Scripts	12-26
Makefile Syntax and New Maps	12-28
The make Utility	12-28
First Section of Makefile	12-29
Second Section of Makefile	12-29
Fourth Section of Makefile	12-30
Third Section of Makefile	12-31
Building NIS Maps	12-32
Exercise: Configuring NIS	12-34
Preparation	12-34
Task Overview	12-34
Tasks	12-36
Exercise Summary	12-47
Check Your Progress	12-48
JumpStart™ – Automatic Installation	13-1
Objectives	13-1
Additional Resources	13-2
Introduction to JumpStart	13-3
Who Should Use JumpStart and Why?	13-3
JumpStart Components	13-4
Adding Install Clients	13-5
Setting Up Boot Services	13-6
JumpStart Client Boot Sequence	13-7
Boot Operation Support Files	13-9
Adding a Bootable Image	13-14
Using add_install_client	13-14
Setting Up Client Identification	13-16
Using the sysidcfg File to Identify a Client	13-17
Setting Up Locale	13-20

Setting Up an Install Server	13-24
The <code>add_to_install_server</code> Script	13-25
The <code>modify_install_server</code> Script	13-25
Setting up the Configuration Server	13-26
Setting Up a Configuration Server Directory	13-27
Creating the rules File	13-29
Creating the Class Files	13-33
Keywords and Arguments	13-33
Testing the Configuration with the <code>pfinstall</code> Command ...	13-36
Running the <code>pfinstall</code> Command.....	13-36
Using <code>add_install_client</code>	13-40
Running the <code>add_install_client</code> Script	13-40
Adding a Client Using a Solaris CD-ROM Image on the Local Disk.....	13-41
Adding a Client Using a Solaris CD-ROM Image From the CD-ROM.....	13-42
The <code>/etc/bootparams</code> File Content	13-42
The <code>/etc/dfs/dfstab</code> File Content.....	13-43
Initiating a JumpStart Installation	13-45
Boot Install Clients	13-45
JumpStart Capabilities and Limitations	13-46
Worksheet for Configuring for JumpStart Installation Exercise	13-49
Exercise: Configuring for a JumpStart Installation	13-50
Preparation.....	13-50
Tasks	13-51
Exercise Summary.....	13-55
Check Your Progress	13-56
Solaris Administrator Workshop	14-1
Objectives	14-1
Solaris Operating Environment Administration Workshop ...	14-2
Configuration Goals	14-2
Check Your Progress	14-12
The JumpStart rules and Class Files	A-1
Initial Install Keywords and Arguments.....	A-4
Upgrade Parameters.....	A-11
Time Zones.....	A-12
System Configuration Using NIS+	A-13
Configuring System Information.....	A-14
Setting Up Time Zone	A-15
Adding Time Zone Information to the Name Server Switch File	A-15

About This Course

Course Goal

The *Solaris™ 8 Operating Environment System Administrator II* course provides students with the skills necessary to administer Sun™ workstations running the Solaris™ 8 Operating Environment in a network environment.

This course describes how to install and maintain Sun systems, configure and troubleshoot the network file system (NFS) environment, and configure the network information service (NIS) environment.

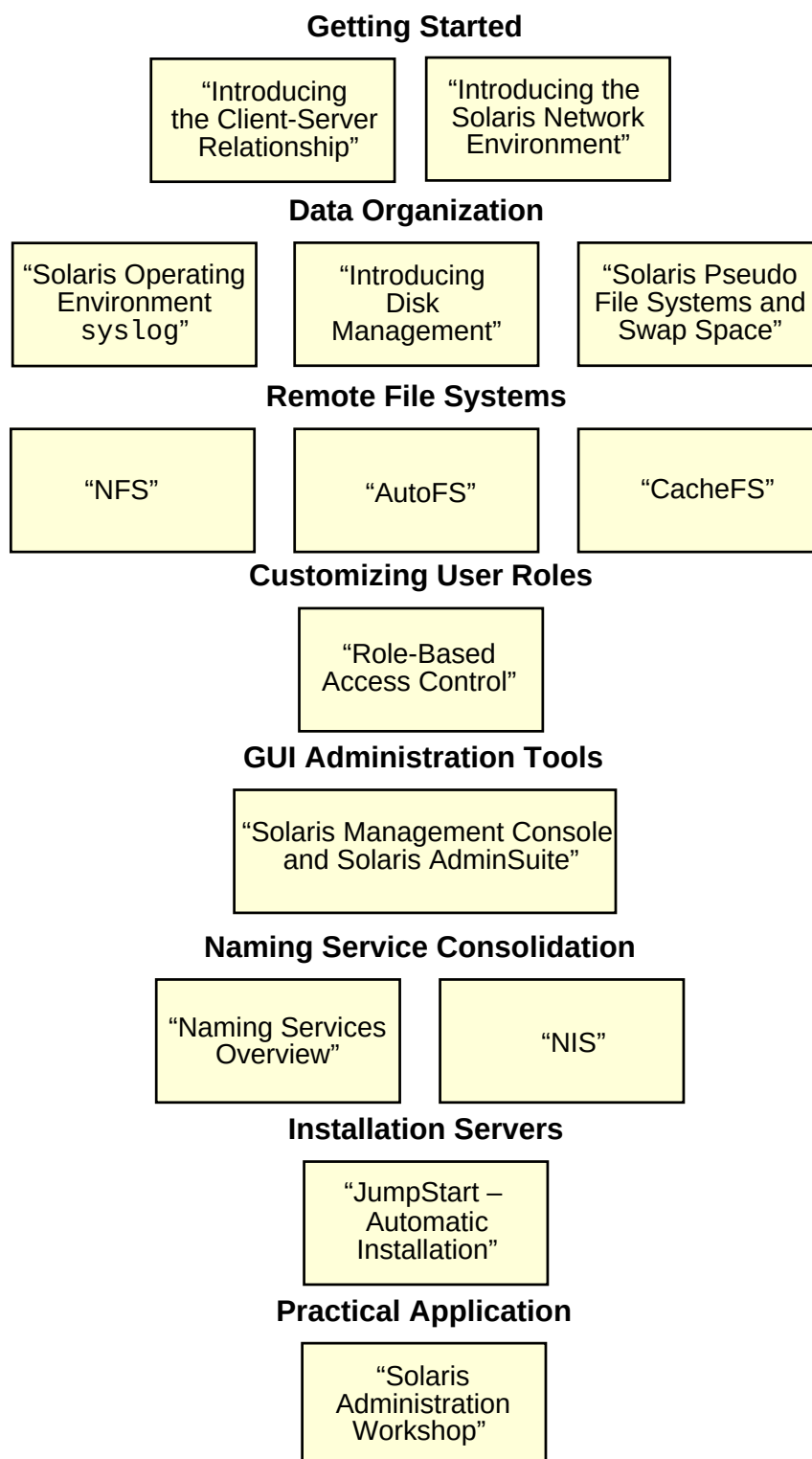
Course Overview

The course is for technical and application support staff who administer a network server running the Solaris Operating Environment. You will learn fundamental network administration skills, including how to:

- Control system logging activity
- Virtual disk management
- Configure the NFS environment
- Automount file systems on demand
- Administer the NIS
- Create system administrator role-based access permissions
- Use the JumpStart™ program within a subnetwork

Course Map

The following course map enables you to see what you have accomplished and where you are going in reference to the course goal.



Module-by-Module Overview

This course contains the following modules:

- Module 1 – “Introducing the Client-Server Relationship”

This module introduces the client-server computing model within the Solaris Operating Environment.

Lab exercise – There is no lab exercise planned for this module.

- Module 2 – “Introducing the Solaris Network Environment”

This module uses the standard network models to describe the configuration of interfaces and services in the Solaris Operating Environment.

Lab exercise – There is no lab exercise for this module.

- Module 3 – “Solaris Operating Environment `syslog`”

This module focuses on the use of the `syslog` utility to manipulate message logs.

Lab exercise – In this lab, you configure and use the `syslog` utility to better monitor your system

- Module 4 – “Introducing Disk Management”

This module provides an introduction to disk management, redundant array of inexpensive disks (RAID) concepts, and the Solstice DiskSuite™ graphical user interface (GUI). You use the tools and concepts learned in this module to grow file systems.

Lab exercise – In this lab, you install the necessary tools and perform a `growfs` operation on a file system that is nearing capacity.

- Module 5 – “Solaris Pseudo File Systems and Swap Space”

This module describes pseudo file systems and swap space.

Lab exercise – In this lab, you manipulate swap space, such as adding (and removing) a swap file, adding a swap slice, and adding permanent swap files to the `/etc/vfstab` file.

- Module 6 – “NFS”

This module addresses networked file systems, the relationship between the NFS server and the NFS client, and the commands that share these file systems.

Lab exercise – In this lab, you configure an NFS server and client to share and mount `/usr/share/man`.

- Module 7 – “AutoFS”

This module provides information on the benefits of using `automount`, the purpose of each type of automount map, and describes how and when to start and stop the automount daemon.

Lab exercise – In this lab, you use the automounter processes to automatically mount manual pages and a user’s home directory.

- Module 8 – “CacheFS”

This module describes the need for cached file systems, how to properly configure `cacheFS`, how to set up `cacheFS` logging, and how to dismantle and delete a `cacheFS` file system.

Lab exercise – In this lab, you mount `/usr/share/man` to a `cacheFS` file system.

- Module 9 – “Role-Based Access Control”

This module configures a user’s execution profile to allow access to a specified subset of system administrator privileges.

Lab exercise – In this lab, you create an execute attribute, a role-based profile, a role identity, and a login identity that can use the role.

-
- Module 10 – “Solaris Management Console™ and Solaris AdminSuite”

This module focuses on the benefits and features of the Solaris Management Console graphical user interface (GUI) and one subordinate GUI, Solaris AdminSuite.

Lab exercise – In this lab, you use a guided-practice to use the GUI features.

- Module 11 – “Naming Services Overview”

This module describes the name service concept, lists the name services available, compares name services functionality, describes the name service switch process, and describes how to determine the configuration that is appropriate for your network.

Lab exercise – There is no lab exercise for this module. Instead, there is a short comprehension check in the form of a question-and-answer exercise.

- Module 12 – “NIS”

This module describes the NIS components, master server, slave server, and client, and the NIS processes. In addition, this module describes how to configure an NIS master, slave, and client, and the steps necessary to add a new NIS map, update maps, and propagate changes to NIS maps.

Lab exercise – In this lab, you configure a NIS master server and one NIS client.

- Module 13 – “JumpStart™ Automatic Installation”

This module describes the JumpStart program. This includes a description, configuration information, and usage of the program. Causes of many common error messages are defined, and possible solutions are offered.

Lab exercise – In this lab, you configure a JumpStart server to support an installed client.

- Module 14 — “Solaris Administrator Workshop”

This module offers a comprehensive, hands-on lab workshop reviewing the major course material covered in earlier modules. This module consists only of lab exercises.

Course Objectives

Upon completion of this course, you should be able to:

- Define client-server functionality within the Solaris 8 Operating Environment
- Describe the functional characteristics between each layer in the seven-layer Open Systems Interconnect (OSI) network model and the five-layer Transmission Control Protocol/Internet Protocol (TCP/IP) network model
- Identify and describe the files and commands that control and monitor access to various machines in a network environment
- Set up event logging
- Administer disks using a volume management utility
- Redirect all core dumps to a single `coredump` directory
- Configure NFS to support the client-server environment
- Use the automounter
- Set up and configure `cacheFS` file systems
- Use Solaris AdminSuite to perform user, group, and serial port management duties
- Describe the various naming services: Domain Name System (DNS), Network Information System (NIS), Network Information System Plus (NIS+), and Lightweight Directory Access Protocol (LDAP)
- Configure and administer the NIS environment
- Set up a role-based system administration account
- Install and configure the Solaris Management Console GUI
- Use JumpStart to automate the Solaris Operating Environment installation

Skills Gained by Module

The skills for *Solaris 8 Operating Environment System Administration II* are shown in the first column of the following matrix. The black boxes indicate the main coverage for a topic; the gray boxes indicate the topic is briefly discussed.

Skills Gained	Module												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Define client-server functionality within the Solaris 8 Operating Environment													
Describe the functional characteristics between each layer in the seven-layer OSI network model and the five-layer TCP/IP network model													
Identify and describe the files and commands that control and monitor access to various machines in a network environment													
Set up event logging													
Administer disks using a volume-management utility													
Redirect all core dumps to a single coredump directory													
Configure NFS to support the client-server environment													
Use the automounter													
Set up and configure CacheFS file systems													
Use Solaris AdminSuite to perform user, group, and serial port management duties													
Describe the various naming services: DNS, NIS, NIS+, and LDAP													
Configure and administer the NIS environment													
Set up a role-based system administration account													
Install and configure the Solaris Management Console GUI													

Skills Gained	Module												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Use JumpStart to automate the Solaris Operating Environment installation													

Guidelines for Module Pacing

The following table provides a rough estimate of pacing for this course:

Module	Day 1	Day 2	Day 3	Day 4	Day 5
"About This Course"	A.M.				
"Introducing the Client-Server Relationship"	A.M.				
"Introducing the Solaris Network Environment"	A.M. /P.M.				
"Solaris Operating Environment syslog"	P.M.				
"Introducing Disk Management"		A.M.			
"Solaris Pseudo File Systems and Swap Space"		A.M. /P.M.			
"NFS"		P.M.			
"AutoFS"			A.M.		
"CacheFS"			A.M.		
"Role-Based Access Control"				P.M.	
"Solaris Management Console and Solaris AdminSuite"				A.M.	
"Naming Services Overview"			A.M.		
"NIS"			A.M. /P.M.		
"JumpStart – Automatic Installation"				A.M. /P.M.	
"Solaris Administration Workshop"				P.M.	A.M. /P.M.

Topics Not Covered

This course does not cover the topics shown below. Many of the topics mentioned here are covered in other courses offered by Sun Educational Services:

- Basic UNIX commands – Covered in SA-118: *Fundamentals of Solaris 8 for System Administrators*
- vi editor – Covered in SA-118: *Fundamentals of Solaris 8 for System Administrators*
- Basic UNIX file security – Covered in SA-118: *Fundamentals of Solaris 8 for System Administrators*
- Software package administration – Covered in SA-238: *Solaris 8 Operating Environment System Administration I*
- Patch maintenance – Covered in SA-238: *Solaris 8 Operating Environment System Administration I*
- Addition of users using Admintool – Covered in SA-238: *Solaris 8 Operating Environment System Administration I*
- Basic system security – Covered in SA-238: *Solaris 8 Operating Environment System Administration I*
- Administration of initialization files – Covered in SA-238: *Solaris 8 Operating Environment System Administration I*
- Advanced file permissions – Covered in SA-238: *Solaris 8 Operating Environment System Administration I*
- Backup and recovery – Covered in SA-238: *Solaris 8 Operating Environment System Administration I*
- The lp print service and print commands – Covered in SA-238: *Solaris 8 Operating Environment System Administration I*
- Process control – Covered in SA-238: *Solaris 8 Operating Environment System Administration I*
- Hardware or software troubleshooting – Covered in ST-350: *Sun Systems Fault Analysis Workshop*
- System tuning – Covered in SA-400: *Enterprise System Performance Management*

-
- Detailed Shell Programming – Covered in SA-225: *Advanced UNIX® Utilities and Shell Programming for Administrators*
 - Detailed network administration concepts – Covered in SA-389: *Solaris™ TCP/IP Network Administration*

Refer to the Sun Educational Services catalog for specific information on course content and registration.

How Prepared Are You?

To be sure you are prepared to take this course, can you perform the activities that are listed below?

- Install and boot the Solaris 8 Operating Environment on a standalone workstation
- Implement basic system security
- Add users to the system using Admintool
- Use the pkgadd command to add software packages
- Set file permissions using access control lists (ACLs)
- Monitor and mount file systems
- Manage disk devices and processes
- Perform backups and restorations

Introductions

Now that you have been introduced to the course, introduce yourself to each other and the instructor, addressing the items shown on the overhead.

How to Use the Course Materials

To enable you to succeed in this course, these course materials employ a learning model including the following components:

- **Course Map** – An overview of the course content appears in the “About This Course” module so you can see how each module fits into the overall course goal.
- **Objectives** – At the beginning of each module is a list of what you should be able to accomplish after completing the module.
- **Lecture** – The instructor presents information specific to the topic of the module. This information helps you learn the knowledge and skills necessary to succeed with the exercises.
- **Exercise** – Lab exercises give you the opportunity to practice your skills and apply the concepts presented in the lecture.
- **Check Your Progress** – Module objectives are restated, sometimes in question format, so you are sure you can accomplish the objectives of the current module before moving on in the course.

Course Icons and Typographical Conventions

The following icons and typographical conventions are used in this course to represent various training elements and alternative learning resources.

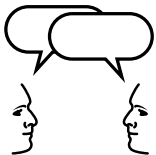
Icons



Additional resources – Indicates additional reference materials are available.



Demonstration – Indicates a demonstration of the current topic is recommended at this time.



Discussion – Indicates a small-group or class discussion on the current topic is recommended at this time.



Exercise objective – Indicates the objective for the lab exercises that follow. The exercises are appropriate for the material being discussed.

Note – This contains additional important, reinforcing, interesting, or special information.



Caution – This points out a potential hazard to data or machinery.



Warning – This warns of anything that poses personal danger or irreversible damage to data or the operating system.

Typographical Conventions

Courier is used for the names of commands, files, and directories, as well as on-screen computer output; for example:

```
Use ls -al to list all files.  
system% You have mail.
```

Courier bold is used for characters and numbers you type; for example:

```
system% su  
Password:
```

Courier italic is used for variables and command-line placeholders that are replaced with a real name or value; for example:

To delete a file, type `rm filename`.

Palatino italics is used for book titles, new words or terms, or words that are emphasized; for example:

Read Chapter 6 in *User's Guide*.
These are called *class* options.
You *must* be *root* to do this.

Introducing the Client-Server Relationship

1 

Objectives

Upon completion of this module, you should be able to:

- Describe the network capabilities of the Solaris 8 Operating Environment in terms of a client-server relationship
- List and define at least two types of servers used in the Solaris 8 network environment
- Describe the SunRay™ Enterprise Appliance client-server relationship

Additional Resources



Additional resources – The following references provide additional details on the topics discussed in this module:

- *About Solaris 8 Documentation*, Part Number 805-6333-06

Advanced System Administrator Activities

This course prepares you to perform system administration tasks in a network desktop environment. It includes modules to:

- Introduce the Solaris network environment, which provides a basic overview of how the Open Systems Interconnect (OSI) model and the Transmission Control Protocol/Internet Protocol (TCP/IP) model apply to a network of desktop systems.
- Explain the purpose and the configuration of the `syslog` system logging utility.
- Configure a disk-management utility to create a virtual volume spanning multiple disks.
- Identify pseudo file system types, and configure a file system as a temporary storage device.
- Set up network file systems to reduce network resource requirements.
- Manipulate local cache disk devices to improve system performance.
- Configure file systems to be mounted on demand and unmounted after a predefined period of inactivity.
- Identify the naming services that are available with the Solaris 8 Operating Environment.
- Provide procedures to centrally administer repositories of administrative files.
- Create role-based execution profiles for users.
- Manipulate the Solaris AdminSuite system administrator GUI tool.
- Configure a JumpStart™ server and build boot profiles to use on the server.

The Client-Server Model for Network Workstations

A key advantage to network environments is the ability to configure applications to run using the client-server model. In today's enterprise-oriented computing environment, applications are often distributed across a network of client-server systems.

Servers

A *server* is a host or a *process* that provides services to other systems on the network. These hosts can include:

- Application servers – Provide network access to applications, such as FrameMaker, Photoshop, and Lotus Notes.
- Boot servers – Respond with boot parameters to install clients booting from the network.
- Installation servers – Enable network file system (NFS) access to release software to installing clients.
- Database servers – Offer database access through applications, such as Oracle, Sybase, and Informix.
- Mail servers – Provide inter-networking transfer of email from the local network.
- home directory servers – Export home directories (typically from /export/home) for user access.
- License servers – Run a license daemon that manages the use of licenses on a per-system, per-application basis.
- Print servers – Offer print services to network systems. Printers can either be locally attached or, in the case of printers with Ethernet interfaces, provide spooling services to a printer with an IP address.
- Name services server – Provide a naming service, such as DNS Domain Name System (DNS), Federated Name Service (FNS), Network Information Service (NIS), or Network Information Service Plus (NIS+).

The term *server* infers that the host offers file systems and services to other network hosts referred to as *clients*.

Clients

A *client* is a host or a *process* that uses services provided by servers. An example of a client is a host that makes use of any of the previously-mentioned servers. One example of a client is the JumpStart client. Figure 1-1 illustrates a JumpStart client-server relationship.

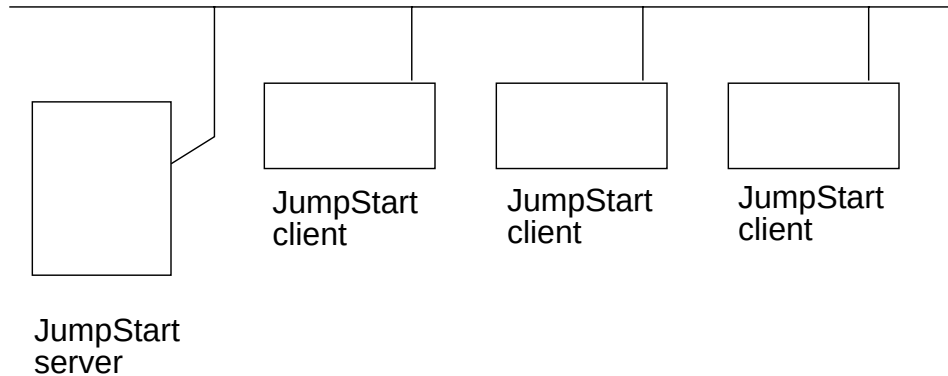


Figure 1-1 JumpStart Client-Server Configuration

You can install a JumpStart client automatically from a preconfigured JumpStart server.

Another approach to client-server architecture is demonstrated with the SunRay Enterprise Appliance. The SunRay architecture uses an approach that removes everything from the desktop except the physical components a user needs: Keyboard, mouse, display, and audio input and output.

All computing is performed on centralized, shared machines, thereby making the SunRay architecture the ultimate in thin-client technology. Everything that previously ran on the user's own desktop, such as a window system, user applications, or mail clients, now runs in a session on the server and is displayed on the SunRay Enterprise Appliance.

Figure 1-2 on page 1-5 illustrates the relationship between the enterprise appliances (clients) and the centralized server.

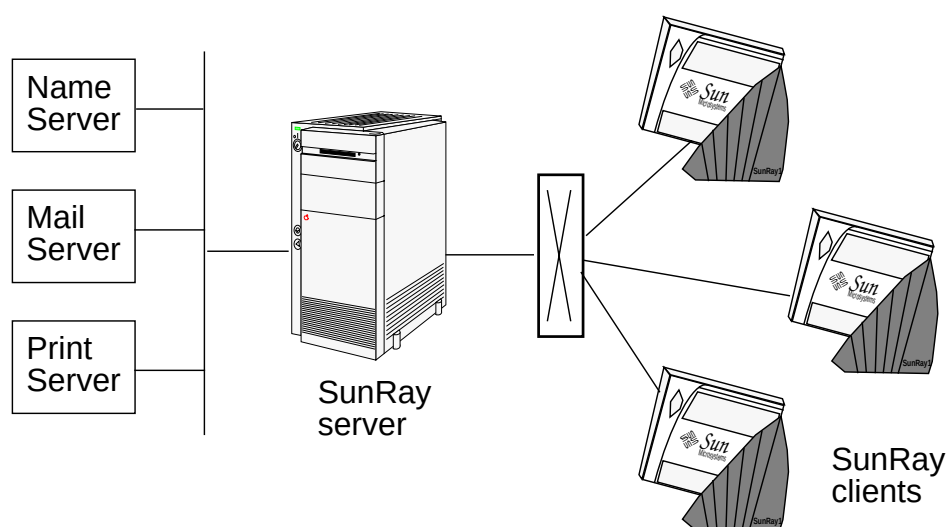


Figure 1-2 SunRay Enterprise Appliance Client-Server

Note – The SunRay Enterprise Appliances can communicate only to the SunRay server; therefore, if any of the previously-mentioned server services are provided by systems other than the SunRay server, it becomes a client system to other server systems on the network.

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ Describe the network capabilities of the Solaris 8 Operating Environment in terms of a client-server relationship
- ☐ List and define at least two types of servers used in the Solaris 8 network environment.
- ☐ Describe the SunRay Enterprise Appliance client-server relationship

Introducing the Solaris Network Environment

2 

Objectives

Upon completion of this module, you should be able to:

- Define the function of each layer within the seven-layer Open Systems Interconnect (OSI) model and the five-layer Transmission Control Protocol (TCP)/Internet Protocol (IP) model
- Describe the contents of various network control files
- Construct command strings to perform basic monitoring operations on an active network
- Start and stop network services using the command line

Additional Resources



Additional resources – The following references provide additional details on the topics discussed in this module:

- *System Administration Guide, Volume 3*, Part Number 806-0916-10.

Overview

The standard Solaris Operating Environment comes with the TCP/IP stack built into it. To understand the protocol stack, you must understand network models.

The most commonly referred to networking models are the seven-layered International Standards Organization (ISO)/ OSI model and the five-layered TCP/IP network model.

Both models provide a framework for describing data communications.

Note – The U.S. Department of Defense created the TCP/IP model.

This module references the TCP/IP model.

The Function of the Layers

Each layer in either of the two network models describes a specific network function. Each function supports the layer above and receives support from the layer below.

The separation of the data-communication process into distinct functions makes it easier for developers to design network components that inter-operate with each other, regardless of the vendor.

Each layer uses separate protocols, such as the IP or the TCP, to complete the required tasks for the particular layer.

The functions of data delivery and connection management are handled by separate protocols. The data delivery protocol is simple and does not deal with connection management. Conversely, the connection management protocol is also simple because it does not concern itself with data delivery.

This separation of the data delivery functions from the connection management functions helps to simplify development using these protocols.

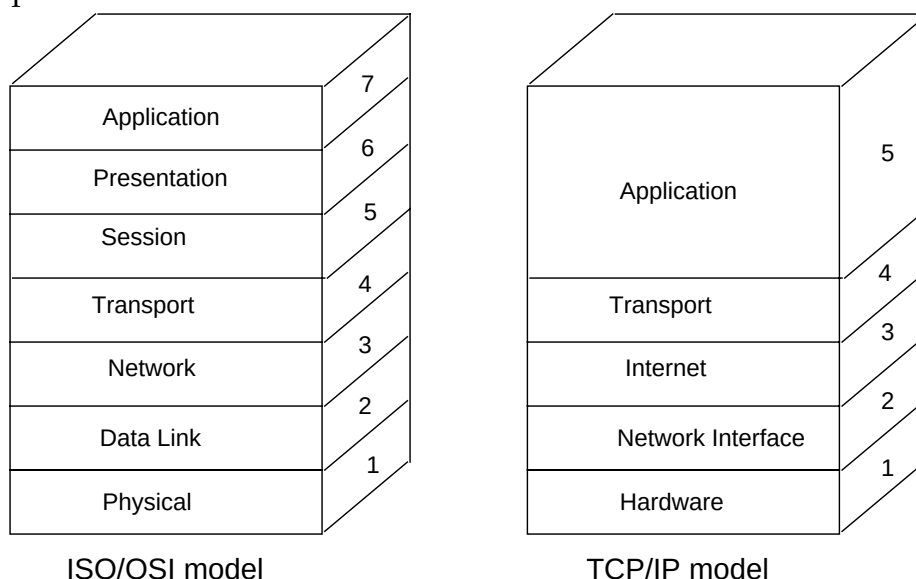


Figure 2-1 ISO/OSI and TCP/IP Model Layers

Protocol layering produces simple protocols, each with a few well-defined tasks. You can then assemble these protocols into a useful whole. You can also remove or replace individual protocols as needed for particular applications.

The function of the individual layers of the ISO/OSI model are described in Table 2-1.

Table 2-1 ISO/OSI Network Model

ISO/OSI Layer	Function
Application	Manages user-accessed application programs and network services (using the underlying layers).
Presentation	Manages the presentation of the data to be independent of the architecture.
Session	Manages communication setup and termination.
Transport	Ensures that messages reach the correct application.
Network	Manages data addressing and delivery between networks, as well as fragmenting data for the Data Link layer. A router functions at this layer by using IP addresses.
Data Link	Manages the delivery of data across the physical network. This layer provides error detection and packet framing. A bridge/switch functions at this layer. Delivery decisions are based on the ethernet address (also known as the Media Access Control [MAC] address).
Physical	Describes the network hardware, including electrical signal characteristics, such as voltage and current. A repeater functions at this layer.

The function of the individual layers of the TCP/IP model are listed in Table 2-2.

Table 2-2 TCP/IP Network Model

TCP/IP Layer	Function
Application	Manages user-accessed application programs and network services (using the underlying layers), manages the presentation of the data to be independent of the architecture, and manages the presentation of the data to be independent of the architecture.
Transport	Ensures that messages reach the correct application.
Internet	Manages data addressing and delivery between networks, as well as fragmenting data for the data link layer. A router functions at this layer by using IP addresses.
Network Interface	Manages the delivery of data across the physical network. This layer provides error detection and packet framing. A bridge/switch functions at this layer. Delivery decisions are based on the ethernet address (also known as the Media Access Control [MAC] address).
Hardware	Describes the network hardware, including electrical signal characteristics, such as voltage and current. A repeater functions at this layer.

Peer-to-Peer Communication

In contrast to the client/server model, the peer-to-peer communication model is one in which each party has the same capabilities and either party can initiate communication.

Encapsulation and De-encapsulation

When systems communicate with each other, data can be thought of as flowing down the model from the application layer to the hardware layer, across the network connection, and then flowing up the model on the target system from the hardware layer to the application layer.

A header is added to each segment received on the way down the model, and a header is removed from each segment on the way up the model, as shown in Figure 2-2.

Each header contains specific address information so that the layers on the remote system know how to forward the communication.

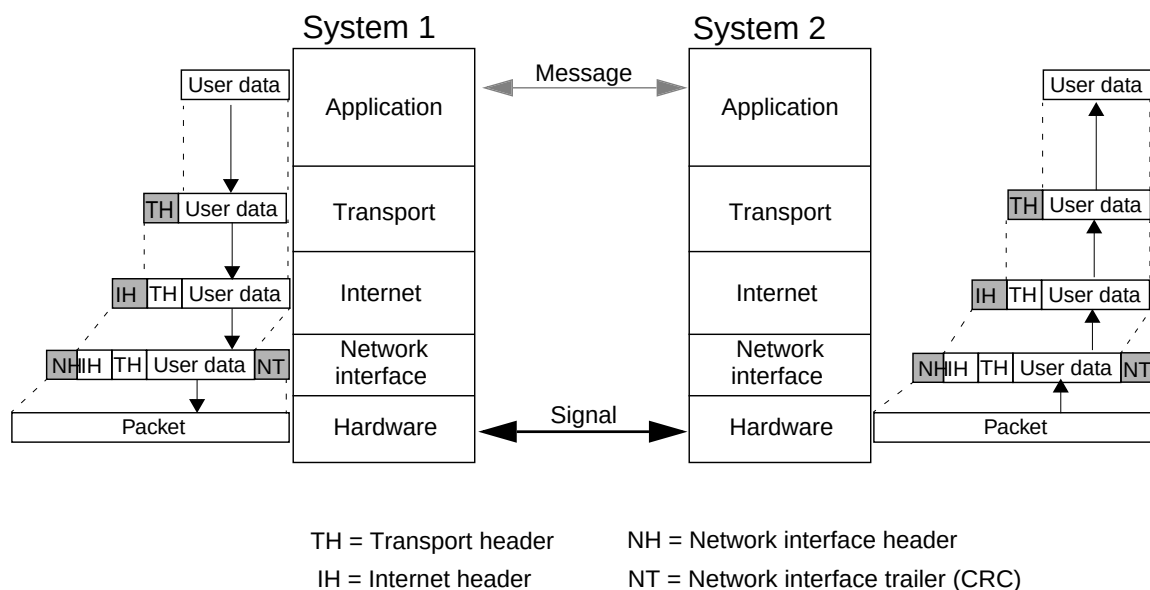


Figure 2-2 Simplified Encapsulation and De-encapsulation Between System 1 and System 2 Using the TCP/IP Model

Common Protocols and Applications in the Solaris Operating Environment

A protocol is a set of rules governing the exchange of data between two entities. Protocols can exist at each layer in a telecommunication session. Both end points must recognize and observe the protocols.

Protocols are described in an industry or international standard. For example, on the Internet, there are the TCP/IP protocols consisting of:

- Transmission Control Protocol (TCP), which uses a set of rules to exchange messages with other Internet points at the information packet level.
- Internet Protocol (IP), which uses a set of rules to send and receive messages at the Internet-address level.

Specific protocols are related with each layer of the network models. Table 2-3 shows some of the protocols associated with each layer of the TCP/IP network model.

Table 2-3 Protocols and Network Model Layers

TCP/IP Layer	TCP/IP Protocol and Applications
Application	NFS, NIS+, DNS, SMTP, DHCP, SNMP, HTTP, RPC, RIP, rlogin, telnet, and ftp,
Transport	TCP and UDP
Internet	IP, ARP, RARP, and ICMP
Network interface	Ethernet, ATM, FDDI, and PPP

TCP/IP Protocol Descriptions

The following sections describe the TCP/IP protocols.

Network Interface Layer Protocols

The network layer protocols consist of the following:

- Ethernet is a type of local area network (LAN) that enables real-time communication between machines connected directly through cables.
- Asynchronous Transfer Mode (ATM) is a dedicated, connection-switching technology that organizes digital data into 53-byte cell units and transmits them over a physical medium using digital signal technology.
- Fiber Distributed Data Interface (FDDI) specifies a 100-Mbytes-per-second, token-passing, dual-ring LAN using a fiber-optic transmission medium. It defines the physical layer and media-access portion of the link layer.
- Point-to-Point Protocol (PPP) transmits IP datagrams over serial point-to-point links.

Internet Layer Protocols

The internet layer protocols consist of the following:

- Internet Protocol (IP) determines the path a packet must take, based on the destination host's IP address. Both IPv4 and IPv6 are supported.
- Address Resolution Protocol (ARP) defines the method that map a 32-bit IP address to a 48-bit Ethernet address.
- Reverse Address Resolution Protocol (RARP) is the reverse of ARP. It maps a 48-bit Ethernet address to a 32-bit IP address.

ARP and RARP are not used in Internet Protocol, version 6 (IPv6). The network layer protocols consist of the following:

- Internet Control Message Protocol (ICMP) defines a set of error and diagnostic feedback messages for the IP. ICMP has support for IPv4 (with ICMPv4) and IPv6 (with ICMPv6).

Transport Layer Protocols

The transport layer protocols consist of the following:

- Transmission Control Protocol (TCP) is a connection oriented protocol that provides the full duplex, reliable service on which many application protocols depend.
- User Datagram Protocol (UDP) provides a half-duplex, non-acknowledged delivery service.

Application Layer Protocols

The application layer protocols consist of the following:

- Network File System (NFS) is a client-server application that enables you to view and, optionally, store and update files on a remote system as though they were on your own system.
- Network Information System (NIS) and Network Information System Plus (NIS+) are network-naming and administration systems.
- Dynamic Host Configuration Protocol (DHCP) automates the assignment of IP addresses in an organization's network.
- Domain Name System (DNS) is a distributed database that maps host names to IP addresses.
- Hypertext Transfer Protocol (HTTP) is used by the world wide web to display text, pictures, sounds, and other multimedia information with a web browser.
- Remote Procedure Call (RPC) is a protocol that one program can use to request service from a on another system in the network without needing to understand network details.

- Routing Information Protocol (RIP) provides for automated distribution of routing information between systems.
- Simple Mail Transport Protocol (SMTP) provides for delivery of mail messages.
- Simple Network Management Protocol (SNMP) is the language that allows for the monitoring and control of network devices.
- `rlogin` is a service, offered primarily by UNIX® systems, which enables users of one system to connect to other systems across the intranet, and to interact as if their terminals were connected to the systems directly.
- `telnet` is a service that enables users of one system to connect to other systems across the Intranet, and to interact as if their terminals were connected to the systems directly.
- File Transfer Protocol (FTP) transfers a file by copying a file from one system to another system.

Network Files and Commands

You must configure network interfaces to allow peer-to-peer communication. You can use many files and commands to manipulate the networking characteristics of a system installed with the Solaris Operating Environment.

This section introduces to some of the common files and commands, including those used for:

- Identifying a host
- Determining network configuration
- Troubleshooting a network
- Providing network services
- Providing remote procedure calls

Displaying the MAC Address

There are numerous ways to display a system's hardware address, also known as the media access control (MAC) address and as the Ethernet address. The MAC address is usually required by system administrators when configuring a system needing to be jump-started.

The ifconfig -a Command

You can use the `ifconfig` command with the `-a` switch to display the system's hardware address. This address is displayed only if the root user issues the `ifconfig` command. Only the IP address information is displayed if a non-root user issues the `ifconfig` command.

```
# ifconfig -a
lo0: flags=849<UP,LOOPBACK,RUNNING,MULTICAST> mtu 8232
    inet 127.0.0.1 netmask ff000000
hme0: flags=863<UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST> mtu 1500
    inet 192.168.10.25 ffffffff00 broadcast 192.168.10.255
    ether 8:0:20:a2:11:de
#
```

The banner Command

You can also retrieve the MAC address from a system that has not yet been booted by typing **banner** at the ok prompt.

ok **banner**

```
Sun Ultra 5/10 UPA/PCI (UltraSPARC-IIIi 300MHz), Keyboard Present
OpenBoot 3.1.1 64 MB memory installed, Serial #9361102.
Ethernet address 8:0:20:8e:d6:ce, HostID: 808ed6ce.
```

Configuring Interfaces at Boot Time

System interfaces can be automatically configured at boot time if the supporting files have appropriate entries.

The /etc/rcS.d/S30network.sh File

The `/etc/rcS.d/S30network.sh` file is one of the startup scripts that is run each time the system is booted. This script uses the `ifconfig` utility to configure each interface with an IP address and other required network information. The script searches for files called `hostname.xxn` in the `/etc` directory where `xx` is an interface type and `n` is the instance of the interface. The `/etc/hostname.hme0` is an example of a host-name file.

*The /etc/hostname.xx*n* File*

The `/etc/hostname.xxn` file contains only an entry for the interface. This host name must exist in the `/etc/hosts` file so that it can resolve to an IP address at system boot time. An example of the file contents is:

```
# cat /etc/hostname.hme0
host1
#
```

Note – Creating an empty `/etc/hostname6.xxn` file causes the Solaris Operating Environment to automatically generate an IP address for the IPv6 interface. This also occurs if the IPv6 is enabled during installation of the Solaris Operating Environment.

The /etc/hosts File

The /etc/hosts file contains at least loop-back and host information.
For example:

```
# cat /etc/hosts
# Internet host table
127.0.0.1      localhost      loghost
192.168.10.25 host1
```

The localhost and loghost are both assigned to the loop-back address and the interface name, host1, is assigned to a different IP address.

Important Files and Utilities

The following files and commands play a key role in the administration of the Solaris 8 Operating Environment.

The /etc/nodename File

Each Solaris Operating Environment has a host name, which is used by persons when referring to a system. You can change the host name by editing the `/etc/nodename` file and rebooting. The following is an example of a system's `/etc/nodename` file:

```
# cat /etc/nodename
host1
```

A system's host name and the name of its network interfaces do not need to be the same and are often different.

Determining the Current Network Configuration

Use the `ifconfig -a` command to display the settings of all configured interfaces; for example:

```
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ffffffff
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 192.168.10.25 netmask ffffffff broadcast 192.168.10.255
    ether 8:0:20:a2:11:de
hme1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 192.9.200.201 netmask ffffffff broadcast 192.9.200.255
    ether 8:0:20:a2:11:de
#
```

The `hme0` interface is up, running, and configured with `192.168.10.25` as its IP address.

You can also use the `ifconfig` utility to manually change the IP address of an interface. For example, to change the IP address to 192.168.10.37, execute the following commands:

```
# ifconfig hme0 down
# ifconfig hme0 192.168.10.37 up
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
    inet 127.0.0.1 netmask ffffffff
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 192.168.10.37 netmask fffffff0 broadcast 192.168.10.255
    ether 8:0:20:a2:11:de
hme1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
    inet 192.9.200.201 netmask fffffff0 broadcast 192.9.200.255
    ether 8:0:20:a2:11:de
```

Network Troubleshooting Utilities

Two of the most common network troubleshooting utilities are the packet internet groper (`ping`) and the `snoop` utility. Use the `ping` utility to determine if another system can be contacted over the TCP/IP network; for example:

```
# ping host2
host2 is alive
```

A response of `no answer` from `host2` indicates that `host2` is not available on the network.

Use the `snoop` utility to determine what information is actually traveling between systems. The `snoop` utility can show what actually happens when one system uses the `ping` utility to communication with another system. For example:

```
# snoop host1 host2
host1 -> host2 ICMP Echo request
host2 -> host1 ICMP Echo reply
```

The `snoop` utility can also use audible clicks to notify you of any network traffic by using the `-a` switch. Although noisy, this is especially useful when troubleshooting a JumpStart™ or Dynamic Host Configuration Protocol (DHCP) boot without the help of a second person in a large room.

For example, to hear audible clicks for all network traffic related to a DHCP boot, execute the following:

```
# snoop -a dhcp
```

Some additional snoop options include:

- `snoop -V`
Provides a summary verbose output
- `snoop -v`
Provides a detailed verbose output
- `snoop -o filename`
Redirects the snoop activity output to *filename*
- `snoop -i filename -V |more`
Displays packets that were previously captured in *filename*

Network Services

Each network service requires a server process to respond to a client request.

The Internet Service Daemon (inetd)

A special network process, `inetd`, runs on each system to listen on behalf of many server processes that are not started at boot time. The `inetd` process starts these server processes when the appropriate service is requested.

The `inetd` process is informed of the services to listen for and the corresponding processes to start through the `/etc/inet/inetd.conf` file. For example:

```
# grep ftp /etc/inet/inetd.conf
ftp      stream  tcp    nowait  root    /usr/sbin/in.ftpd    in.ftpd
```

If a change is made to the `/etc/inet/inetd.conf` file, a hang-up signal must be sent to the `inetd` process to force it to reread the configuration file. For example:

```
# pkill -HUP inetd
```

Port Numbers

Each network service uses a port that represents an address space, which is reserved for that service. A client usually communicates with a server through a well-known port. Well-known ports are listed in the `/etc/services` file; for example:

```
# grep telnet /etc/services
telnet      23/tcp
#
```

The example shows that the `telnet` service uses well-known Port 23 and uses the TCP protocol.

Remote Procedure Call (RPC)

Each network service must have a unique port number that is agreed upon by all hosts in the network. This is an increasingly difficult task given the number of systems on any network and the number of network services that the systems are capable of running.

Sun Microsystems™ developed an extension to the client-server model known as a remote procedure call (RPC). When using an RPC service, a client connects to a special server process, `rpcbind`, which is a well-known registered Internet service.

The `rpcbind` process registers port numbers associated with each RPC service listed in the `/etc/rpc` file. The `rpcbind` process receives all RPC-based client application connection requests and sends the client the appropriate server port number. For example, the `sprayd` entry is listed in the `/etc/rpc` file, and looks like the following:

```
# grep spray /etc/rpc
sprayd      100012  spray
#
```

This shows that the `sprayd` daemon has a program number of 100012 and is also known as `spray`.

Checking for Registered Services

Use the `rpcinfo` utility with the `-p` switch to list registered RPC programs. For example, to determine if the `sprayd` daemon is registered, execute the following:

```
# rpcinfo -p host1 | grep sprayd
100012    1    udp  32805  sprayd
#
```

Stopping a Network Service

Use the `rpcinfo` utility with the `-d` switch to unregister an RPC program, which effectively stops the service. For example, to stop the `spray` service, execute the following:

```
# rpcinfo -d sprayd 1
```

To verify the service has been stopped, execute the following:

```
# rpcinfo -p | grep sprayd
#
```

Starting a Network Service

You can register RPC network services by sending an HUP (Hangup) signal to the `inetd` process. For example, to start the `spray` service again, execute the following:

```
# pkill -HUP inetd
```

To verify the service has been registered again, execute the following:

```
# rpcinfo -p | grep sprayd
    100012    1    udp  42288  sprayd
#
```

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish the following:

- ☐ Define the function of each layer within the seven-layer OSI model and the five-layer TCP/IP model
- ☐ Describe the contents of various network control files
- ☐ Construct command strings to perform basic monitoring operations on an active network
- ☐ Start and stop network services using the command line

Objectives

Upon completion of this module, you should be able to:

- Configure syslog message routing
- Modify log message priority and severity
- Determine the effect of the LOGHOST variable on the syslog process
- Describe the two methods of starting the syslogd daemon
- Add entries to a system log using the logger utility

Additional Resources



Additional resources – The following references provide additional details on the topics discussed in this module:

- *System Administration Guide, Volume 2*, Part Number 805-7229-10
- *System Administration Guide, Volume 3*, Part Number 806-0916-10

The syslog Facility

The `syslog()` function sends messages generated by the kernel and system utilities to the `syslogd` daemon. Depending on the configuration of the `/etc/syslog.conf` file, this daemon can:

- Write messages to a system log
- Write messages to the system console
- Forward messages to a list of users
- Forward messages to the `syslogd` on other hosts over the network

The most valuable feature of `syslog` is that it puts you in control of message logging. This enables you to decide which messages are to be kept and where the messages are to be placed.

The syslog Concept

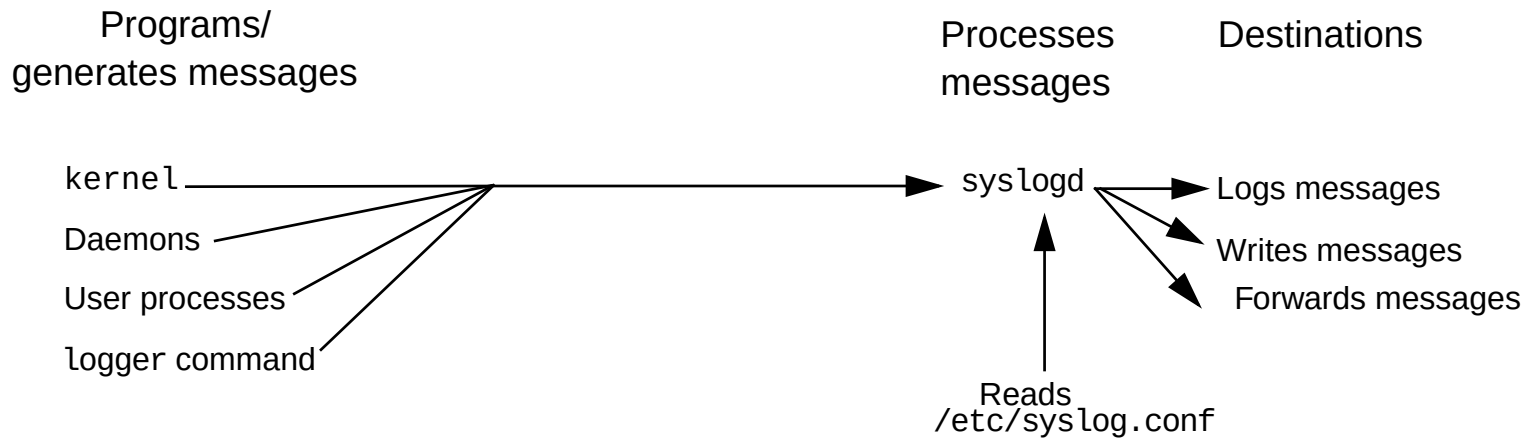


Figure 3-1 The syslog Concept

Controlling the Behavior of syslogd

Many processes are programmed to generate messages at various levels of importance in response to actions taken, or conditions encountered, during operation.

You can control the manner in which `syslogd` manages these messages by modifying the `/etc/syslog.conf` configuration file. From this configuration file, you can instruct `syslogd` to sort messages by their source or their importance and route them to a specified destination.

Configuring the `/etc/syslog.conf` File

A configuration entry in the `/etc/syslog.conf` file consists of two tab-separated fields: *selector* and *action*.

The selector field consists of a facility and a level written as *facility.level*. Facilities represent categories of system processes that can generate messages. Levels represent the severity or importance of the message.

The action field determines where to send the message.

For example, placing the following entry in the `/etc/syslog.conf` file causes error messages for all facilities to be sent to the `/var/adm/messages` file:

```
*.err                /var/adm/messages
```

where

```
*.err                Is the selector field; * is the facility, . is the  
                    delimiter, and err is the level of the message
```

```
/var/adm/messages    Is the action field
```



Caution – Only use *tabs* as white space in the `/etc/syslog.conf` file.

Selector Field

The selector field is a semicolon-separated list of priority specifications of the form:

facility.level; facility.level.

Facility is a system facility that is defined by the items shown in Table 3-1.

Table 3-1 Facility

kern	Messages generated by the kernel.
user	Messages generated by user processes. This is the default priority for messages from programs or facilities not listed in this file.
mail	The mail system.
daemon	System daemons, such as <code>in.ftpd</code> and <code>telnetd</code> .
auth	The authorization system including <code>login</code> , <code>su</code> , and <code>getty</code> .
syslog	Messages generated internally by <code>syslogd</code> .
lpr	The line printer spooling system – <code>lpr</code> and <code>lpc</code> .
news	Files reserved for the USENET network news system.
uucp	The UNIX-to-UNIX copy (UUCP) system; does not use <code>syslog</code> .
cron	The <code>cron</code> and <code>at</code> facilities, including <code>crontab</code> , <code>at</code> , and <code>cron</code> .
local0-7	A field reserved for local use.
mark	Time-stamp messages produced internally by <code>syslogd</code> .
*	All facilities, except the <code>mark</code> facility.

Note – You can use the `*` to select all facilities (for example `*.err`); however, you cannot use it to select all levels for a facility (for example, `kern.*`)

Level is the severity of the message. Levels in order of descending order of severity are shown in

Table 3-2 Levels

<code>emerg</code>	Panic conditions that are normally be broadcast to all users.
<code>alert</code>	Conditions that should be corrected immediately, such as a corrupted system database.
<code>crit</code>	Warnings about critical conditions, such as hard device errors.
<code>err</code>	Other errors.
<code>warning</code>	Warning messages.
<code>notice</code>	For conditions that are not error conditions, but might require special handling.
<code>info</code>	Informational messages.
<code>debug</code>	Messages that are normally used only when debugging a program.

The `none` message is normally used only when debugging a program. The `none` message appears when messages are not sent from the indicated facility to the selected file; for example, a selector of `*.debug;mail.none` sends all messages except mail messages to the selected file.

Note – Not all levels of severity are implemented for all facilities in the same way. For more information, refer to the online manual pages.

Action Field

The action field defines where the message should be forwarded. It can have any one of the following forms:

- */filename*

The absolute path for log file is required.

Note – This file must be manually created if it does not exist.

- *@host*

You must prefix the host name or IP address with an @ sign. Messages are forwarded to the `syslogd` of the remote system.

- *user1, user2*

`user1` and `user2` receive messages if they are logged in.

- ***

All logged-in users will receive messages.

The /etc/syslog.conf File

A sample /etc/syslog.conf configuration file is:

```
#ident  "@(#)syslog.conf  1.5   98/12/14 SMI"   /* SunOS 5.0 */
#
# Copyright (c) 1991-1998, by Sun Microsystems, Inc.
# All rights reserved
#
# syslog configuration file.
#
# This file is processed by m4 so be careful to quote (``) names
# that match m4 reserved words.  Also, within ifdef's, arguments
# containing commas must be quoted.
#
*.err;kern.notice;auth.notice           /dev/sysmsg
*.err;kern.debug;daemon.notice;mail.crit /var/adm/messages

*.alert;kern.err;daemon.err             operator
*.alert      root

*.emerg      *

# if a non-loghost machine chooses to have authentication messages
# sent to the loghost machine, un-comment out the following line:
#auth.notice      ifdef('LOGHOST', /var/log/authlog, @loghost)

mail.debug      ifdef('LOGHOST', /var/log/authlog, @loghost)

#
# non-loghost machines will use the following lines to cause "user"
# log messages to be logged locally.
#
ifdef      ('LOGHOST', ,
user.err      /dev/sysmsg
user.err      /var/adm/messages
user.alert    'root, operator'
user.emerg    *
)
```

Starting and Stopping syslogd

The configuration file is read each time `syslogd` starts. The `/etc/rc2.d/S74syslog` file starts `syslogd` during each system boot.

You can manually start or stop `syslogd`, if the configuration file has been modified, with the command:

```
# /etc/init.d/syslog start | stop
```

syslogd and the m4 Macro Processor

The syslogd daemon, the m4 macro processor, and the /etc/syslog.conf file interact, in conceptual phases, to determine correct message routing. These conceptual phases are described as:

1. syslogd runs m4.
2. m4 processes ifdef statements in /etc/syslog.conf.
3. syslogd uses m4 output to route messages to the appropriate places.

On first evaluation, it appears the syslogd daemon receives message-log routing information from the /etc/syslog.conf file. However, syslogd does not read the /etc/syslog.conf file directly. Instead, syslogd starts m4, which parses the /etc/syslog.conf file for ifdef statements that can be interpreted by m4.

If m4 does not recognize any m4 commands on a line, it passes the output back to syslogd as a two-column output that syslogd then uses to route messages to appropriate destinations. If m4 encounters an ifdef statement within the /etc/syslog.conf file, the ifdef is evaluated for a true or false condition, and message routing occurs relative to the output of the test.

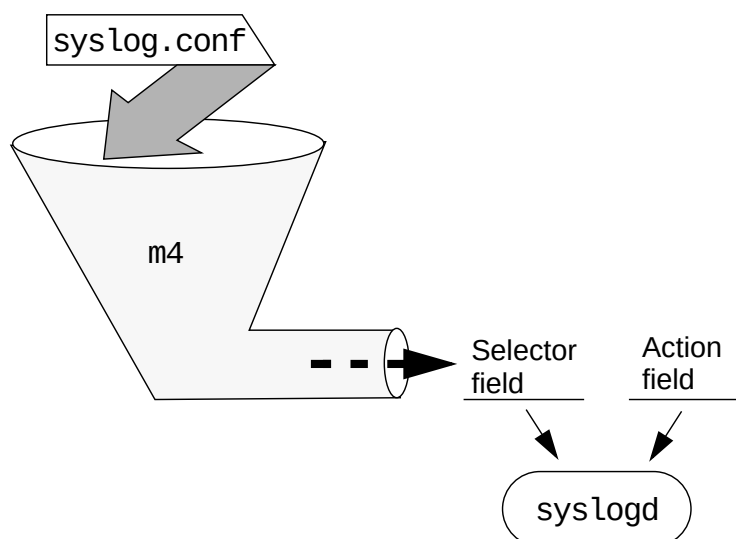


Figure 3-2 The m4 Macro Processor

Detailed Operation

The following examples identify the `loghost` in the `/etc/hosts` file.

Note – These `/etc/hosts` file examples have been excerpted for brevity.

Example A

```
192.9.200.1 host1 loghost
192.9.200.2 host2
```

Example B

```
192.9.200.1 host1
192.9.200.2 host2 loghost
```

Phase 1

When `syslogd` starts on boot, `syslogd` evaluates the `/etc/hosts` file to check the IP address associated with the *hostname* compared to the IP address associated with the `loghost`.

In Example A, `host1` and `loghost` are both associated with IP address `192.9.200.1`; therefore, `syslogd` runs the second command line, `/usr/ccs/bin/m4 -D LOGHOST` that causes the `m4 LOGHOST` variable to be evaluated as `TRUE` during the parsing of the `/etc/syslog.conf` file.

In Example B, `host1` is associated with IP address `192.9.200.1`, while `host2` and `loghost` are both associated with IP address `192.9.200.2`; therefore, `syslogd` runs the first command line, `/usr/ccs/bin/m4` (no `-D LOGHOST`) that causes the `m4 LOGHOST` variable to be evaluated as `FALSE` during the parsing of the `/etc/syslog.conf` file.

Phase 2

In the second phase, the `m4` macro processor parses the `/etc/syslog.conf` file. For each uncommented line that is parsed, `m4` searches the line for an `ifdef` statement. If no `ifdef` is encountered on the line, `m4` passes the line back to `syslogd` daemon.

If the `m4` finds a line with an `ifdef` statement, the line is evaluated for the `TRUE` or `FALSE` condition of the `LOGHOST` variable, and `m4` passes `syslogd` the output, accordingly. For example,

```
mail.debug          ifdef('LOGHOST', /var/log/authlog, @loghost)
```

Consider, if the `LOGHOST` variable was evaluated as `TRUE` in Phase 1, then the `m4` processor returns:

```
mail.debug          /var/log/authlog
```

If the `LOGHOST` variable was evaluated as `FALSE` in Phase 1, then the `m4` processor returns:

```
mail.debug          @loghost
```

In either case, the output has an entry in the selector field and an entry in the action field.

Phase 3

In phase 2, for each line that was parsed in the `/etc/syslog.conf` file, `m4` produced output in a two-column field: A selector field and an action field. This information is returned to `syslogd`, and `syslogd` uses the information to route messages to their appropriate destinations.

Once configured, `syslogd` continues to run with this configuration.

Modifying inetd to Use syslog

The `inetd` is the server process for many network services. The `inetd` process listens for service requests on the TCP (or UDP) ports associated with each of the service listed in its configuration file. When a request arrives, `inetd` executes the server program associated with the service. You can modify the `inetd` to log TCP connections using the `syslogd`.

inetd Manual Page Excerpt

The following online manual page excerpt for `inetd` shows that only the `daemon` facility and the `notice` message level is supported:

```
% man inetd
Maintenance Commands
inetd(1M)

NAME
    inetd - Internet services daemon
...
...
-t          Instructs inetd to trace the incoming
connections for all of its TCP services. It does this by
logging the client's IP address and TCP port number,
along with the name of the service, using the syslog(3)
facility. UDP services can not be traced. When tracing is
enabled, inetd uses the syslog facility code ``daemon''
and ``notice'' priority level.
...
```

Note – The Internet daemon, `inetd`, provides services for many network protocols including the `telnet` protocol and File Transfer Protocol (FTP).

The inetd Startup File

Using the `-t` option as an argument to the `inetd` command enables TCP tracing. You must enable the trace option for the `inetd` daemon for `syslog` messaging. You add the `-t` option to the entry, which starts `inetd` in the `inetsvc` script in the `/etc/init.d` directory.

The modified entry looks similar to the following:

```
# grep inetd /etc/init.d/inetsvc
/usr/sbin/inetd -s -t &
#
```

Note – You must restart the `inetd` process for the new option to take effect.

The `/etc/syslog.conf` file configures the `syslogd` to selectively distribute the messages sent to it; in the previous example, from `inetd`.

```
# grep daemon.notice /etc/syslog.conf
*.err;kern.debug;daemon.notice;mail.crit      /var/adm/messages
```

The `notice` entry in the `/etc/syslog.conf` file causes all `daemon` messages of level `notice` to be sent to the `/var/adm/messages` file.

Note – The `/var/adm/messages` file must exist and you must stop and start the `syslog` daemon.

Example of syslog Logged Entry

You can monitor the `syslog` file, `/var/adm/messages`, in real time using the command `tail -f`. This holds the file open so you view messages being routed into this file by `syslog`.

```
# tail -f /var/adm/messages
```

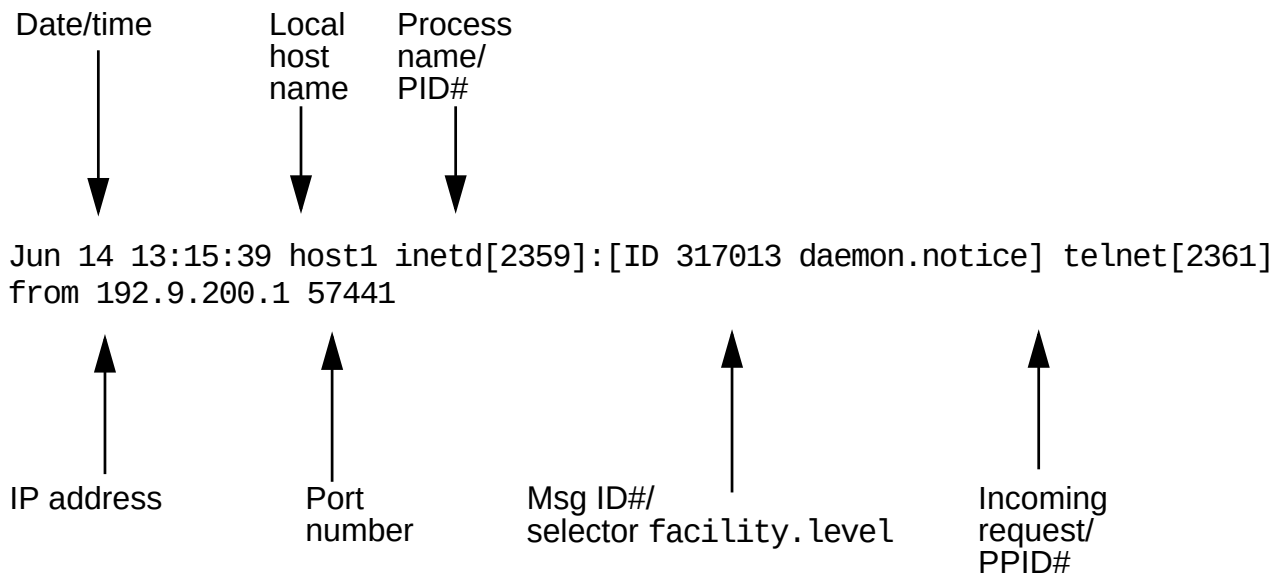


Figure 3-3 Example of syslog Logged Entry

The preceding output logs a `telnet` request to system `host1` from IP address `192.9.200.1` on port `45800`.

To exit, press Control-C.

Note – You can use scripts to automatically parse the log files and send notification to support personnel should any unusual activity exist.

The logger Utility

With the `logger` command, you can add one-line entries to a system log file. Typically, you can the `logger` command as part of a script.

Command Format

```
logger [ -i ] [ -f file ] [ -p priority ] [ -t tag ] [ message ]
```

Command Options

- `-f file`
Uses the contents of *file* as the message to log (*file* must exist).
- `-i`
Logs the process ID of the logger process with each line.
- `-p priority`
Enters the message with the specified priority.
- `-t tag`
Marks each line added to the log with the specified tag.
- `message`
Concatenates the string arguments in the message together, in the order specified, separated by single-space characters.

Examples

The following example logs the `System rebooted` message to the default priority level `notice` and the facility `user` for `syslogd`.

```
# logger System rebooted
```

The `System rebooted` message *should* be logged to the file designated for the `user.notice` selector field. However, if you investigate further, you will find that the `user.notice` selector field is not configured (by default) in the `/etc/syslog.conf` file. You can either add the `user.notice` selector field to the `/etc/syslog.conf` file, or you can prioritize the output as follows:

```
# logger -p user.err System rebooted
```

Changing the priority of the message to `user.err` will route the message to the `/var/adm/messages` file as indicated in the `/etc/syslog.conf` file.

Exercise: Using `syslog` and Auditing Utilities



Exercise objective – In this lab, you configure and use the `syslog` utility to better monitor your system.

Preparation

Ensure that your system boots without errors and that you can log in as `root`.

Task Summary

In this exercise, you accomplish the following:

- Configure `syslog` logging for the `login` and `telnet` daemons
 - ▼ Use the `syslog` utility to write logs to the `/var/adm/messages` file
 - ▼ Configure `syslog` to log the `auth` and `daemon` facilities
 - ▼ Use the `notice` selector level
- Use the `tail` command to monitor the `syslog` log in real time
- Use the `telnet` command to test logging

Tasks

To configure and use the `syslog` utility to better monitor your system, perform the following steps:

1. Configure `syslog` logging for the `login` and `telnet` daemons.
 - a. Make a backup file of the original `/etc/syslog.conf` file.
`cp /etc/syslog.conf /etc/syslog.conf.orig`

- b. Edit the `/etc/syslog.conf` file and add the selector field `auth.notice` to the second entry, which should look like the following:

```
*.err;kern.debug;daemon.notice;mail.crit;auth.notice    /var/adm/messages
```

- c. Save your changes, and quit the editor.
- d. Edit the `/etc/init.d/inetsvc` file and change the line for the `inetd` command to include the `-t` option.

```
/usr/sbin/inetd -s -t &
```

- e. Save your changes, and quit the editor.
- f. Stop and start the `syslogd` process.

```
# /etc/init.d/syslog stop  
# /etc/init.d/syslog start
```

- g. Stop and start the `inetd` process.

```
# /etc/init.d/inetsvc stop  
# /etc/init.d/inetsvc start
```

2. Use the `tail` command in a terminal window to monitor the `syslog` log in real time.

```
# tail -f /var/adm/messages
```

3. In another terminal window, test logging by using the `telnet` command to log in to your own system.

```
# telnet your_hostname
```

Notice how the `syslog` entry is updated as reported by the `tail` command.

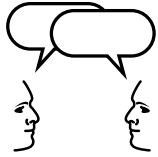
4. Exit the `telnet` session, and observe the `syslog` entry.

```
# exit  
Connection closed by foreign host.
```

Note – Nothing is logged when you exit the `telnet` session.

5. Press Control-C to stop the output of the `tail` command running in the other window.

Exercise Summary



Discussion – Take a few minutes to discuss the experiences, issues, or discoveries that you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish the following:

- ☐ Configure `syslog` message routing
- ☐ Modify log message priority and severity
- ☐ Determine the effect of the `LOGHOST` variable on the `syslog` process
- ☐ Describe the two methods of starting the `syslogd` daemon
- ☐ Add entries to a system log using the `logger` utility

Objectives

Upon completion of this module, you should be able to:

- List the three utilities used to create, check, and mount file systems
- Identify the physical path name differences between physical disks and virtual disks
- List the potential advantages of any virtual disk management application
- List the basic difference between Solstice DiskSuite™ and Sun StorEdge Volume Manager™
- List the main advantages of using a concatenated virtual file system
- List the main advantage of using a striped virtual file system
- Install the Solstice DiskSuite applications
- Use the Solstice DiskSuite application to dynamically grow a file system

Additional Resources



Additional resources – The following references provide additional details on the topics discussed in this module:

- *System Administration Guide, Volume I*, Part Number 805-7228
- *System Administration Guide, Volume II*, Part Number 805-7229
- *Solstice DiskSuite 4.2.1 Reference Guide*, Part Number 806-3204-10
- *Solstice DiskSuite 4.2 User's Guide*, Part Number 806-3205-10
- *Sun Enterprise Volume Manager 2.5 Administration Guide*, Part Number 805-1607

Physical Disks

In a standard Solaris 8 Operating Environment installation, memory-resident drivers access all physical disks. Each type of disk device has a unique driver.

Typical Physical Disk Drivers

Typical physical disk drivers include:

- dad – IDE disk driver
- sd — The SCSI disk drive driver

For efficiency, most drivers are loaded into memory at system boot time.

Access Paths

The access path to all physical disks is through path names defined in the `/dev` directory. For every slice on every physical disk, there are two unique access paths—the block device path and the raw device path.

Block Device Path

The block device path is used by commands (such as `mount`), utilities, and processes that refer to the slice as a file system. For example, the following are typical block device path names:

- `/dev/dsk/c0t0d0s0`
- `/dev/dsk/c0t0d0s7`

The following is a typical `mount` command using the block device path name:

```
# mount /dev/dsk/c0t0d0s7 /mnt
```

Raw Device Path

The raw device access path is used by utilities and processes that do not use the device as a file system but transfer data sector by sector. For example, the following are typical raw device path names:

- `/dev/rdisk/c0t0d0s0`
- `/dev/rdisk/c0t0d0s7`

The following are typical commands that can be used with the raw device path name:

```
# newfs /dev/rdisk/c0t0d0s7
# fsck /dev/rdisk/c0t0d0s7
```

Virtual Disk Access Paths

The key feature of all virtual volume management applications is that they transparently provide a virtual partition that can consist of many disk partitions. In the Solaris Operating Environment, the virtual partition appears to be the same as any other. The logical device names are similar to special devices in that they have both a raw device path and a block device path.

The following are typical virtual volume raw and block device path names for disks created with Solstice DiskSuite:

- `/dev/md/rdisk/d42`
- `/dev/md/dsk/d42`

The following are typical virtual volume raw and block device path names for disks created with Sun StorEdge Volume Manager:

- `/dev/vx/rdisk/apps/logvol`
- `/dev/vx/dsk/apps/logvol`

You can use virtual volume device paths the same way as any other device path by system utilities; for example:

```
# mount /dev/md/dsk/d42 /mnt
# newfs /dev/md/rdisk/d42
# fsck /dev/vx/rdisk/apps/logvol
```



To eliminate the limitation of one slice per file system, there are virtual volume management applications that can create virtual volume structures in which a single file system can consist of almost an unlimited number of disks or slices.

Two virtual volume managers are available through Sun:

- Solstice DiskSuite
- Sun StorEdge Volume Manager

Virtual Volume Management

Solstice DiskSuite and Sun StorEdge Volume Manager assemble large volumes from multiple disk drives, but they use different approaches.

Solstice DiskSuite

Solstice DiskSuite uses standard partitioned disk slices that have been created using the `format` utility. A typical volume structure is assembled and managed transparently.



Figure 4-1 Solstice DiskSuite Management of Disk Slices

Sun StorEdge Volume Manager

Sun StorEdge Volume Manager manages disk space by using contiguous sectors. The application formats the disks into only two slices, Slice 3 and Slice 4. Slice 3 is called a private area, and Slice 4 is a public area.

Slice 3 maintains information about the virtual to physical device mappings, while the sectors in Slice 4 provide space to build the virtual devices. Contiguous sector groups can be configured into subdisks; see Figure 4-2 on page 4-7.

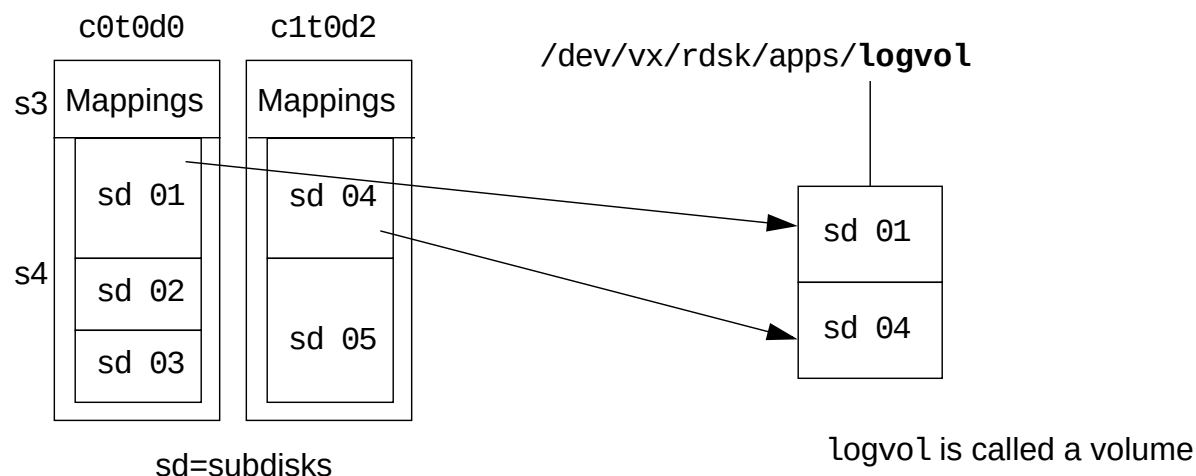


Figure 4-2 Sun StorEdge Volume Management of Disk Slices

An advantage of this approach is there is almost no limit to the number of subdisks you can create on a single disk drive. In a partitioned environment, there is an eight-partition limit.

Concatenated Volumes

A *concatenated volume* combines portions of one or more physical disks into a single virtual structure. The portions are contiguous, and the first portion tends to fill with data before the next portion is used. Figure 4-3 illustrates a sample layout of concatenated volumes.

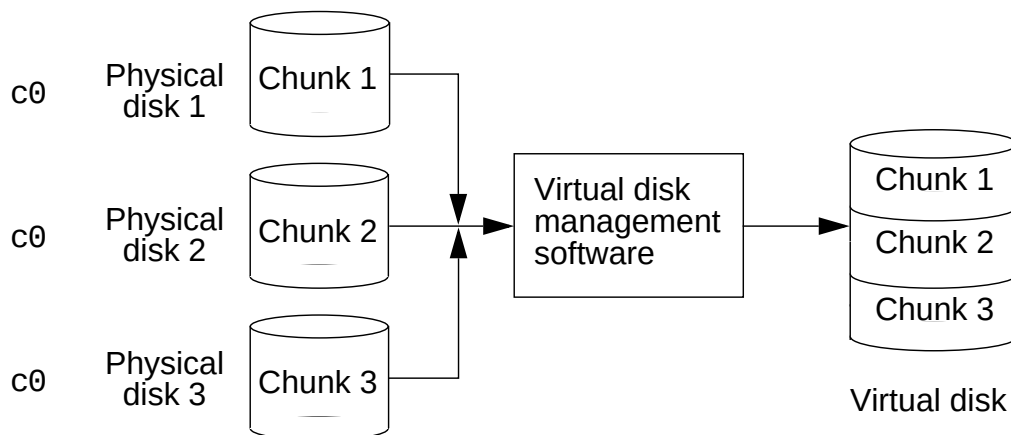


Figure 4-3 Concatenated Volumes

The following describes some of the features of a concatenated volume:

- It creates a virtual volume that is larger than one physical disk.
- You can grow a volume (file system) as needed by concatenating an additional physical disk to it. That is, you can increase the size of a file system while it is mounted and in use.

It is not uncommon for file systems to run out of space due to unexpected company expansion or poor planning. As a system administrator, you might need to increase the size of a file system. The Solstice DiskSuite package is bundled with the Solaris 8 Operating Environment server release. One feature of Solstice DiskSuite is that it can expand the size of a file system using concatenation.

Adding a Disk

Before you can use the disk management tools to configure additional disk space, you must first add the additional device and then modify the device configuration directories to make the device visible to the system. The two methods of making the device visible are:

- A reconfiguration boot
- An execution of the `devfsadm` daemon

Reconfiguration Boot

Traditionally, you would perform a reconfiguration boot operation to recognize new devices on the system. The three basic methods are:

- Execute a `boot: -r` from the boot PROM's ok prompt
- Execute a `reboot: -r` from the `#` (system's superuser shell) prompt
- Create a `/reconfigure` file and reboot the system

The disadvantage to these methods is that each requires you to reboot the system. In today's computer environments, many systems have a 24 hours a day, seven days a week (24x7) uptime requirement; therefore, rebooting the system to add new devices is not an option.

The devfsadm Daemon

For systems that have a 24x7 uptime requirement, you can add new devices without requiring a reboot.

Before the Solaris 8 Operating Environment, you needed a suite of `devfs` administration tools, including `drvconfig(1M)`, `disks(1M)`, `tapes(1M)`, `ports(1M)`, `audlinks(1M)`, and `devlinks(1M)` to create the `/dev` and `/devices` entries necessary for the Solaris Operating Environment to access new devices.

These commands still exist in the Solaris 8 Operating Environment; however, each of them is linked to the new `devfsadm` administration command that maintains the name space for `/dev` and `/devices` entries.

Therefore, after adding new hardware (or hot-plugable hardware where permitted), the `devfsadm` command is executed, thereby transparently building the necessary configuration entries. The new device is then ready for assignment by the system.

Installing the Solstice DiskSuite Software

The following steps describe how to install the Solstice DiskSuite application software to manage the disk partitions of multiple disks and multiple controllers:

1. Insert the Solaris 8 Software (2 of 2) CD-ROM (Solstice DiskSuite is contained in the Early Access [EA] folder on this CD-ROM), and change to the correct directory.

```
# cd /cdrom/cdrom0/Solaris_8/EA/products/DiskSuite_4.2.1
```

2. Start the installer program.

```
# ./installer
```

With the exception of the installer's actual GUI windows themselves, feedback generated from the `installer` program is displayed in the terminal window from which the program was started.

The Web Start Additional Information window (Figure 4-5) is displayed.



Figure 4-4 Web Start Additional Information Window

3. Click Next.

The Web Start Installer Welcome window (Figure 4-5) is displayed.



Figure 4-5 Web Start Installer Welcome Window

4. Click Next.

The Select Type of Install window (Figure 4-6) is displayed.

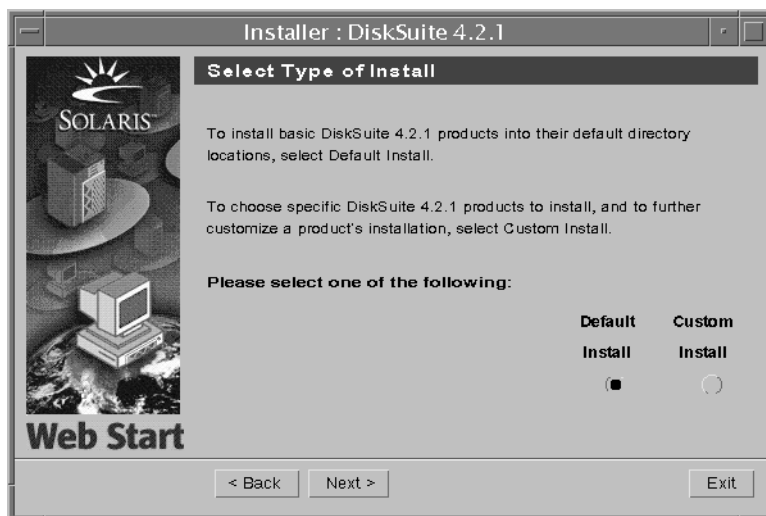


Figure 4-6 Select Type of Install Default Window

To see how the installer processes the Default Install, leave it selected and click Next.

The installer tests for available disk space and displays the Ready to Install window with a scrolling list of all items from the directory on the CD-ROM.

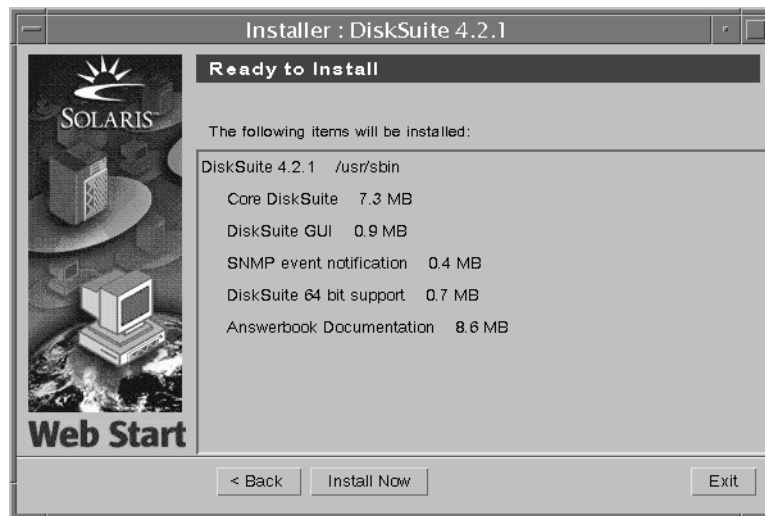


Figure 4-7 Ready to Install Window

5. To enable custom selection of each component listed, click Back to return to the Select Type of Install window.

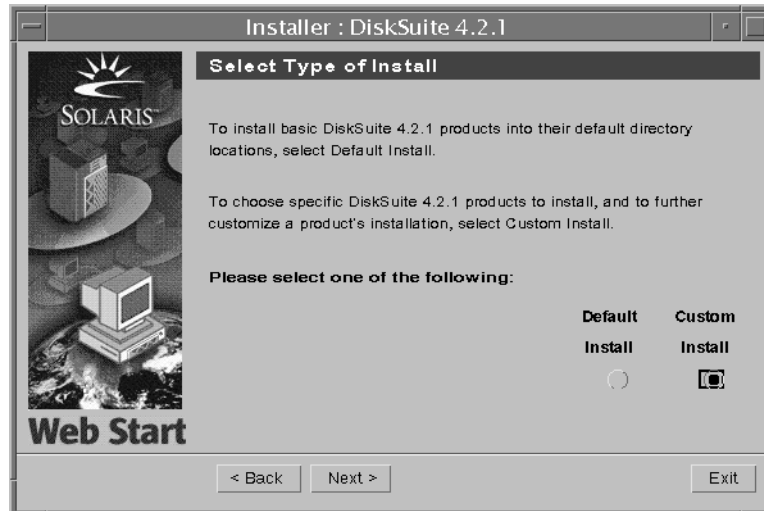


Figure 4-8 Select Type of Install Window

6. Click Custom Install
7. Click Next.

The Component Selection window (Figure 4-9) is displayed. All products available for installation are listed. Typically, you would leave all items selected, however, this window enables selection of each component.



Figure 4-9 Component Selection Window

8. Click Next.

The Ready to Install window (Figure 4-10) is displayed. The Solstice DiskSuite 4.2 products should be the only product listed for installation. If others are displayed, click Back to deselect them.



Figure 4-10 Ready to Install Window

9. Click Install Now.

The Installing window (Figure 4-11) is displayed to enable you to monitor progress. The bar reflects the progress of the current product being installed, and the bottom bar monitors the progress of the set of products being installed.



Figure 4-11 Installing Window

Refer also to the terminal window in which you originally started the installer program. It displays the following message:

```
Installing DiskSuite 4.2.1  
Log file:/var/sadm/install/logs/DiskSuite_4.2.1_install.A0555045
```

The final number following "install." is unique to each installation.

When the installation is complete, the Installation Summary window (Figure 4-12) is displayed, confirming success.

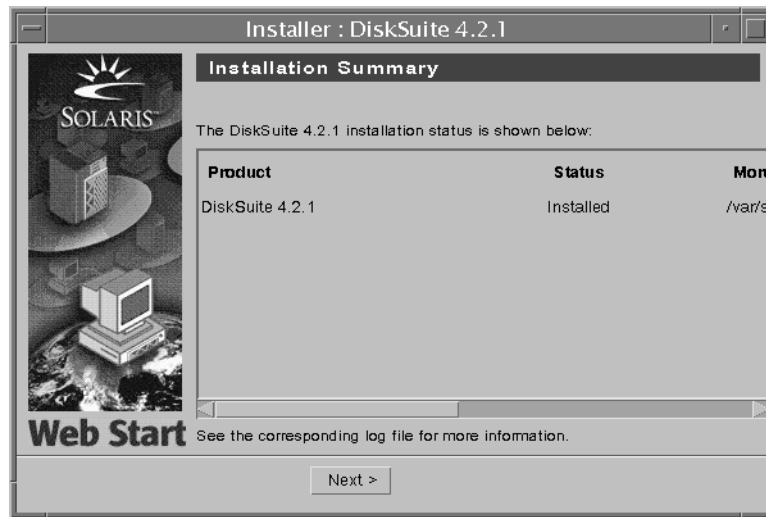


Figure 4-12 Installation Summary Window

10. Click Next.

The Additional Information window (Figure 4-13) is displayed, providing you with information on how to add or remove products using the Solaris Product Registry, `/usr/bin/prodreg`, or the Solaris Management Console.

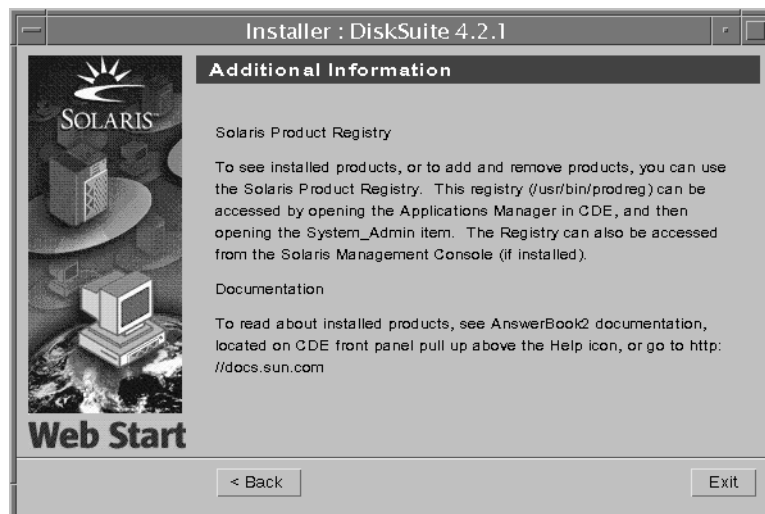


Figure 4-13 Additional Information Window

11. Click Exit.

A dialog box is displayed (Figure 4-14) offering you the selection of Reboot.

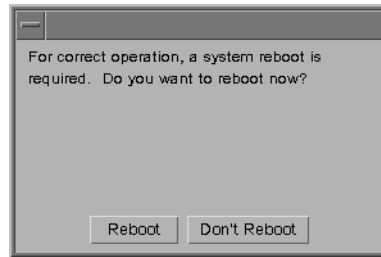


Figure 4-14 Reboot Dialog Box

12. Click Reboot.

Solaris Product Registry

The `prodreg` command (a Java program) referred to previously in Figure 4-13 on page 4-16, is the viewer for the Solaris Product Registry (ProdReg), a system for maintaining records of the software products installed on a given Solaris Operating Environment. You can use it to install and uninstall software packages.

To start the Product Registry program, run the `/usr/bin/prodreg` command:

```
# /usr/bin/prodreg
```

The viewer is shown in Figure 4-15.

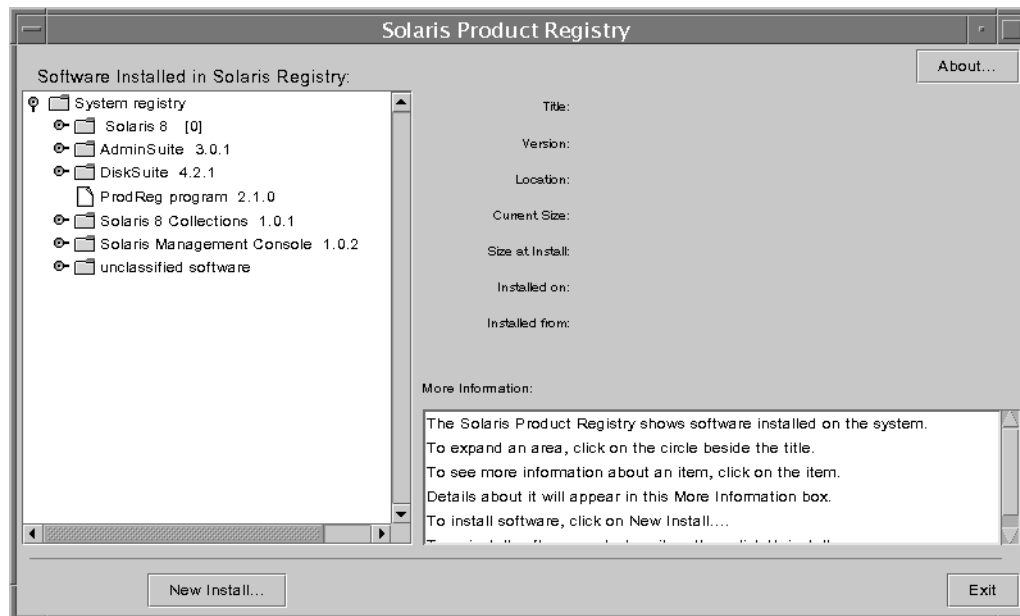


Figure 4-15 Solaris Product Registry Window

Click DiskSuite to view more information about the application.

The DiskSuite application information is displayed in the window (Figure 4-16) and offers an Uninstall DiskSuite button at the bottom. Using that button is the appropriate way to uninstall the DiskSuite software; this method of removal is true of any application installed with the installer.

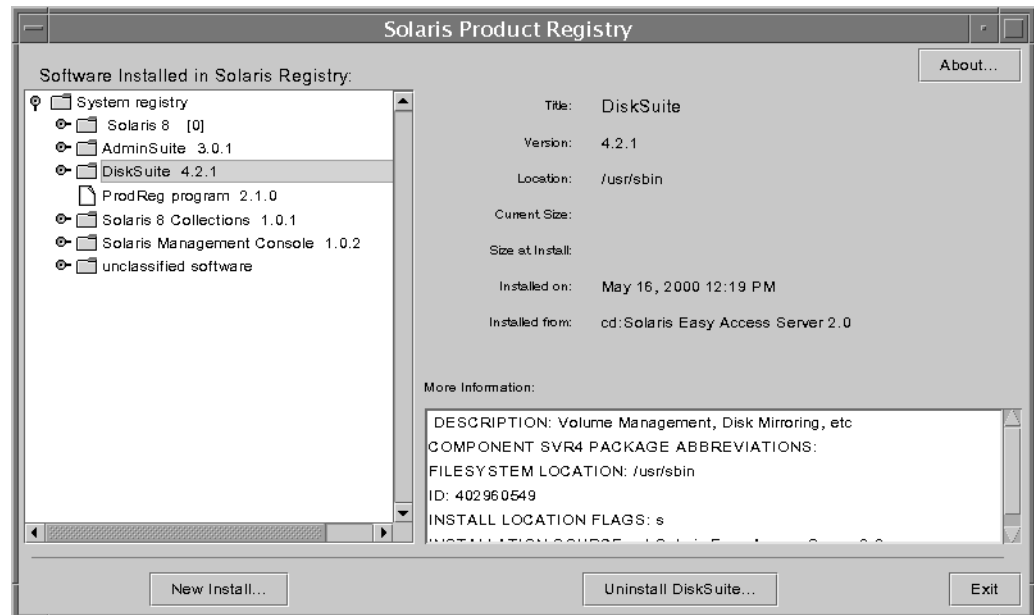


Figure 4-16 Solaris Product Registry Window With DiskSuite Information

If you click Uninstall DiskSuite, the Solaris Product Registry program removes the application and modifies necessary system files to reflect removal.

Starting the DiskSuite Tool

To start and use the DiskSuite tool, perform the following steps:

✓ *These steps assume you have available unused partitions for the following exercise.*

1. After the system presents the login screen, log in as root.
2. Make a backup of the /etc/vfstab file.
3. Use the command-line to invoke the Solstice DiskSuite Tool as follows:

```
# /usr/sbin/metatool &
```

Note – Running the DiskSuite Tool in the background frees the window for subsequent commands.

The first time you run the DiskSuite software after installation, the Metastate Database Warning window is displayed, as shown in Figure 4-17.

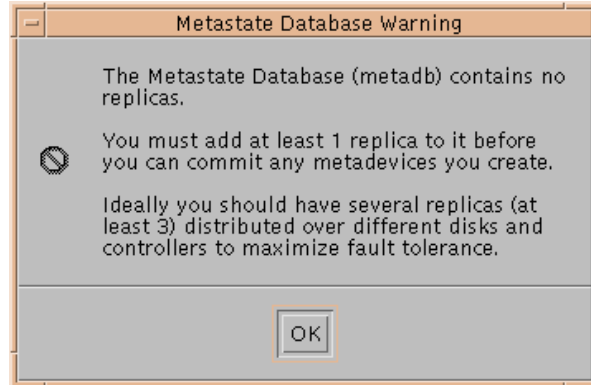


Figure 4-17 Metastate Database Warning Window

You must create the metastate database before you can manage disks.

4. Click OK.

The DiskSuite Tool – Metadevice Editor window (Figure 4-18) is displayed.

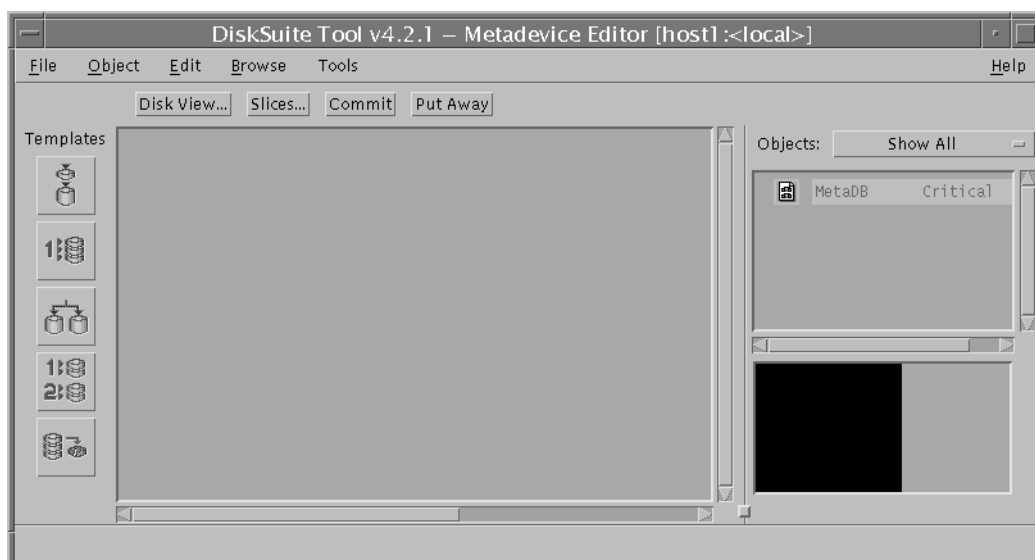


Figure 4-18 DiskSuite Tool – Metadevice Editor Window

5. Click Disk View to display the DiskSuite Tool – Disk View window (Figure 4-19).

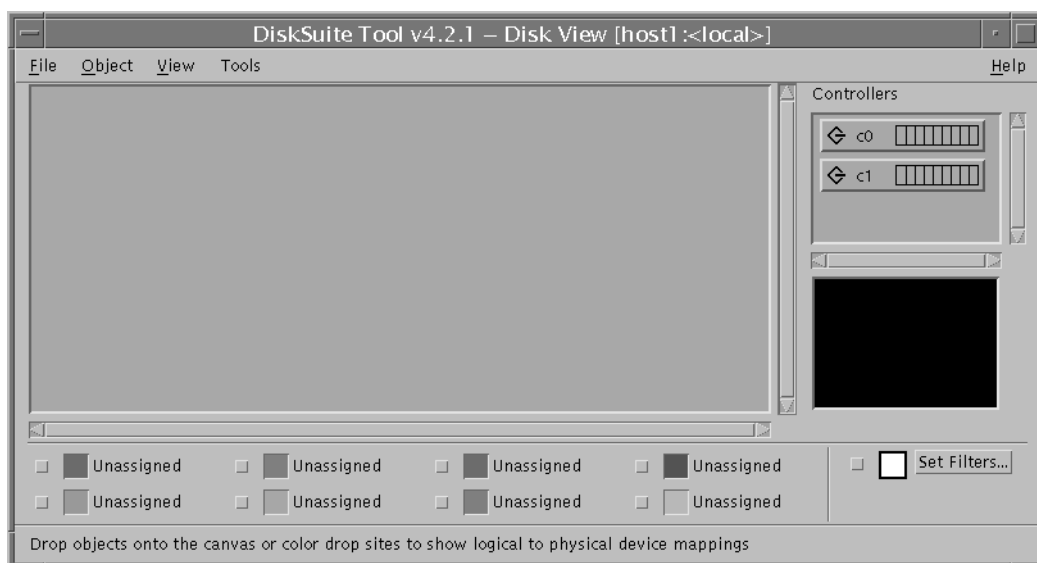


Figure 4-19 DiskSuite Tool – Disk View Window

6. Select View All Controllers from the View menu.

The DiskSuite Tool – Disk View window is displayed, as shown in Figure 4-20. This example indicates there is one disk attached to each of two controllers (c0 and c1) on this system.



Figure 4-20 DiskSuite Tool – Disk View Window Viewing All Controllers

7. Return to the DiskSuite Tool – Metadevice Editor window and click Slices.

As illustrated in Figure 4-21, the window lists information about each disk partition known to the system. (You might need to enlarge the window to see all of the information about a slice.)

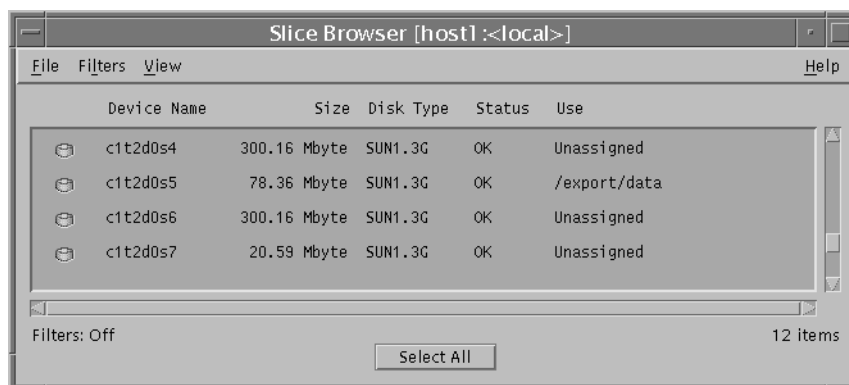


Figure 4-21 Slice Browser Window

Creating the State-Database Replicas

State-database replicas are required by Solstice DiskSuite to maintain disk configuration information. The following steps enable you to create this initial database:

1. Double-click the small MetaDB icon in the Objects area of the DiskSuite Tool to create the initial state-database replica (MetaDB).

The MetaDB icon moves to the large area of the DiskSuite Tool Metadevice Editor called the canvas. This is the active area of the tool where management of disks takes place.

The status of this object is Critical .

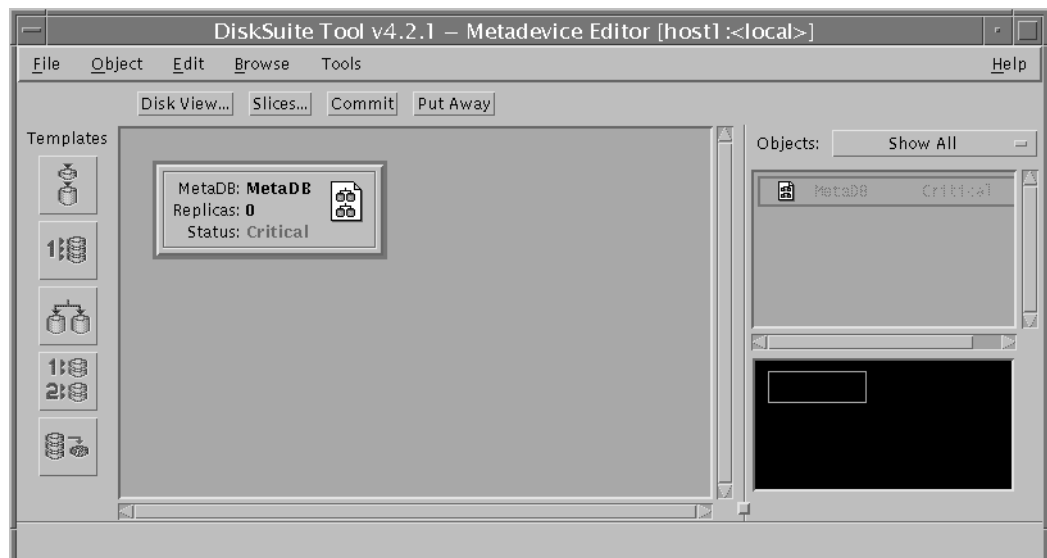


Figure 4-22 DiskSuite Tool – Metadevice Editor Window With MetaDB Icon Information

2. Use the MENU (right) mouse button to click the MetaDB object to display its menu. Select Info item.

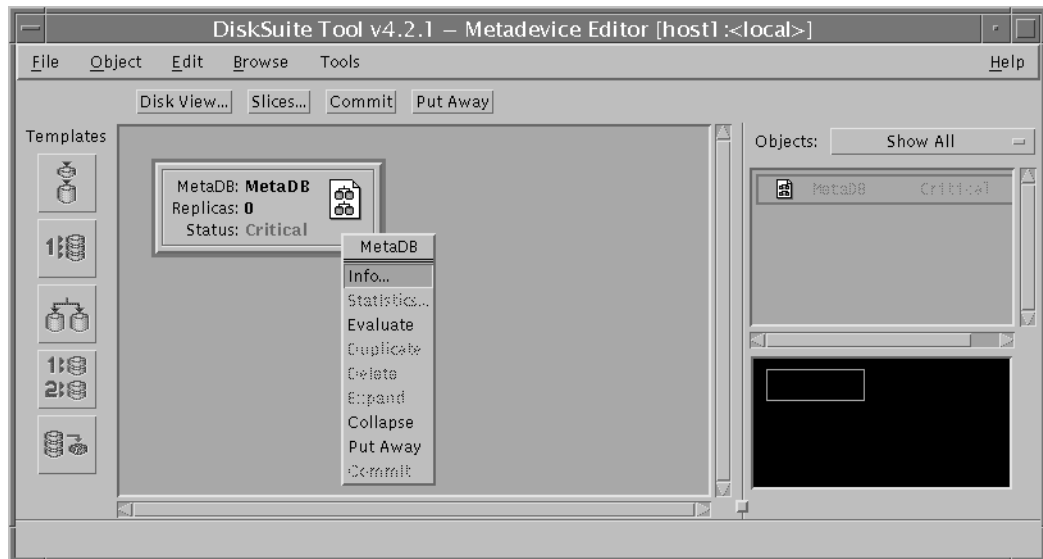


Figure 4-23 DiskSuite Tool – Metadevice Editor Window With MetaDB Information Menu

The Information window (Figure 4-22) is displayed.

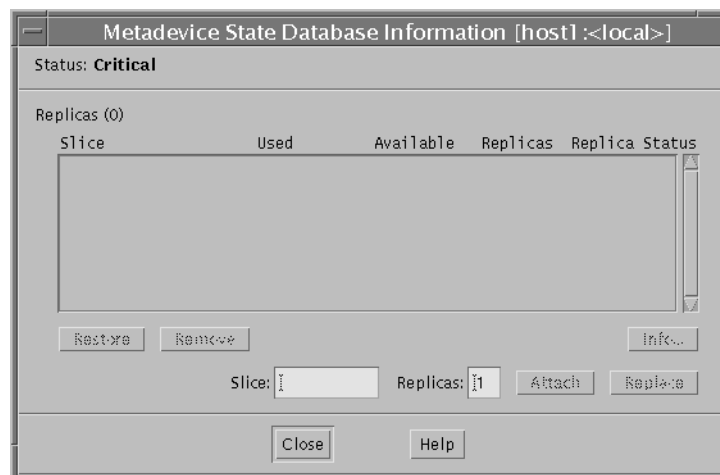


Figure 4-24 Metadevice State Database Information Window

3. In the field labeled Slice, enter the name of a small (approximately 5 or more Mbytes) slice; for example, c1t2d0s0.

4. In the field labeled Replicas, replace the initial value of 1 with 3.

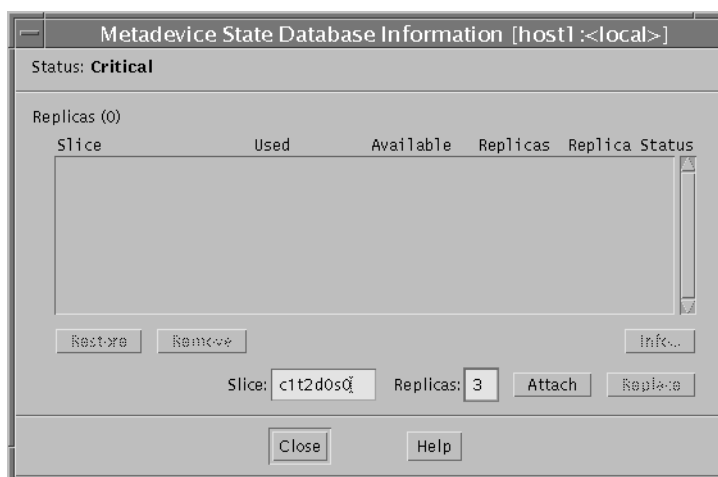


Figure 4-25 Metadevice State Database Information Window

5. Click Attach.

6. Click Close.

The Information window closes.

7. Return to the DiskSuite Tool Metadevice Editor and verify that the MetaDB object is selected in the canvas area (Figure 4-26), and click Commit.

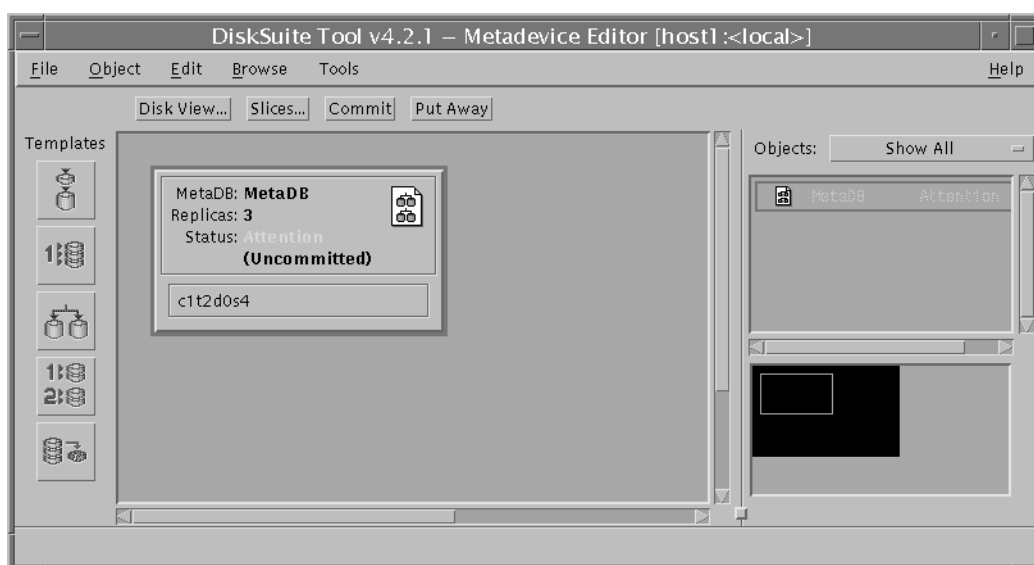


Figure 4-26 DiskSuite Tool – Metadevice Editor Window

A warning message is displayed that indicates that all of the replicas you are making are on the same controller.



Figure 4-27 Metastate Database Commit Warning Window

On a production system you would avoid doing this; however, you can accept this situation for the purposes of this lab.

8. Click Really Commit.

The MetaDB object status now displays Attention in yellow.

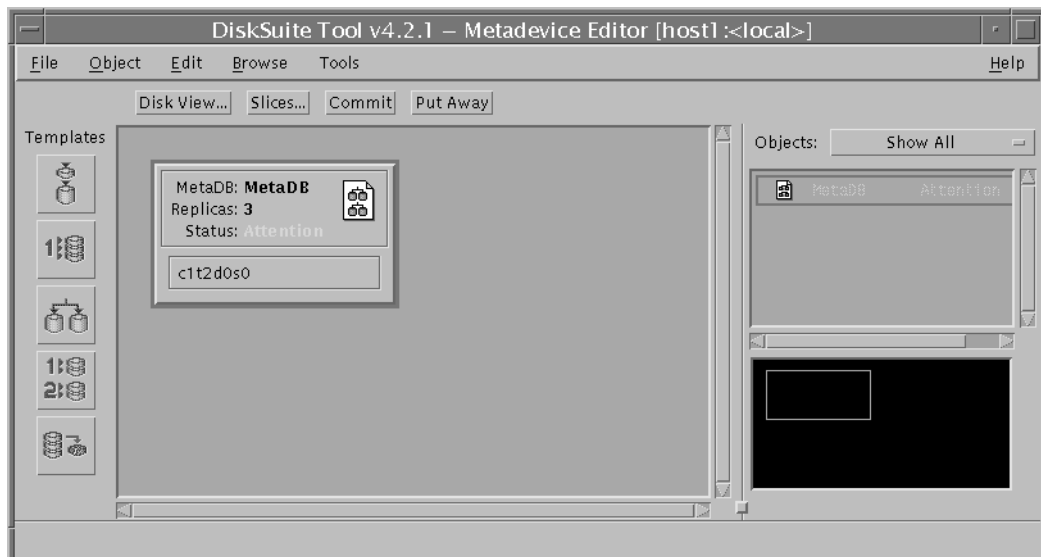


Figure 4-28 DiskSuite Tool – Metadevice Editor Window With Committed Metadevice

9. Click Put Away to remove the MetaDB object from the canvas.

The MetaDB icon moves out of the work area to the Objects list.

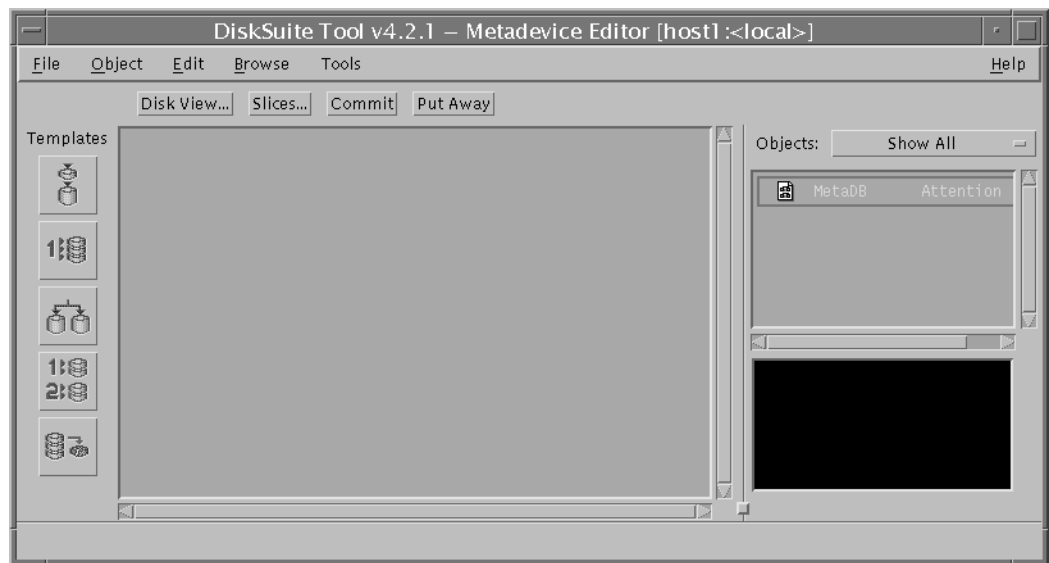


Figure 4-29 DiskSuite Tool – Metadevice Editor Window with Committed Metadevice in Objects Area

Concatenating File Systems

If your `/export/data` file system is filling up and you cannot afford to move the entire file system, using the Solstice DiskSuite application you can concatenate available space from other disk partitions to increase the available size of the `/export/data` file system.

To increase the size of the `/export/data` file system, perform the following steps:

1. Determine the current space available in the `/export/data` file system.

```
# df -k /export/data
```

Filesystem	kbytes	used	avail	capacity	Mounted on
/dev/dsk/c1t2d0s5	74975	65273	2205	97%	/export/data

The output of the preceding command indicates that the `/export/data` file system, which is approximately 75 Mbytes in size, is almost full.

- Click the Concatenate/Stripe template.

A Concatenate/Stripe icon is displayed in the canvas of the DiskSuite Tool Metadevice Editor (Figure 4-30).

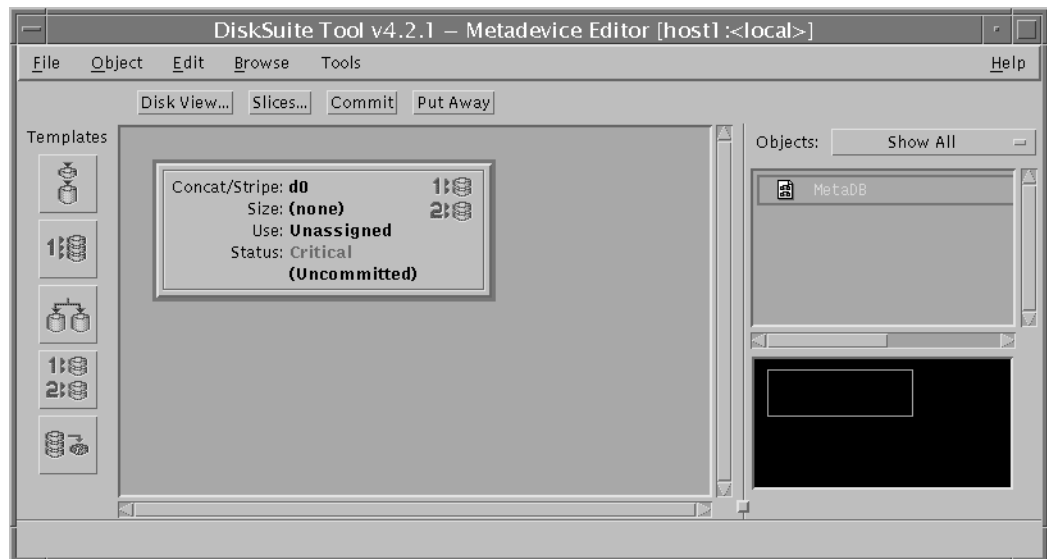


Figure 4-30 DiskSuite Tool – Metadevice Editor Window With Concatenate/Stripe Icon

- Click Slices to view the disk slices (Figure 4-31) if they are not already displayed.

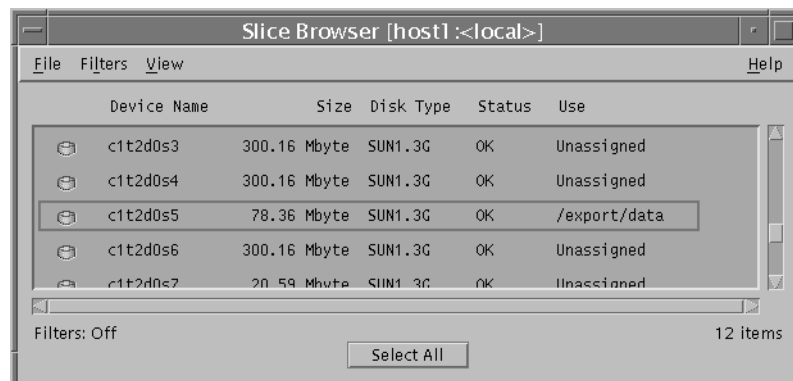


Figure 4-31 Slice Browser Window

- Drag and drop the /export/data file system from the Slices view onto the Concatenate/Stripe template icon.

Note – DiskSuite Tool warns you that you have mounted file systems for the /export/data file system. Click Continue in the warning pop-up window (Figure 4-32).

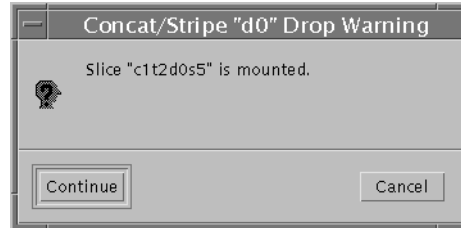


Figure 4-32 Concat/Stripe “d0” Drop Warning Window

The Concat/Stripe icon in the DiskSuite Metadevice Editor work area displays the Status of OK (Uncommitted) for slice c1t2d0s5 as stripe 0 of d0.

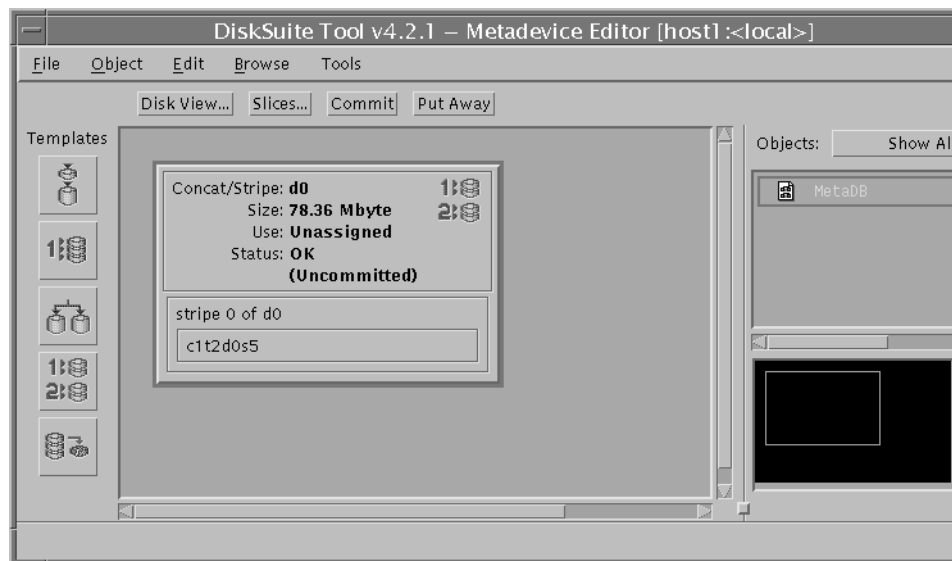


Figure 4-33 DiskSuite Tool – Metadevice Editor Window After Drag and Drop

5. Click Commit to save the change.

This modifies the /etc/vfstab file logical device name entry for /export/data from /dev/dsk/c1t2d0s5 to a new entry for the metadisk information, /dev/md/dsk/d0.

6. When the pop-up window labeled Concat/Stripe "d0" Commit Warning appears, click Really Commit.



Figure 4-34 Concat/Stripe "d0" Commit Warning Window

The Metadevice Editor window reflects the Status as OK.

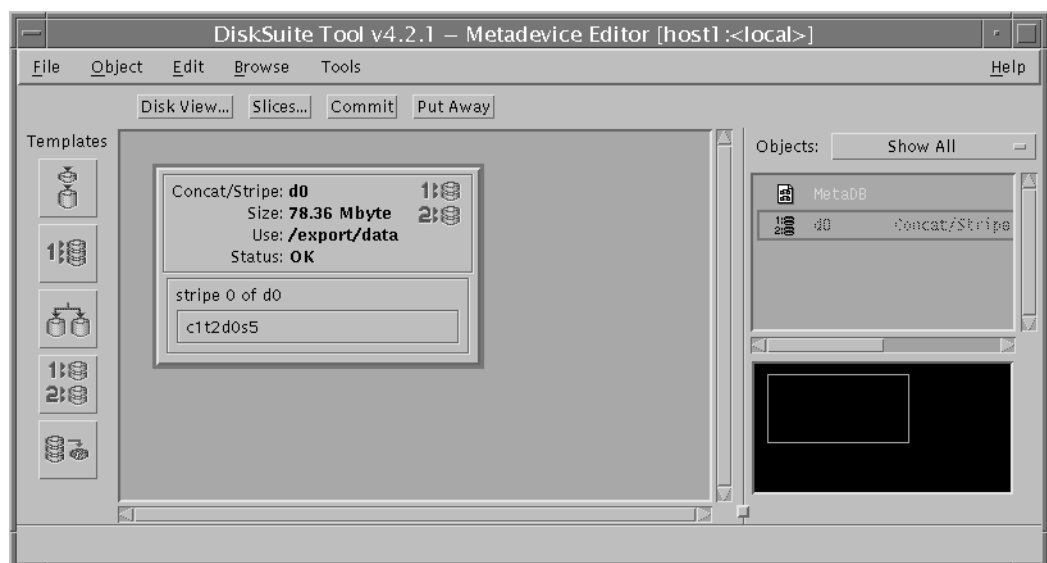


Figure 4-35 DiskSuite Tool – Metadevice Editor Window With d0 Status OK

7. Click Put Away.

The Concat/Stripe object moves from the work area of the Metadevice Editor to the Objects area.

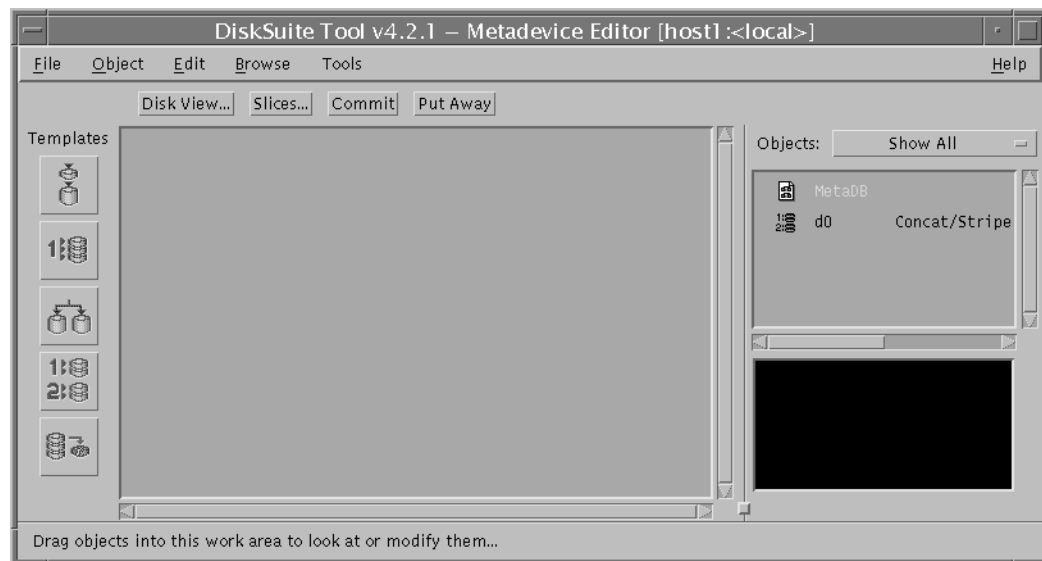


Figure 4-36 DiskSuite Tool – Metadevice Editor Window

8. Exit DiskSuite Tool and reboot the system or simply unmount and remount the `/export/data` file system to see the logical device name change from `/dev/dsk/c1t2d0s5` to `/dev/md/dsk/d0` in the output of the `mount` command.

Before unmounting `/export/data`:

```
# mount -p | grep data
/dev/dsk/c1t2d0s5 - /export/data ufs - yes
rw,intr,largefiles,onerror=panic,suid,dev=2200000
```

Note – The `-p` option (of the `mount` command) prints the list of mounted file systems in the `/etc/vfstab` format. It must be the only option specified. You can also use the `df -k` command.

After unmounting and remounting `/export/data`:

```
# umount /export/data
# mount /export/data
# mount -p | grep data
/dev/md/dsk/d0 - /export/data ufs - yes
rw,intr,largefiles,onerror=panic,suid,dev=1540000
```


9. If you rebooted, wait for the system to present the Common Desktop Environment Welcome screen, log in as `root` and use a terminal window to start DiskSuite.

If you unmounted and remounted the `/export/data` file system, start DiskSuite Tool from a terminal window.

```
# /usr/sbin/metatool &
```

The DiskSuite Tool –Metadevice Editor window is displayed.

10. Double-click the Concat/Stripe object in the Objects area (on the right side of the DiskSuite Tool window) to bring it to the Work Area canvas (Figure 4-37).

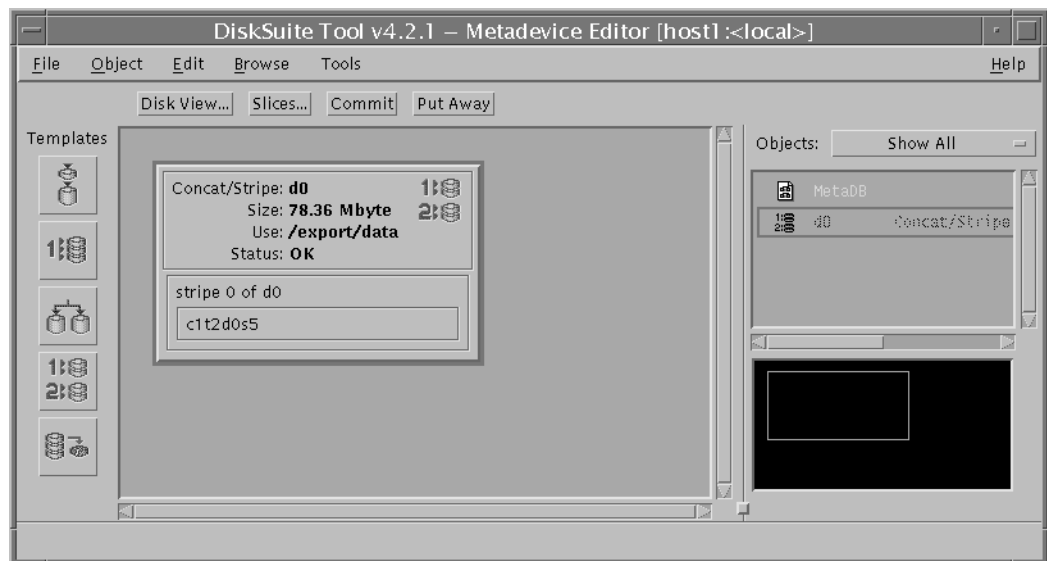


Figure 4-37 DiskSuite Tool – Metadevice Editor Window Displaying Metadevice `d0`

11. Click Slices to display the Slice Browser if it is not already running.

12. Select a slice containing the word “Unassigned” in the Use column from the Slice Browser.

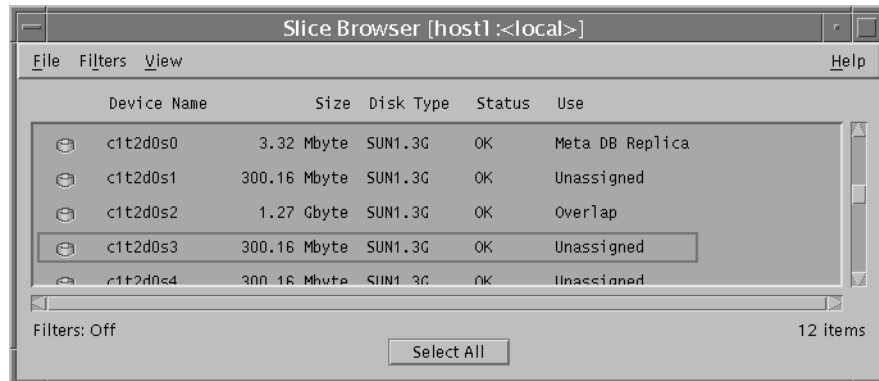


Figure 4-38 Slice Browser Window Displaying Unassigned Slice 3

13. Drag and drop that slice onto the Concat/Stripe icon in the canvas area.

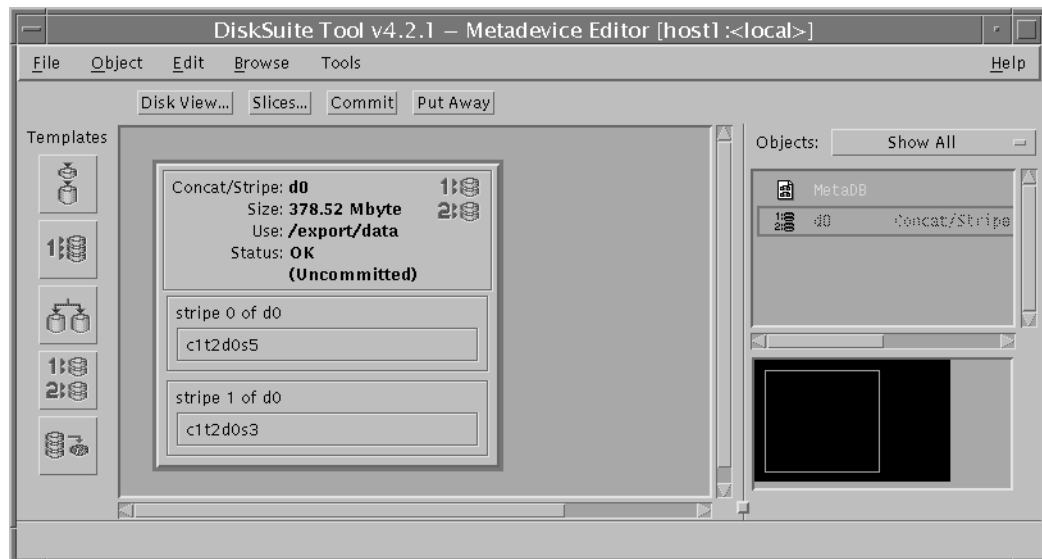


Figure 4-39 DiskSuite Tool – Metadevice Editor Window About to Concatenate Slice 3 and 5

14. Click Commit in the Metadevice Editor to save the change.

The Run GrowFS Command window (Figure 4-40) is displayed.



Figure 4-40 Run GrowFS Command Window

15. Click Grow Now.

While the change is taking place, a GrowFS Running window is displayed.

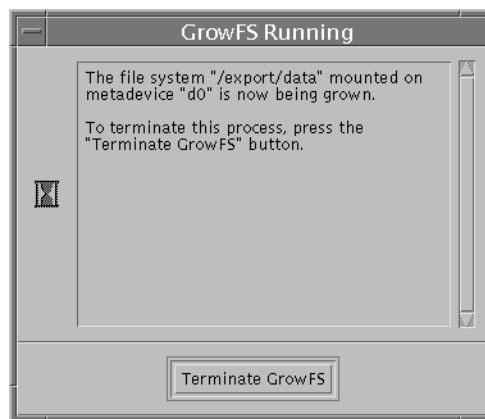


Figure 4-41 GrowFS Running Window

Note – Using the Solstice DiskSuite metadvice instead of the traditional disk device is a *permanent* change.

16. Run the `df` command to verify the increase in size of the `/export/data` directory:

```
# df -k /export/data
```

Filesystem	kbytes	used	avail	capacity	Mounted on
/dev/md/dsk/d0	363967	65273	291197	19%	/export/data

The output from this command shows the entry in the `Filesystem` column to be `/dev/md/dsk/d0` rather than `/dev/dsk/c1t2d0s5` as previously displayed (*prior* to the creation of the metadvice for that slice). This output verifies that the `/export/data` file system has grown to include the size of Slice 3.

Note – The original data in the `/export/data` directory is preserved.

Exercise: Managing Disks



Exercise objective – In this lab, you install Solstice DiskSuite software, partition a spare disk, and use DiskSuite Tool to grow the /export/data file system.

Preparation

Locate the Solstice DiskSuite software on the Solaris 8 software CD-ROM (2 of 2) as described in the “Installing the Solstice DiskSuite Software” section on page 4-11. Identify a disk attached to your system that is available for use in this lab. Refer to the lecture notes as necessary to perform the steps listed.

Your instructor will give you any last minute exercise preparation details that might be required.

Task Summary

In this exercise, you accomplish the following:

- Install the Solstice DiskSuite 4.2.1 software from the Solaris 8 software CD-ROM (2 of 2) and reboot the system.
- Use the `format` utility to partition a spare disk to assign 5 Mbytes to Slice 0; 20 Mbytes to Slice 7; 20 percent of the remaining disk space to Slices 1, 3, and 4; and allow Slice 5 to hold all remaining space.
- Use Solstice DiskSuite to create two concatenations. Grow the existing /export/data and /export/home file systems using Slices 3 and 4 of the spare disk. This task is done in three phases:
 - ▼ Modify the MetaDB object to hold three state-database replicas on the 5-Mbyte Slice 0 of the spare disk.
 - ▼ Create one Concat/Stripe object (d0) for the /export/data file system. Create one Concat/Stripe object (d1) for the /export/home file system. Unmount and remount each file system or reboot the system.

- ▼ Add Slice 3 from the spare disk to the Concat/Stripe object (d0) for /export/data. Grow the /export/data file system onto the new space. Add Slice 4 from the spare disk to the Concat/Stripe object (d1) for /export/home. Grow the /export/home file system onto the new space. Check the /etc/vfstab file for changes. Verify the new disk space exists for both of these file systems.

Tasks

Installing Solstice DiskSuite

To install the Solstice DiskSuite 4.2.1 application software, perform the following steps:

1. Follow the steps described in “Installing the Solstice DiskSuite Software” section on page 4-11.

Partitioning a Spare Disk

2. In a terminal window, start the **format** utility.

format

3. From the list of disks displayed, select the disk that you did not use during the Solaris installation process, being certain to avoid selecting any disk that is currently in use.

4. From the format menu, select the partition item.

```
format> partition
partition>
```

5. From the partition menu, select **modify**. Use the All Free Hog method to partition your spare disk. Use Partition 5 as the Free Hog. Assign space to partitions according to the following table:

Slice	Size
0	5 Mbytes
1	200 Mbytes
3	200 Mbytes
4	200 Mbytes
6	0 Mbytes
7	0 Mbytes

The following example reflects responses to the format prompts when partitioning a 4-Gbyte external SCSI disk on an Ultra 10 system:

```
(partition table)
partition> modify
Select partitioning base:
0. Current partition table (unnamed)
1. All Free Hog
Choose base (enter number) [0]? 1
Do you wish to continue creating a new partition table based on the above
table [yes]? y
Free Hog partition[6]? 5
Enter size of partition '0' [0b, 0c, 0.00mb, 0.00gb]: 5mb
Enter size of partition '1' [0b, 0c, 0.00mb, 0.00gb]: 800mb
Enter size of partition '3' [0b, 0c, 0.00mb, 0.00gb]: 800mb
Enter size of partition '4' [0b, 0c, 0.00mb, 0.00gb]: 800mb
Enter size of partition '6' [0b, 0c, 0.00mb, 0.00gb]: <Return>
Enter size of partition '7' [0b, 0c, 0.00mb, 0.00gb]: 20mb
Okay to make this the current partition table [yes]? y
Enter table name (remember quotes): "test"
Ready to label disk, continue? y
```

6. Quit the partition menu and the format utility.

```
partition> q
format> q
```

Creating Concatenations

This section of the exercise is divided into three parts:

1. Creating state database replicas
2. Creating concatenation objects for /export/data and /export/home
3. Attaching new slices to the /export/data and /export/home metadisks.

You need to reboot the system between the first and second parts of the lab.

Creating State-Database Replicas

4. Open a terminal window and start DiskSuite Tool.

```
# /usr/sbin/metatool &
```

DiskSuite Tool displays an warning message indicating that no state- database replicas exist.

5. Click OK to dismiss this message.
6. In the object window, double-click the MetaDB object.

This places the object on the DiskSuite canvas. Note that the status of this object is Critical (see Figure 4-22 on page 4-23).
7. Display the menu on the MetaDB and select the Info item.

The information window displays (see Figure 4-24 on page 4-24).
8. In the text field labeled Slice, enter the name of the 5-Mbyte slice you created earlier. This should be Slice 0 of the spare disk (for example, c1t3d0s0).
9. In the text field labeled Replicas, replace the initial value of 1 with 3.
10. Click Attach, and then close the information window.
11. Verify that the MetaDB object is selected in the canvas area, and click Commit.

A warning message indicating that all of the replicas you are making are on the same controller is displayed. Although on a production system you would avoid doing this if possible, accept the situation for the purposes of this lab.

12. Click Really Commit.

The MetaDB object status should now display Attention in yellow.

13. Click Put Away to remove the MetaDB object from the canvas.

Creating Concatenation Objects for /export/data and /export/home

14. Click Slices to display the Slice Browser.

This should resemble Figure 4-31 on page 4-29.

15. Click the Concat/Stripe icon to the left of the canvas area. (It is the icon with the two disks, labeled 1 and 2.)

This places a new Concat/Stripe object labeled d0 in the Metadevice Editor canvas area.

16. In the Slice Browser, locate the slice that is currently used to hold the /export/data file system, and use the SELECT button to drag that slice from the Slice Browser onto the new Concat/Stripe object in the canvas area.

A warning indicating that the slice is mounted is displayed.

17. Click Continue.

18. Verify that the Concat/Stripe object is selected in the canvas area, and click on Commit.

A warning indicating that changes will be made in /etc/vfstab is displayed.

19. Click Really Commit.

20. Click Put Away to place the d0 Concat/Stripe object back in the Object list area.

21. Repeat Step 15 through Step 20 to create a second Concat/Stripe object, called d1, and attach the slice used for /export/home.

At the end of this process you should have three objects in your object list:

- ▼ One MetaDB object, which uses cxtxdxs0 to hold three state database replicas
- ▼ One Concat/Stripe object for the /export/data directory called d0, where Stripe 0 of the object is cxtxd0s5
- ▼ One Concat/Stripe object for the /export/home directory called d1, where Stripe 0 of the object is cxtxd0s7

22. Examine the entries in the Slice Browser for the `cxtxd0s4` and `cxtxd0s7` slices. What has changed in their Use column?
23. Exit DiskSuite Tool.
24. Reboot the system or unmount and remount the `/export/data` and `/export/home` file systems to see the change in the logical disk device name.
25. If you rebooted, wait for the system to present the Common Desktop Environment Welcome screen, log in as `root`, and use a terminal window to start DiskSuite Tool.

If you unmounted and remounted the `/export/data` and `/export/home` file systems, start DiskSuite Tool from a terminal window.

Attaching New Slices to /export/data and /export/home

26. Open a terminal window and start DiskSuite Tool.
/usr/sbin/metatool &
27. Double-click on the `d0` object to move it into the canvas area.
28. Click Slices to display the Slice Browser.
29. In the Slice Browser, locate the unassigned Slice `s3` from your spare disk (check the Use column). Drag this slice onto the `d0` Concat/Stripe object in the canvas area, and be certain to drop it on the main `d0` object, and not on the other slice already in place.
30. Click Commit.

A dialog box asks if you want to grow the file system now or later. While a file system grows it must be idle. DiskSuite Tool prevents I/O to this device while growing the file system.

31. Click Grow Now.

The Concat/Stripe object reflects the change in Status of **OK**.

32. Put away the `d0` object.
33. Repeat Step 27 through Step 32 for the `d1` Concat/Stripe object, but use the unassigned slice `s4`.

At the end of this process you should have three objects in your object list similar to the following:

- ▼ One MetaDB object, which uses `cxtxd0s0` to hold three state database replicas
- ▼ One Concat/Stripe object called `d0`, where Stripe 0 of the object is `cxtxd0s5`, and Stripe 1 is `cxtxd0s4`
- ▼ One Concat/Stripe object called `d1`, where Stripe 0 of the object is `cxtxd0s7`, and Stripe 1 is `cxtxd0s3`

Note – The actual controller (`cx`) and target number (`tx`) can vary, depending on system disk configuration.

34. Exit the DiskSuite Tool.
35. Use the `df -k` command to check for the new space on the `/export/data` and `/export/home` file systems.
36. Examine the `/etc/vfstab` file to see the changes that have been made there.

Exercise: Managing Disks

Exercise Summary



Discussion – Take a few minutes to discuss the experiences, issues, or discoveries that you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ List the three utilities used to create, check, and mount file systems
- ☐ Identify the physical path name differences between physical disks and virtual disks
- ☐ List the potential advantages of any virtual disk management application
- ☐ List the basic difference between Solstice DiskSuite and Sun StorEdge Volume Manager
- ☐ List the main advantages of using a concatenated virtual file system
- ☐ List the main advantage of using a striped virtual file system
- ☐ Install the Solstice DiskSuite applications
- ☐ Use the Solstice DiskSuite application to dynamically grow a file system

Further Study

The Solstice DiskSuite and Sun StorEdge Volume Manager are complex applications. Sun Educational Services has training available for each of these products.

- ES-310: *Volume Manager With SPARCstorage Array*
- ES-220: *Disk Management With DiskSuite*

Solaris Pseudo File Systems and Swap Space

5 

Objectives

Upon completion of this module, you should be able to:

- List the Solaris pseudo file system types
- Describe the relationship between system processes and the `/proc` directory
- Describe how the `tmpfs` file system improves performance
- Use the `dumpadm` program to display system dump configuration
- Use the `coreadm` command to display core file configuration
- Create and add a swap file or partition to the swap space

Additional Resources



Additional resources – The following references provide additional details on the topics discussed in this module:

- *System Administration Guide, Volume I*, Part Number 805-3727-10
- *System Administration Guide, Volume II*, Part Number 805-3728-10

Solaris Pseudo File Systems

Pseudo file systems are sometimes called RAM-based file systems. Their most distinguishing feature is that they do not reside on hard physical media. They reside only in physical memory while the operating system is running.

You use pseudo file systems to increase performance. They enhance performance is enhanced because they provide access to data in physical memory, as opposed to disk-based structures. They enable the use of typical file system operation semantics (for example, the use of the standard system calls) for access to the underlying data structures.

The pseudo file systems supported in the Solaris 8 Operating Environment include:

- **procfs** – The Process file system contains a list of active processes, named according to process number, in the `/proc` directory. Information in this directory is used by commands, such as the `ps` command. See the `proc(4)` man page.
- **tmpfs** – The Temporary file system for file storage in memory without the overhead of writing to a disk-based file system. It is created and destroyed every time the system is rebooted.
- **fdfs** – The File Descriptor file system provides explicit names for opening files using file descriptors (for example, `/dev/fd/0`, `/dev/fd/1`, `/dev/fd/2`) in the `/dev/fd` directory.
- **swapfs** – The Swap file system used by the kernel to manage swap space on disk(s).

The /proc File System

The /proc directory is a mount point for a pseudo file system that provides access to the state of each process and light-weight process (LWP) in the system. You can write applications to access this state information using the standard system calls.

The process information stored in the /proc file system changes as the process moves through its life cycle.

Beginning with the Solaris 2.6 Operating Environment release, the previously flat /proc file system was restructured into a directory hierarchy that contains additional subdirectories for state information and control functions.

The following are the characteristics of the new directory structure of /proc:

- The name of each entry in the /proc directory is a decimal number corresponding to a process ID.
- Each process ID named directory in /proc has files that contain more detailed information about that process.
- The owner of each file in /proc directory and below is determined by the user ID of the process.

The /proc directory is mounted at system boot time by scripts called from /sbin/rcS. The following example from the /etc/vfstab file shows the mounting of the proc file system on the /proc mount point:

#device	device	mount	FS	fsck	mount	mount
#to mount	to fsck	point	type	pass	at boot	options
/proc	-	/proc	proc	-	no	-

The tmpfs File System

The tmpfs file system uses the virtual memory (VM) subsystem. Once this file system is mounted, it supports standard file operations and semantics. Files and directories in this file system are temporary and are released when the tmpfs is unmounted or the system reboots.

This file system supports better performance by maintaining files and directories in RAM. This performance enhancement is most noticeable when a large number of short-lived files are written and accessed on this file system.

The following example from the `/etc/vfstab` file shows the mounting of tmpfs on the virtual memory subsystem at boot time:

#device	device	mount	FS	fsck	mount	mount
#to mount	to fsck	point	type	pass	at boot	options
swap	-	/tmp	tmpfs	-	yes	-

As a result of using tmpfs, all data written to `/tmp` is written to RAM if space is available. If RAM space is not available, then any data written to `/tmp` is written to swap space instead.

The `fdfs` File System

The `fdfs` file system is a pseudo file system that maintains a repository of file descriptors for open files. Running programs access files by using these file descriptors.

The following example from the `/etc/vfstab` file shows the mounting of the `ifdfs` file system on the `/dev/fd` mount point at system boot time:

<code>#device</code>	<code>device</code>	<code>mount</code>	<code>FS</code>	<code>fsck</code>	<code>mount</code>	<code>mount</code>
<code>#to mount</code>	<code>to fsck</code>	<code>point</code>	<code>type</code>	<code>pass</code>	<code>at boot</code>	<code>options</code>
<code>fd</code>	<code>-</code>	<code>/dev/fd</code>	<code>fd</code>	<code>-</code>	<code>no</code>	<code>-</code>

Table 5-1 describes each file descriptor.

Table 5-1 File Descriptor Usage

File Descriptor	Description
<code>/dev/fd/0</code>	Standard input (<code>stdin</code>)
<code>/dev/fd/1</code>	Standard output (<code>stdout</code>)
<code>/dev/fd/2</code>	Standard error (<code>stderr</code>)
<code>/dev/fd/3</code>	Name of file (<code>file</code>)

The swap's File System

The Solaris Operating Environment software can use disk partitions for temporary memory storage, in addition to using partitions to store file systems. Partitions used to store memory images are called *swap partitions*.

Swap partitions are used as virtual memory storage areas when the system does not have enough physical memory to handle the needs of the currently running processes. Additionally, *swap files* can be used to augment swap space.

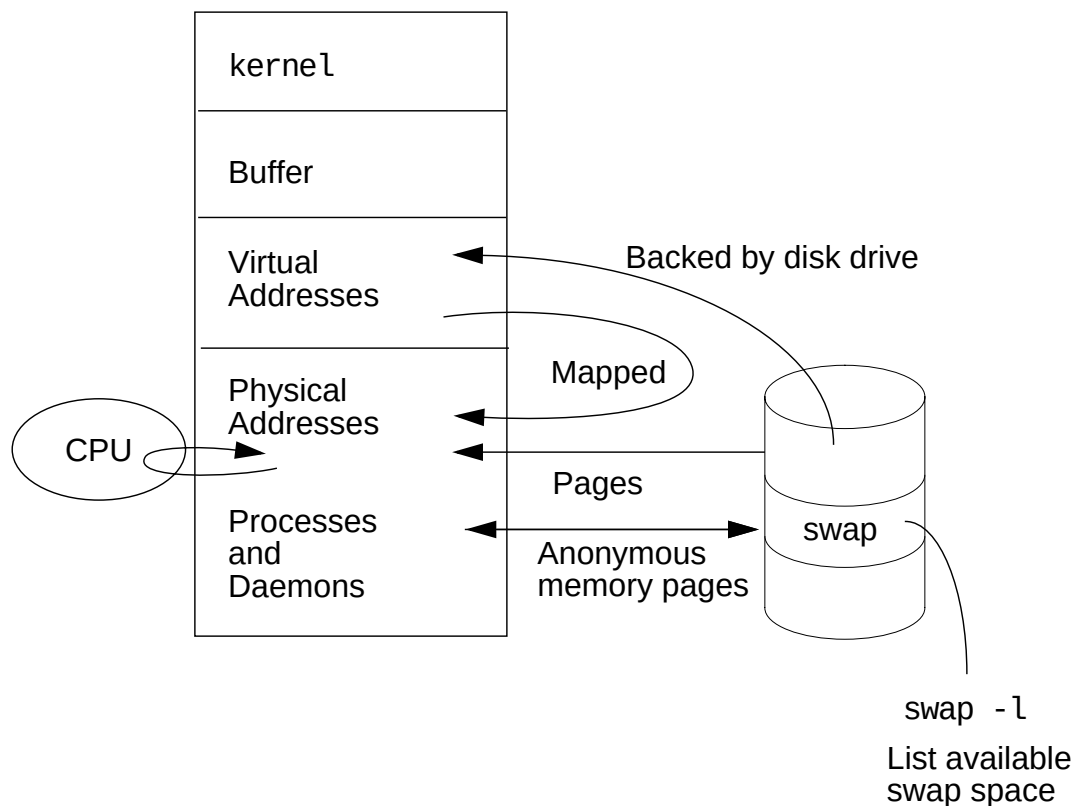


Figure 5-1 Swap Space Definition

Virtual and Physical Addresses

The Solaris virtual memory system maps the files on disk to virtual addresses in memory. As the instructions or static data in those files are needed, the virtual memory system maps the virtual addresses in memory to real physical addresses in memory. The data or instructions in those files are then paged from the disk into physical memory for use by the CPU. These types of physical pages of memory are always backed by known files on the disk.

Anonymous Memory Pages

In addition to containing program instructions or static data, physical memory pages can contain private data or stack information generated by running processes. The information in these pages of physical memory is not backed by a file in the file system. Therefore, these pages can be backed only by swap space on disk in the event that they must be temporarily paged out of memory. Because these private data or stack pages in physical memory are not backed by an actual file on the disk, but solely by swap space, they are referred to as anonymous memory pages.

Reserving Swap Space

When a process is run by the kernel, swap space for any private data or stack space used by the process must be reserved. This reservation occurs just in case the private data or stack information would have to be paged out of physical memory, due to multiple processes contending for limited memory space.

Without the use of virtual swap, large amounts of physical swap space would have to be configured on systems to accommodate these reservations. Even systems capable of avoiding paging by having large amounts of physical memory available would still need large swap areas configured for these reservations just in case.

However, with the virtual swap space provided in the Solaris Operating Environment by the `swapfs` file system, the need for configuring large amounts of physical swap space can be reduced on systems with large amounts of available memory. This reduced need for physical swap space can occur because `swapfs` provides virtual swap space addresses rather than real physical swap space addresses in response to the requests to reserve swap space.

With `swapfs` providing virtual swap space, real physical swap space is required only with the onset of paging, due to processes contending for memory. In this situation, `swapfs` must convert the virtual swap space addresses to physical swap space addresses for paging to actual swap space to occur.

Criteria for Swap Space

With the addition of `swapfs`, the size of swap space is based entirely on two criteria:

- To save any possible panic dumps resulting from a fatal system failure, there must be sufficient swap space to hold the necessary memory pages in RAM at the time of the failure.
- The amount of RAM + swap memory must be at least equal to the requirements of both the Solaris Operating Environment and any concurrently running processes.

Swap Space

If you use `tmpfs`, you should be aware of some constraints involved in mounting a `tmpfs` file system. The resources used by `tmpfs` are the same as those used when commands are executed. This means that large sized `tmpfs` files can affect the amount of space left over for programs to execute. Likewise, programs requiring large amounts of memory use up the space available to `tmpfs`. Users running into this constraint (for example, running out of space on `tmpfs`) can allocate more swap space by using the `swap` command.

You can add or delete swap space using the `swap` command. When swap files or swap partitions are mounted for access by the kernel memory manager, the file type used is `swap` (observe the contents of the `/etc/vfstab` file).

Using the `swap` Command

As the system administrator, you can add swap files or partitions.

Command Format

```
swap [ options ] [ argument ]
```

Options

- `-l`
Lists swap space
- `-a`
Adds to swap
- `-d`
Deletes from swap
- `-s`
Summarizes swap space

Adding a Swap File

Figure 5-1 illustrates the allocation of swap space.

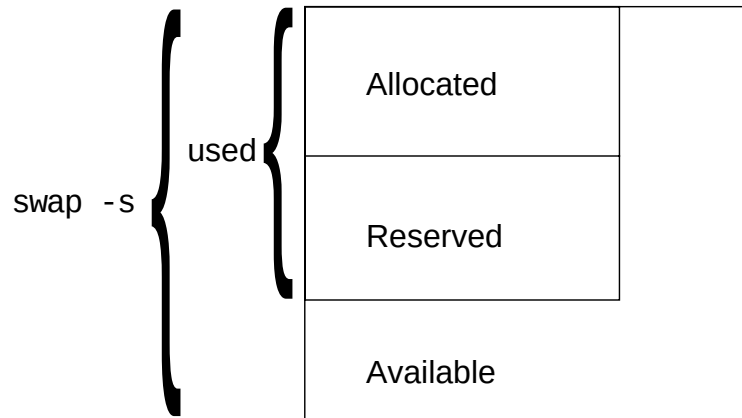


Figure 5-2 Swap Space Allocation

To add a swap file, complete the following steps:

1. List a summary of the system's virtual swap space.

```
# swap -s
total: 25728k bytes allocated + 6140k reserved = 31868k used, 56496k
available
```

2. List the details of the system's physical swap space.

```
# swap -l
swapfile      dev  swaplo blocks   free
/dev/dsk/c0t3d0s1  32,28      8 98792  90384
```

3. Using the `df` command, display the amount of disk space occupied by currently mounted file systems, the amount of used and available space, and how much of the file system's total capacity has been used. From this output, determine which partition has enough space for a swap file of at least 20 Mbytes.

```
# df -k
Filesystem      kbytes    used   avail capacity  Mounted on
/dev/dsk/c0t3d0s0 245455   87061  158149    36%      /
/dev/dsk/c0t3d0s6 480815  375163  105172    79%     /usr
/proc              0         0         0      0%     /proc
fd                 0         0         0      0%    /dev/fd
/dev/md/dsk/d0    231815     82   231502     1%  /export/data
/dev/md/dsk/d1     67159      9    67083     1%  /export/swap
swap             103844    204   103640     1%    /tmp
```


The /export/data file system appears to have more than enough space to create an additional swap file. Create a 20-Mbyte swap file named swapfile in the /export/data directory.

```
# mkfile 20m /export/data/swapfile
```

4. Add a swap file to the system's swap space.

```
# swap -a /export/data/swapfile
```

5. List the details of the modified system swap space.

```
# swap -l
swapfile          dev      swaplo      blocks free
/dev/dsk/c0t3d0s1  32,28      8          98792  90384
/export/data/swapfile  -          8          20472  20472
```

6. List a summary of the modified system swap space.

```
# swap -s
total: 25728k bytes allocated + 6140k reserved = 31868k used, 66708k
available
```

Removing a Swap File

To remove a swap file, complete the following steps:

1. To delete a swap file while online, issue the following command. (Deleting the swap file stops swapping and empties the specified disk space.)

```
# swap -d /export/data/swapfile
```

2. Remove the swap file to free disk space.

```
# rm /export/data/swapfile
```

Adding a Swap Slice

To add a swap slice, complete the following steps:

1. Add information about the swap partition you created to the file system table (the `/etc/vfstab` file).

```
# vi /etc/vfstab
```

#device	device	mount	FS	fsck	mount	mount
#to mount	to fsck	point	type	pass	at boot	opt
/dev/dsk/c0t2d0s1	-	-	swap	-	no	-

2. Reboot the system or use the `swap -a` command to add the additional swap area.

Adding a Permanent Swap File Using the /etc/vfstab File

To add a permanent swap file, complete the following steps:

1. Edit the `/etc/vfstab` file and add the entry for the file.

```
# vi /etc/vfstab
```

#device	device	mount	FS	fsck	mount	mount
#to mount	to fsck	point	type	pass	at boot	opt
/export/data/swapfile	-	-	swap	-	no	-

2. Reboot the system or use the `swap -a` command to add additional swap space.

The dumpadm Command

The `dumpadm` program is an administrative command that manages the configuration of the operating system crash dump facility.

Note – A panic dump contains a copy of the “interesting portions” of physical memory at the time of a fatal system error.

If a fatal operating system error occurs, a message describing the error is printed to the console. The operating system then generates a crash dump by writing the contents of physical memory to a predetermined dump device, which is typically a local disk partition. The dump device can be configured by using `dumpadm`. Once the crash dump has been written to the dump device, the system reboots.

Fatal operating system errors can be caused by bugs in the operating system, its associated device drivers and loadable modules, or by faulty hardware. Whatever the cause, the crash dump itself provides invaluable information to your support engineer to aid in diagnosing the problem.

Following an operating system crash, the `savecore(1M)` utility is executed automatically during a boot up to retrieve the crash dump from the dump device. It then writes the crash dump to a pair of files in your file system named `unix.X` and `vmcore.X`, where `X` is an integer identifying the dump.

The kernel core information placed in the file `/var/crash/'uname -n'/vmcore.X` is accessed from the device `/dev/mem`. The name list information placed in the file `/var/crash/'uname -n'/unix.X` is accessed from the device `/dev/ksyms`.

Together, these data files form the saved crash dump. The directory in which the crash dump occurred is saved when you reboot, and you can use the `dumpadm` command to configure it.

By default, the dump device is configured to be an appropriate swap partition. Swap partitions are disk partitions reserved as virtual memory backing store for the operating system, and thus no permanent information resides there to be overwritten by the dump.

To view the current dump configuration, execute `dumpadm` with no arguments. For example:

```
# dumpadm
Dump content: kernel pages
Dump device: /dev/dsk/c0t0d0s1 (swap)
Savecore directory: /var/crash/host1
Savecore enabled: yes
```

When no options are specified, `dumpadm` prints the current crash dump configuration. The previous example shows the set of default values: the dump content is set to kernel memory pages only, the dump device is a swap disk partition, the directory for savecore files is set to `/var/crash/hostname`, and savecore is set to run automatically on reboot. The default values are set in the `/etc/dumpadm.conf` file. For example:

```
# cat /etc/dumpadm.conf
# dumpadm.conf
#
# Configuration parameters for system crash dump.
# Do NOT edit this file by hand -- use dumpadm(1m) instead.
#
DUMPADM_DEVICE=/dev/dsk/c0t0d0s1
DUMPADM_SAmdir=/var/crash/host1
DUMPADM_CONTENT=kernel
DUMPADM_ENABLE=yes
```

Note – All modifications to the `dumpadm` configuration should be done at the command line using the `dumpadm` utility, rather than attempting to edit the `/etc/dumpadm.conf` file. This could result in an inconsistent system dump configuration.

Command Format

```
/usr/sbin/dumpadm [-nuy] [-c content-type] [-d dump-device]  
[ -m min k | min m | min%] [-s savecore-dir] [-r root-dir]
```

- *-c content-type* – Specifies the contents of the crash dump.
 - ▼ *kernel* – Indicates kernel memory pages only.
 - ▼ *all* – Indicates all memory pages.
- *-d dump-device* – Modifies the dump configuration to use the specified dump device.
 - ▼ *dump-device* – Specifies a specific dump device specified as an absolute path name, such as */dev/dsk/c#t#d#s#*.
 - ▼ *swap* – Specifies the special token swap. If this swap is specified as the dump device, *dumpadm* examines the active swap entries and selects the most appropriate entry to configure as the dump device. See *swap(1M)*.
- *-m min k | min m | min%* – Creates a *minfree* file in the current *savecore* directory indicating that *savecore* should maintain at least the specified amount of free space in the file system where the *savecore* directory is located.
 - ▼ *k* – Indicates a positive integer suffixed with the unit *k* specifying kilobytes.
 - ▼ *m* – Indicates a positive integer suffixed with the unit *m* specifying megabytes.
 - ▼ *%* – Indicates a percent (%) symbol, indicating the *minfree* value should be computed as the specified percentage of the total current size of the file system containing the *savecore* directory.
- *-n* – Modifies the dump configuration so it does not run *savecore* automatically on reboot.
- *-r root-dir* – Specifies an alternative *root* directory relative to which *dumpadm* should create files. If no *-r* argument is specified, the default root directory *"/* is used.

- `-s savecore-dir` – Modifies the dump configuration to use the specified directory to save files written by `savecore`. The default `savecore` directory is `/var/crash/hostname` where *hostname* is the output of the `-n` option to the `uname(1)` command.
- `-y` – Indicates that `savecore` is automatically run on reboot. This is the default for this dump setting.

The coreadm Command

Use the `coreadm` command to specify the name or location of core files produced by abnormally-terminating processes.

The `coreadm` command provides flexible core file naming conventions and better core file retention. For example, you can use the `coreadm` command to configure a system so that all process core files are placed in a single system directory. This means it is easier to track problems by examining the core files in a specific directory whenever a Solaris process or daemon terminates abnormally.

Two new configurable core file paths, `per-process` and `global`, can be enabled or disabled independent of each other. When a process terminates abnormally, it produces a core file in the current directory, as in previous Solaris Operating Environment releases. But if a global core file path is enabled and set to `/corefiles/core`, for example, then each process that terminates abnormally produces two core files: one in the current working directory and one in the `/corefiles` directory.

Note – If the core file path does not exist, you must create it.

Command Format

The following command can be run by regular users and is used to specify the file name pattern to be used by the operating system when generating a per-process core file.

```
coreadm [-p pattern] [pid]...
```

The following command is run by `root` only and is used to configure system-wide core file options.

```
coreadm [ -g pattern ] [ -i pattern ] [ -d option ... ]  
        [ -e option ... ]
```

Default coreadm Command Without Options

Using the `coreadm` with no options displays the typical default settings from the `/etc/coreadm.conf` file.

```
# coreadm
  global core file pattern:
    init core file pattern: core
      global core dumps: disabled
    per-process core dumps: enabled
  global setid core dumps: disabled
per-process setid core dumps: disabled
  global core dump logging: disabled
```

The first line of output identifies the name to use for core files placed in a global directory. When generated, a global core file is created with mode 600 and is owned by the superuser. Non-privileged users cannot examine such files.

The second line of output identifies the name to be used if the `init` process generates a core file.

The third line indicates that global core files are disabled.

The fourth line indicates that core files in the current directory are enabled. Ordinary per-process core files are created with mode 600 under the credentials of the process. The owner of the process can examine such files.

In the fifth and sixth lines, if `setid` core files are enabled, they are created with mode 600 and are owned by the superuser.

The seventh line identifies whether the global core dump logging is enabled.



Caution – A process that has a `setuid` mode presents security issues with respect to dumping core files, as it might contain sensitive information in its address space to which the current non-privileged owner of the process should not have access. Normally `setuid` core files are not generated because of this security issue.

Note – Complete all modifications to the `coreadm` configuration at the command line using the `coreadm` utility instead of editing the `/etc/coreadm.conf` file. If you manually edit the `coreadm` configuration file, you must reboot the system or run `coreadm -u`.

Viewing the `/etc/coreadm.conf` file verifies the same configuration parameters that were described on page 5-18:

```
# cat /etc/coreadm.conf
# coreadm.conf
#
# Parameters for system core file configuration.
# Do NOT edit this file by hand -- use coreadm(1) instead.
COREADM_GLOB_PATTERN=
COREADM_INIT_PATTERN=core
COREADM_GLOB_ENABLED=no
COREADM_PROC_ENABLED=yes
COREADM_GLOB_SETID_ENABLED=no
COREADM_PROC_SETID_ENABLED=no
COREADM_GLOB_LOG_ENABLED=no
```

Patterns

A core file name pattern is a normal file system path name with embedded variables, specified with a leading percent (%) character. These variables are expanded from values in effect when a core file is generated by the operating system. The possible variables are:

- %p – Process ID
- %u – Effective user ID
- %g – Effective group ID
- %f – Executable file name
- %n – System node name (`uname -n`)
- %m – Machine hardware name (`uname -m`)
- %t – Decimal value of `time(2)`
- %% – Literal %

Examples

The following examples show various ways to use the `coreadm` command.

Example 1 – Setting the Core File Name Pattern as a Regular User

When executed from a user's `$HOME/.profile` or `$HOME/.login`, the following command sets the core file name pattern for all processes run during the login session:

```
$ coreadm -p core.%f.%p $$
```

Note – `$$` is the process ID of the currently running shell. The per-process core file name pattern is inherited by all child processes.

Example 2 – Dumping a User's Files into a Subdirectory

The following command dumps all of the user's core dumps into the `corefiles` subdirectory of the `home` directory, discriminated by the system node name. This is useful for users who use many different machines but have a shared `home` directory.

```
$ coreadm -p $HOME/corefiles/%n.%f.%p $$
```

Example 3 – Enabling and Setting the Core File Global Name Pattern

The following is an example of setting *system-wide* parameters that add the executable file name and PID to the name of any potential core file that might be created:

```
# coreadm -g /var/core/core.%f.%p -e global
```

For example, the core file name pattern: `/var/core/core.%f.%p` causes the `foo` program with process ID 1234, to generate the core file `/var/core/core.foo.1234`.

To verify that this parameter is now part of the `coreadm` configuration, run the `coreadm` command again:

```
# coreadm
  global core file pattern: /var/core/core.%f.%p
  init core file pattern: core
    global core dumps: enabled
    per-process core dumps: enabled
    global setid core dumps: disabled
per-process setid core dumps: disabled
  global core dump logging: disabled
```

Example 4 – Checking the Core File Configuration for Specific Process IDs

The `coreadm` command with only a list of process IDs reports each process's per-process core file name pattern, for example:

```
$ coreadm 278 5678
278: core.%f.%p
5678: /home/george/cores/%f.%p.%t
```

Only the owner of a process or the superuser can interrogate a process in this manner.

When a core dump occurs, the operating system generates two possible core files, the global core file and the per-process core file. Depending on the system options in effect, one file, both files, or no files can be generated. When generated, a global core file is created in Mode 600 and is owned by the superuser. Non-privileged users cannot examine such files.

Ordinary per-process core files are created in Mode 600 under the credentials of the process. The owner of the process can examine such files.

Options Supported by coreadm

The following are some useful options to the `coreadm` command.

- *-i pattern*

Sets the per-process core file name pattern for init to *pattern*. This is the same as `coreadm -p pattern 1` except that the setting is persistent across `reboot`. Only a superuser can use this option.

- *-e option*

Enables the specified core file option. Specifies the option as one of the following:

- ▼ `global`

Allows core dumps using the global core pattern.

- ▼ `process`

Allows core dumps using the per-process core pattern.

- ▼ `global-setid`

Allows `setid` core dumps using the global core pattern.

- ▼ `proc-setid`

Allows `setid` core dumps using the per-process core pattern.

- ▼ `log`

Generates a `syslog(3)` message when a user attempts to generate a global core file. Only superusers can use this option.

- *-d option*

Disables the specified core file option. See the *-e* option for descriptions of possible options. Multiple *-e* and *-d* options can be specified on the command line. Only *root* can use this option.

- *-u*

Updates system-wide core file options from the contents of the configuration file */etc/coreadm.conf*. If the configuration file is missing or contains invalid values, default values are substituted. Following the update, the configuration file is resynchronized with the system core file configuration. Only superusers can use this option.

Exercise: Managing Pseudo File Systems and Swap Space



Exercise objective – In this lab, you understand the pseudo file systems described in the module.

Preparation

Your instructor will give you any last-minute exercise preparation details that might be required.

Task Summary

In this exercise, you accomplish the following:

- Select a process from the process table and verify the process number, ownership, and process name being run with the appropriate entry in the `/proc` directory.
- Determine the amount of disk space being used by the swap file system (`/tmp`).
- Add swap space from a spare slice.
- Create a swap file and add it to the permanent swap space.
- Use the `dumpadm` command to display the location of the dump `Savecore` directory and force a `kernel panic` reboot to create `unix.X` and `vmcore.X` files in that directory.
- Use the `coreadm` command to display default initial configuration information and modify this information to change core file name and location.

Tasks

Viewing Processes and the Equivalent entries in the /proc Directory

1. Run the `ps -ef` command and select a process from the first 20 processes to investigate this process using the output (this can be done as a user or root).

```
# ps -ef | head -20
root      0      0 0   May 23 ?        0:18 sched
root      1      0 0   May 23 ?        0:01 /etc/init -
root      2      0 0   May 23 ?        0:00 pageout
root      3      0 0   May 23 ?        7:25 fsflush
root     398      1 0   May 23 ?        0:00 /usr/lib/saf/sac -t 300
root     241      1 0   May 23 ?        0:00 /usr/lib/utmpd
root     138      1 0   May 23 ?        0:00 /usr/sbin/keyserv
root      53      1 0   May 23 ?        0:00 /usr/lib/devfsadm/devfseventd
root      55      1 0   May 23 ?        0:00 /usr/lib/devfsadm/devfsadmd
root     184      1 0   May 23 ?        0:38 /usr/lib/autofs/automountd
root     135      1 0   May 23 ?        0:01 /usr/sbin/rpcbind
root     121      1 0   May 23 ?        0:00 /usr/lib/inet/in.ndpd
root     176      1 0   May 23 ?        0:00 /usr/sbin/inetd -s
root     219      1 0   May 23 ?        0:00 /usr/lib/lpsched
```

For the purpose of this exercise, select the process information from the output for `/usr/lib/lpsched` (PID 219 in this output).

2. Verify that there is a matching directory entry for that process in the `/proc` directory:

```
# ls -R /proc/219
dr-x--x--x  5 root      root      736 May 23 17:45 /proc/219
```

3. Run the `man -s4 proc` command and answer the following three questions.

a. What is contained in the file named `as`?

b. What is the `psinfo` file used for?

- c. What is the purpose of the `fd` directory?
-

Determining the Amount File Disk Space Used by the `swapfs` File System

- Run the `swap -s` command to view the total number of bytes actually allocated (and currently in use), the number of bytes allocated (and not currently in use but reserved by processes for possible future use), the total amount of swap space (both allocated and reserved), and the total swap space currently available for future reservation and allocation. For example:

```
# swap -s
total: 93344k bytes allocated + 12376k reserved = 105720k used, 197552k
available
```

- Run the command `swap -l` to list the physical swap area configured on your system. The output lists how much total swap space is in each swap device listed, and how much space is available for each of these devices.

```
# swap -l
swapfile      dev  swaplo blocks   free
/dev/dsk/c0t0d0s1 32,1      16 524384 361024
```

- Run the `df -k /tmp` command.

```
# df -k /tmp
Filesystem      kbytes    used   avail capacity  Mounted on
swap            196080      608  195472      1%      /tmp
```

- Using the procedure provided in the “Adding a Swap File” section on page 5-10, create a swap file and add it to the system swap space. Use both `swap -l` and `swap -s` to verify that the new swap space is available.
- Using the procedure provided in the “Removing a Swap File” section on page 5-11, remove the swap file created in the previous step. Once again use `swap -l` and `swap -s` to verify that the swap space is no longer available.
- Using the procedure provided in the “Adding a Swap Slice” section on page 5-12, add a disk partition as a swap slice to your existing swap space. Use Slice 1, which was created in Lab 4.

10. Add the new swap partition to the `/etc/vfstab` file to make it permanent. To verify this change, you must reboot the system.

Using dumpadm to Display the core File Directory Location

11. Use the `dumpadm` command to view the current dump configuration. Execute `dumpadm` with no arguments. For example,

```
# dumpadm
```

12. Fill in the configuration parameters from the output:

Dump content: _____

Dump device: _____

Savecore directory: _____

Savecore enabled: _____

13. Use the command `dumpadm` to change the dump device to be the new swap partition created in step 10.

```
# dumpadm -d /dev/dsk/c##t##s##
```

14. Reboot the system using the `-d` flag to `reboot(1M)` command, which forces the kernel to panic and save a crash dump.

Warning – This command should not be used on a system in a production environment.



```
# reboot -d
```

15. When the system reboots, make sure the crash dump succeeded by using the following commands to view the content of the `savecore` directory (output shown should be similar):

```
# cd /var/crash/<savecore directory>
```

```
# ls
```

```
bounds      unix.0      vmcore.0
```

```
# file vmcore.0
```

Note – Further investigation of the core files, `unix.0` and `vmcore.0` requires that you use the `mdb(1)`, the modular debugger.

Using coreadm to Display Default Configuration for Potential core Files

16. Use the `coreadm` command to display default initial configuration. The command and resulting output should be similar to the following:

```
# coreadm
  global core file pattern:
    init core file pattern: core
      global core dumps: disabled
    per-process core dumps: enabled
  global setid core dumps: disabled
per-process setid core dumps: disabled
  global core dump logging: disabled
```

17. View the `/etc/coreadm.conf` file:

```
# cat /etc/coreadm.conf
```

18. Enable a global core file path and create the core file directory.

```
# coreadm -e global -g /var/core/core.%f.%p
# mkdir /var/core
```

19. Enable global setid core dumps.

```
# coreadm -e global-setid
```

20. Turn on logging to generate a message when a global core file is attempted.

```
# coreadm -e log
```

21. Display the configuration information to verify the changes:

```
# coreadm
  global core file pattern: /var/core/core.%f.%p
    init core file pattern: core
      global core dumps: enabled
    per-process core dumps: enabled
  global setid core dumps: enabled
per-process setid core dumps: disabled
  global core dump logging: enabled
```

22. In another terminal window, run the `ps` command to get the PID of the new shell. Also, run the `pwd` command to set the correct working directory. Send a SIGFPE signal (signal 8) to the new shell using the `kill` command. (SIGFPE will force a `core` file.)

```
# mkdir /dir
# cd /dir
# ps
# kill -8 PID
```

23. In the original terminal window, check to see if a `core` file exists in the current working directory of the old shell. Use the `file` command to verify the `core` file is from the old shell.

```
# cd /dir
# ls
# file core
```

24. Use the `ls` command to check for a `core` file in `/var/core`.

```
# ls /var/core
```

Exercise Summary



Discussion – Take a few minutes to discuss the experiences, issues, or discoveries that you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Task Solutions

Viewing Processes and the Equivalent entries in the /proc Directory

25. Run the `man -s4 proc` command and answer the following three questions.

- a. What is contained in the file named `as`?

This file contains the address-space image of the process, and it can be opened for both reading and writing.

- b. What is the `psinfo` file used for?

This file contains miscellaneous information about the process and the representative `lwp` needed by the `ps(1)` command. `psinfo` is accessible after a process becomes a zombie.

- c. What is the purpose of the `fd` directory?

This directory contains references to the open files of a process. Each entry is a decimal number corresponding to an open file descriptor in the process.

Using `dumpadm` to Display the core File Directory Location

26. Fill in the configuration parameters from the output:

Dump content: `kernel` pages

Dump device: `/dev/dsk/c0t0d0s1`

savecore directory: `/var/crash/hostname`

savecore enabled: *yes*

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ List the Solaris pseudo file system types
- ☐ Describe the relationship between system processes and the `/proc` directory
- ☐ Describe how the `tmpfs` file system improves performance
- ☐ Use the `dumpadm` program to display system dump configuration
- ☐ Use the `coreadm` command to display core file configuration
- ☐ Create and add a swap file or partition to the swap space

Objectives

Upon completion of this module, you should be able to:

- Describe the functions of an NFS server and an NFS client
- Make resources available and unavailable for mounting
- Edit the `/etc/dfs/dfstab` file on an NFS server to enable automatic sharing of resources
- Display a server's available resources for mounting
- Mount a resource from another system
- Edit the `/etc/vfstab` file to mount resources on an NFS client
- Describe the function of these commands: `mountall`, `umountall`, `shareall`, and `unshareall`
- Describe and configure NFS logging

Additional Resources



Additional resources – The following references provide additional details on the topics discussed in this module:

- *System Administration Guide, Volume I*, Part Number 805-7228-10
- *System Administration Guide, Volume II*, Part Number 805-7229-10
- *System Administration Guide, Volume III*, Part Number 806-0916-10

The NFS Distributed File System

The Solaris Operating Environment supports the sharing of remote file system resources and presents them to users as if they were local files and directories.

The sharing of remote file system resources is administered through distributed file systems (DFS) file system types. This file system type provides the architectural support required for mounting resources over the network.

The NFS environment contains the following components:

- *NFS server* – A system that contains the file resources to be shared with other systems on the network.
- *NFS client* – A system that mounts the file resources shared over the network and presents the file resources as if they were local.

Figure 6-1 illustrates an NFS environment.

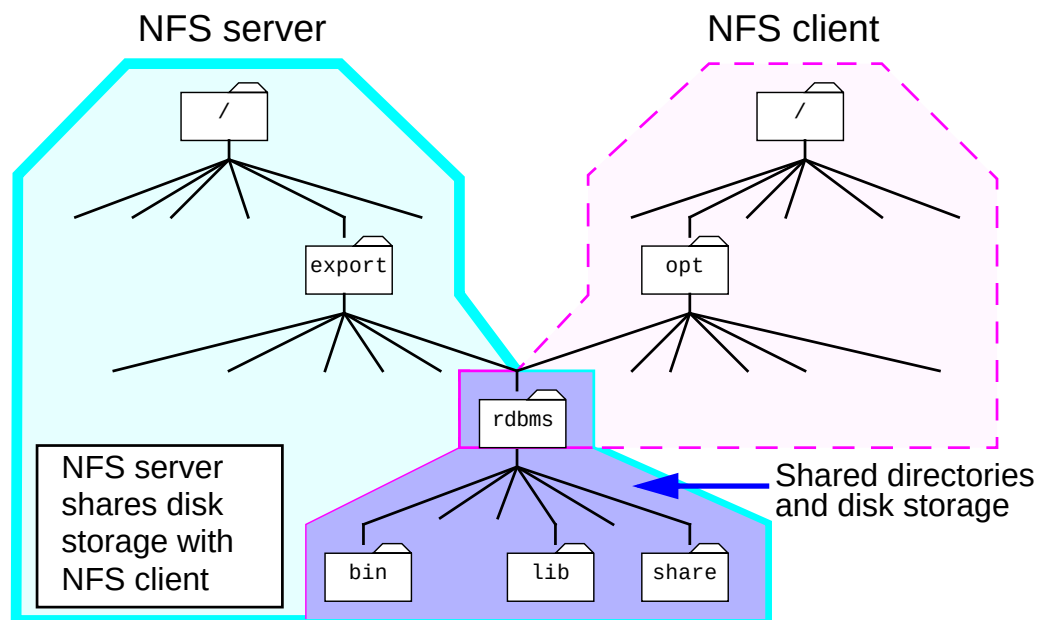


Figure 6-1 NFS Distributed File System

The Benefits of a Network File System

The benefits of an NFS include:

- Centralized file access

Files are located in centralized locations. You can make a copy of a file accessible to many users or systems simultaneously. This is an especially useful feature with home directories or common data files.

- Common software access

Systems can share one or more software packages, located in a central location. This reduces the disk space requirements for individual systems.

- Easy to use

Remote file sharing is transparent to the user and to any applications, because these resources appear as if they were resident on the local system.

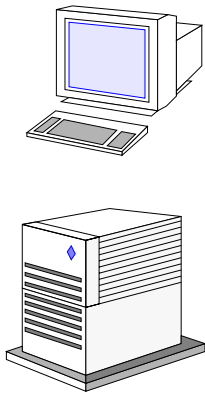
The NFS environment provides file sharing in a heterogeneous environment, potentially containing many different operating systems, including UNIX®, MS-DOS, and Virtual Memory System (VMS).

Note – NFS uses remote procedure calls (RPCs) and external data representation (XDR). XDR library routines allow programmers to describe arbitrary data structures in a machine-independent fashion.

NFS Distributed File System Components

The DFS administration files, commands, and daemons necessary for sharing and mounting NFS file resources are illustrated in Figure 6-2, for both an NFS server and an NFS client.

NFS server

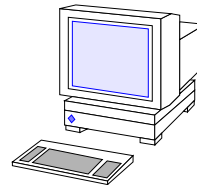


Daemons:
mountd, nfsd, statd, and
lockd, nfslogd

Files:
- /etc/dfs/dfstab
- /etc/dfs/sharetab
- /etc/dfs/fstypes
- /etc/rmtab
- /etc/nfs/nfslog.conf
- /etc/default/nfslogd
- /etc/nfs/nfslogtab

Commands:
share
unshare
shareall
unshareall
dfshares
dfmounts

NFS client



Daemons:
statd and lockd

Files:
- /etc/vfstab
- /etc/mnttab
- /etc/dfs/fstypes

Commands:
mount
umount
mountall umountall
dfshares
dfmounts

Figure 6-2 NFS Files, Commands, and Daemons

The NFS Daemons

NFS operation requires daemons running on the NFS server and NFS client.

The Mount Daemon

When an NFS client issues an NFS mount request, the mount process contacts the NFS server's mount daemon, `/usr/lib/nfs/mountd`, to get a *file handle* (pointer) for the file resource to be mounted.

The NFS client mount process then writes the file handle (along with other information about the mounted resource) to the `/etc/mnttab` file.

NFS Server Daemons

When a process on a client attempts to access a remote file resource, the NFS server daemon, `/usr/lib/nfs/nfsd`, on the server gets the request (along with the resource's file handle) and performs the file operation. It then returns any data to the requesting process on the client.

The server daemons are started from the `/etc/init.d/nfs.server` script. The `nfs.server` script also defines the maximum number of `nfsd` threads that can be started.

If a system has entries in its `/etc/dfs/dfstab` file, these server daemons are started when the system enters run level 3.

NFS Daemons on the Client and Server

Two other NFS daemons, `/usr/lib/nfs/statd` and `/usr/lib/nfs/lockd` run on both the NFS servers and clients. These daemons are started automatically when a system enters run level 2.

The two daemons work together to provide locking services in NFS. If the server crashes, clients can quickly re-establish the connections to files they were using. The server has a record of the clients that were using NFS. It contacts each client to obtain the information about which files were in use to allow continued operation. Both daemons are started from the `/etc/init.d/nfs.client` script, and typically do not require administrative intervention.

NFS File Handles

File handles are client references that identify a unique file or directory on the server. File handles encode the file's inode number, inode generation number, and disk device number.

Once a client successfully completes an NFS mount request, an entry is made in the `/etc/rmtab` file by the `mountd` daemon on the server. The `/etc/rmtab` file contains a table of file systems that are remotely mounted by NFS clients. It also contains a line entry for each remotely mounted file system. For example:

```
hostname:fsname
```

These line entries are removed from this file by the `mountd` command when it is first started.

Stale entries can accumulate in this file for clients that have crashed, and could not send an unmount request. Removing these entries allows the client to remount the resource.

The NFS Server

The following commands and files are used in conjunction with the NFS server.

The share Command

When the `mountd` daemon is running, use the `/usr/sbin/share` command to make file resources available for mounting by remote systems.

Command Format

```
share [ -F FSType ] [ -o options ] [ -d description ] pathname
```

Options

The following options can be used with the `share` command:

- **-F *nfs***

Specifies the file system type. This option is not typically required as `nfs` is the default remote file system type.

Note – If you do not use the option `-F fstypes`, the system takes the file system type from the first line of the `/etc/dfs/fstypes` file.

- **-o *options***

Controls a client's access to an NFS-shared resource.

- **-d *description***

Describes the file resource being shared. This information is displayed by the `share` command when used with no arguments.

- ***pathname***

Specifies the resource to be shared.

File Resource Sharing

To share a file resource from the command line, execute the following:

```
# share -F nfs -o ro /usr/share/man
```

The `share` command writes information for all shared file resources to the `/etc/dfs/sharetab` file. The file contains a table of the local resources shared.

If no argument is specified, the `share` command displays a list of all file resources currently shared.

```
# share
/usr/share/man
/export/install ro
```

The /etc/dfs/dfstab File

The `/etc/dfs/dfstab` file gives the system administrator a method for the automatic sharing of local file systems. Each line of the `dfstab` file consists of a `share` command.

```
# cat /etc/dfs/dfstab
# Place share(1M) commands here for automatic execution
# on entering init state 3.
#
# Issue the command '/etc/init.d/nfs.server start' to run the NFS
# daemon processes and the share commands, after adding the very
# first entry to this file.
#
# share [-F fstype] [-o options] [-d "<text>"] <pathname> [resource]
# e.g,
# share -F nfs -o rw=engineering -d "home dirs" /export/home2
#
```

The contents of the `/etc/dfs/dfstab` file are executed when:

- The system enters run level 3.
- The superuser runs the `shareall` command. The NFS daemons must be running.

- The superuser runs the `/etc/init.d/nfs.server` script (which contains a `shareall` command) with the `start` argument. This script starts the `nfs` server daemons.

Note – If the `nfs.server` script does not find NFS entries in the `/etc/dfs/dfstab` file, it exits without running the NFS daemons.

NFS Access Management

By default, NFS-mounted resources are available with read and write privileges based on standard Solaris file permissions. Access decisions are based on a comparison of the UID of the client and the owner.

The following `share` command options restrict the read and write capabilities for NFS clients and enable superuser access to a mounted resource.

- `ro`

Informs clients that the server accepts only read requests.

- `rw`

Allows the server to accept read and write requests from the client.

- `root=client`

Informs clients that the `root` user on the specified client system or systems can perform superuser privileged requests on the shared resource.

- `ro=access-list`

Allows read requests from the specified access list.

- `rw=access-list`

Allows read and write requests from the specified access list.

- ▼ `access-list=client:client`

Allows access based on a colon-separated list of one or more clients.

▼ *access-list=@network*

Allows access based on a network number (for example, @192.168.100) or network name (for example, @mynet.com). The network name must be defined in /etc/networks.

▼ *access-list=.domain*

Allows access based on a DNS domain; the dot (.) identifies it as a DNS domain.

▼ *access-list=netgroup_name*

Allows access based on a configured net group (NIS or NIS+ only).

▼ *anon=n*

Sets *n* to be the effective user ID of unknown users. By default, unknown users are given the effective user ID `UID_NOBODY`. If *n* is set to `-1`, access is denied.

You can combine these options by separating each with commas, forming intricate access restrictions.

Examples

```
# share -F nfs -o ro directory
```

This command line restricts access to NFS mounted resources to read-only access.

```
# share -F nfs -o ro,rw=client1 directory
```

This command line restricts access to NFS-mounted resources to read-only access; however, the NFS server accepts both read and write requests from the client named `client1`.

```
# share -F nfs -o root=client2 directory
```

This command line allows the `root` user on the client named `client2` to have superuser access to the NFS mounted resources.

```
# share -F nfs -o anon=0 directory
```

By setting the option `anon=0`, the EUID for access to shared resources is set to the UID of the user who is accessing the shared resource.

The unshare Command

The `/usr/sbin/unshare` command makes file resources unavailable for mounting by remote systems. It reads the `/etc/dfs/sharetab` file.

Command Format

```
unshare [ -F nfs ] pathname
```

Options

The following options can be used with the `unshare` command:

- **-F nfs**

Specifies `nfs` as the file system type. This option is not typically required because `nfs` is the default remote file system type.

- ***pathname***

Specifies the path name of the file resource to be unshared.

The following example makes the resource unavailable for mounting:

```
# unshare /usr/share/man
```

The shareall and unshareall Commands

Use the `/usr/sbin/shareall` and `/usr/sbin/unshareall` commands to share and unshare all NFS resources.

The shareall Command

Without any arguments, the `shareall` command shares all file resources listed in the `/etc/dfs/dfstab` file.

```
shareall [ -F nfs ]
```

The unshareall Command

Without any arguments, the `unshareall` command unshares currently shared file resources. It does this by reading the `/etc/dfs/sharetab` file.

```
unshareall [ -F nfs ]
```

Configuring the NFS File Server

To set up an NFS server, complete the following steps:

1. Edit the `/etc/dfs/dfstab` file and add those file resources to be automatically shared whenever the system enters run level 3. For example:

```
share -F nfs /usr/share/man
```

2. Start the NFS server daemons by invoking the following:

```
# /etc/init.d/nfs.server start
```

This shares the contents of the `/etc/dfs/dfstab` file.

Note – You can use the `dfshares` command to verify that the resources are available.

NFS Informational Commands

Use the following commands to get information about NFS resources.

The dfshares Command

The `dfshares` command displays the NFS resources currently being shared.

Command Format

```
dfshares [ -F nfs ] [ host ]
```

Without arguments, the `dfshares` command displays shared resources for the local server.

dfshares

RESOURCE	SERVER	ACCESS	TRANSPORT
host1:/usr/share/man	host1	-	-

It is also used to display shared resources by a specified server name.

dfshares host2

RESOURCE	SERVER	ACCESS	TRANSPORT
host2:/export	host2	-	-

The dfmounts Command

This command displays mounted resource information.

Command Format

```
dfmounts [ -F nfs ]
```

Without arguments, the dfmounts command displays the shared resource and clients mounting the resource for the local server.

```
# dfmounts
```

RESOURCE	SERVER	PATHNAME	CLIENTS
-	host1	/usr/share/man	host5,host9

This command is also used to display mounted resource information for a specified server name.

```
# dfmounts host2
```

RESOURCE	SERVER	PATHNAME	CLIENTS
-	host2	/export	host5,host9

The NFS Client

The following commands and files are used with the NFS client.

The mount Command

The `/usr/sbin/mount` command is used to attach either a local or remote file resource to the file system hierarchy.

Command Format

```
mount [ -F nfs ] [ -o options ] server:pathname mount_point
```

Options

The following options can be used with the `mount` command:

- `-F nfs`

Specifies `nfs` as the file system type. This option is not required because `nfs` is the default remote file system type.

- `-o options`

Specifies a comma-separated list of file-system specific options, such as `rw`, to mount the file resource as read, write, and `ro` to mount the file resource read-only. (The default is `rw`.)

- `server:pathname`

Specifies the name of the server and the path name of the remote file resource; these are separated by a colon (:).

- `mount_point`

Specifies the path name of the mount point on the local system (which must already exist).

Accessing a Remote File Resource

Use the `mount` command to access a remote file resource. For example:

```
# mount host1:/usr/share/man /usr/share/man
```

To mount a remote read-only file resource from the first-available host in a comma-separated list of hosts, execute the following:

```
# mount -o ro host1,host2,host3:/usr/share/man /usr/share/man
```

For file systems shared as read-only, if multiple hosts are named and the first server in the list is down, the `failover` utility uses an alternative server in the list to access files.

The /etc/vfstab File

To have remote file resources mounted at boot time, enter the appropriate entries in the client's `/etc/vfstab` file. For example:

#device	device	mount	FS	fsck	mount	mount
#to mount	to fsck	point	type	pass	at boot	options
#						
host1:/usr/share/man	-	/usr/share/man	nfs	-	yes	soft,bg

The fields in the `/etc/vfstab` file include:

- **device to mount**

The name of the server and the path name of the remote file resource; these are separated by a colon (:).

- **device to fsck**

NFS resources are not checked from the client, because the file system is not owned by the client. This field is always dash (-) for NFS resources.

- **mount point**

The default mount point for the file resource.

- **FS type**

Use `nfs` for NFS resources.

- `fsck pass`

NFS resources are not checked from the client, because the file system is not owned by the client. This field is always dash (-) for NFS resources.

- `mount at boot`

Either `yes` or `no`, which indicates whether the file resource should be mounted when the system enters run level 2 or when the `mountall` command is issued, respectively.

- `mount options`

A comma-separated list of mount options.

- `rw|ro`

Specifies whether the resource is mounted as read write or read only. The default is read write.

- `bg|fg`

If the first mount attempt fails, retry in the background or foreground. The default is to retry in the foreground.

- `soft|hard`

During an NFS mount request, the `soft` option returns an error if the server does not respond, then it continues boot. The `hard` option continues to retry the mount until the server responds or the `retry/timeout` values are exceeded. The default is a `hard` mount.

Note – Although `soft` or `bg` are not the default settings, combining these two options usually results in the fastest client boot up when NFS mounting problems arise.

- `intr|nointr`

Indicates keyboard interrupts to kill a process that is hung waiting for a response on a hard-mounted file system. The default is `intr`.

- `suid|nosuid`

Indicates whether to enable `setuid` execution. The default enables `setuid` execution.

- `timeo=n`

Sets `timeout` to *n* tenths of a second. The default `timeout` is 11, measured in one-tenth of a seconds (0.1 second) for User Datagram Protocol (UDP) transports and 600 tenths of a second for TCP.

- `retry=n`

Sets the number of times to retry the mount operation. The default is 10,000 times.

- `retrans=n`

Sets the number of NFS retransmissions to *n*. The default is 5 for UDP. For connection-oriented transports (such as TCP), this option has no effect.

Note – If the file resource is listed in the `/etc/vfstab` file, the superuser can specify either `server:pathname` or `mount_point` on the command line because the `mount` command checks the `/etc/vfstab` file for more information.

Recommended Mounting Options

Mounting a file system with the `bg` option indicates that if the server's `mountd` does not respond, the system's attempt to remount the file system occurs in the background. This prevents the remount from interruptions of other system services.

When the file system is mounted, an NFS request waits the amount of time indicated by the `timeo` field (tenths of a second) for a response. If no response is received, the value in the `timeo` field is doubled and the request is retried.

When the retransmission times reach the value in the `retrans` field, a file system mounted with the `soft` option returns an error. A file system mounted with the `hard` option prints a warning message and continues to retry.

Table 6-1 lists the recommended mounting options for some commonly shared file resources.

Table 6-1 Mount Options

NFS File Resource	Read-write/ Read-only	System Startup	Server Crash	Interrupt	Security
/usr	ro	fg	hard	nointr	suid
/export/home	rw	bg	hard	intr	nosuid
/opt/frame	ro	bg	soft	—	nosuid

A Read-Only Directory

The `/usr` file system contains operating system binaries. This essential file system is mounted in the foreground, the booting process does not continue until the mount is completed.

The NFS client hard mounts this directory. This means the client continues to retry the mount request until the server responds.

A Read-Write Directory

The `/export/home` directory is where the users' login directories are commonly placed. A hard mount is recommended for all read-write (`rw`) file systems (for example, users' home directories).

The `nosuid` option provides additional network security because the `setuid` permissions on NFS resources are ignored.

A Read-Only Application Directory

Nonessential applications are commonly mounted as read only (ro) in the background (bg) with a soft mount. The system continues to boot if the server does not respond during boot. If the server crashes, the mount times out.

The umount Command

Use the `/usr/sbin/umount` command to detach either a local or remote file resource from the file system hierarchy.

Command Format

```
umount server:pathname | mount_point
```

The command line can specify either *server:pathname* or *mount_point*.

```
# umount /usr/share/man
```

The mountall and umountall Commands

Use the `/usr/sbin/mountall` and `/usr/sbin/umountall` commands to mount and unmount all file resources.

The mountall Command

Without any arguments, the `/usr/sbin/mountall` command mounts all file resources listed in the `/etc/vfstab` file with a `mount-at-boot` value of yes.

To limit the action of this command to remote file resources, use the `-r` option.

Command Format

```
mountall -r [ -F nfs ]
```

The `-F nfs` option restricts the action of this command to NFS resources only.

```
# mountall -r
```

The umountall Command

Without any arguments, the `/usr/sbin/umountall` command unmounts all currently mounted file resources. To limit the action of this command to remote file resources, use the `-r` option.

Note – `root ( / )`, `/usr`, `/var`, and all pseudo file systems are not unmounted.

Command Format

```
umountall -r [ -F nfs ]
```

```
# umountall -r
```

Option

The following option can be used with the `umountall` command:

- `-F nfs`

Specifies `nfs` as the file system type. This option is not required, because `nfs` is the default remote file system type.

The NFS Client Setup

To set up an NFS client, complete the following steps:

1. Use the `/usr/sbin/dfshares` command to display a server's available resources.

```
# dfshares host1
```

RESOURCE	SERVER	ACCES	TRANSPORT
host1:/usr/share/man	host1	-	-

2. Use the `/usr/sbin/mount` command to access the remote file resource.

```
# mount host1:/usr/share/man /usr/share/man
```

The `/usr/share/man` directory on the client is the mount point in the local system's file hierarchy. This directory should be empty.

3. Once it has been determined that access to the manual pages located on the remote server is no longer needed, you can unmount the remote file resources from the client by using the `/usr/sbin/umount` command.

```
# umount /usr/share/man
```

Occasionally, an attempt to unmount an NFS file system results in the following error message:

```
nfs mount: /usr/share/man: is busy
```

This usually means that a user or program is accessing the resource.

Mounting Using the /etc/vfstab File

Edit the `/etc/vfstab` file to add an entry for the remote resource that is automatically mounted whenever the system enters run level 3.

```
host1:/usr/share/man - /usr/share/man nfs - yes ro,bg
```

NFS Server Logging

A new feature in the Solaris 8 Operating Environment is NFS server logging. This feature records NFS reads and writes on the file system. The daemon, `nfslogd`, provides this operational logging.

When NFS server logging is enabled, records of all NFS operations on the file system are written into a buffer file by the kernel. This data includes a timestamp, the client IP address, the UID of the requestor, the file handle of the resource that is being accessed, and the type of operation that occurred.

The `nfslogd` daemon converts this raw data into ASCII records that are stored in ASCII log files. During the conversion, the IP addresses are modified to host names and the UIDs are modified to logins.

Mappings of file handles to path names is also handled by `nfslogd`. It keeps track of these mappings in a `file-handle-to-path` mapping table.

One mapping table exists for each tag identified in the `/etc/nfs/nfslog.conf` file.

The file handles are also converted into path names. The daemon keeps track of the file handles and stores information in a separate file handle to path name table, this way the path does not have to be re-identified each time a file handle is accessed.

Note – It is important to keep the `nfslogd` daemon running, because there is no tracking of changes to the mappings in the `file_handle-to-path` table if `nfslogd` is turned off.

Enabling NFS Server Logging

To enable nfs server logging, complete the following steps:

1. Become superuser.
2. Optional: Change the file system configuration settings.

In `/etc/nfs/nfslog.conf`, either edit the default settings for all file systems by changing the data associated with the `global` tag or add a new tag for the specific file system. If these changes are not needed, do not edit this file.

3. Add entries for each file system to be shared using NFS server logging.

Edit `/etc/dfs/dfstab` and add one entry to the file for the file system that is to have NFS server logging enabled.

You must enter the tag used with the `log=tag` option in `/etc/nfs/nfslog.conf`

The following example uses the default settings in the `global` tag:

```
share -F nfs -o ro,log=global /export/ftp
```

4. Check that the nfs service is running on the server.

If the nfs daemons are not running, issue the following commands to kill and restart the nfs daemons.

```
# /etc/init.d/nfs.server stop  
# /etc/init.d/nfs.server start
```

5. If the NFS daemons are already running, issue a command to share the file system.

Once you add the entry to `/etc/dfs/dfstab`, the file system can be shared by either rebooting the system or by using the `shareall` command.

```
# shareall
```

If the NFS daemons were restarted earlier, you do not need to run this command because the script runs the command.

6. Verify that the information is correct.

Run the `share` command to check that the correct options are listed:

```
# share
-      /export/share/man    ro    ""
-      /usr/src             rw=eng ""
-      /export/ftp          ro,log=global ""
```

7. Start the NFS log daemon, `nfslogd`, if it is not running already.

```
# /usr/lib/nfs/nfslogd
```

This step is not necessary if you restarted the `nfs` daemons using the `nfs.server` script, because this script also starts this daemon if the `/etc/nfs/nfslog.conf` file exists.

The /etc/nfs/nfslog.conf File

This file defines the path, file names, and type of logging to be used by `nfslogd`. Each definition is associated with a tag.

Starting NFS server logging requires that you identify the tag for each file system. The global tag defines the default values.

The following is an example of an original `nfslog.conf` file:

```
# cat /etc/nfs/nfslog.conf
#ident  "@(#)nfslog.conf      1.5      99/02/21 SMI"
#
.
.
# NFS server log configuration file.
#
<tag>      [ defaultdir=<dir_path> ] \
           [ log=<logfile_path> ] [ fhtable=<table_path> ] \
[ buffer=<bufferfile_path> ] [ logformat=basic|extended ]
#

global defaultdir=/var/nfs \
log=nfslog fhtable=fhtable buffer=nfslog_workbuffer
```

Use the following parameters with each tag, as needed:

- `defaultdir=path`

Specifies the default directory path for the logging files.

- `log=path/filename`

Sets the path and file name for the log files.

- `fhtable=path/filename`

Selects the path and file name for the `file_handle-to-path` database files.

- `buffer=path/filename`

Determines the path and file name for the buffer files.

- `logformat=basic|extended`

Selects the format to be used when creating user-readable log files. The basic format produces a log file similar to some `ftpd` daemons. The extended format gives a more detailed view.

For the parameters that can specify both the path and the file name, if the path is not specified, the path defined by `defaultdir` is used. Also, you can override `defaultdir` by using an absolute path.

To make identifying the files easier, place the files in separate directories. For example,

```
# cat /etc/nfs/nfslog.conf
#ident "@(#)nfslog.conf      1.5      99/02/21 SMI"
#
.
.
# NFS server log configuration file.
#
global defaultdir=/var/nfs \
        log=nfslog fhtable=fhtable buffer=nfslog_workbuffer
publicftp log=logs/nfslog fhtable=fh/fhtables buffer=buffers/workbuffer
```

You must create the directories for `logs`, `fh`, and `buffers` before starting NFS server logging.

In this example, any file system shared with `log=publicftp` uses the following values:

- The default directory is `/var/nfs`.

- The log files are stored in `/var/nfs/logs/nfslog*`.
- The *file_handle-to-path* database tables are stored in `/var/nfs/fh/fhtables`.
- The buffer files are stored in `/var/nfs/buffers/workbuffer`.

The /etc/default/nfslogd File

NFS operations on an NFS server are logged based on the configuration information defined in `/etc/default/nfslogd`.

This file defines some of the parameters used when using NFS server logging. These parameters include:

- **MAX_LOGS_PRESERVE** – Determines the number of log files to be saved. The default value is 10.
- **MIN_PROCESSING_SIZE** – Sets the minimum number of bytes that the buffer file must reach before processing and writing to the log file. The default value for is 524288 bytes. Increasing this number can improve performance by reducing the number of times the buffer file is processed.

This parameter, along with **IDLE_TIME** determines how often the buffer file is processed.

- **IDLE_TIME** – Sets the number of seconds that `nfslogd` should sleep (wait) before checking for more information in the buffer file. It also determines how often the configuration file is checked. The default value is 300 seconds. Increasing this number can improve performance by reducing the number of checks.
- **CYCLE_FREQUENCY** – Determines the number of hours that must pass before the log files are cycled. The default value is 24 hours. This option is used to prevent the log files from growing too large.
- **UMASK** – Specifies the permissions for the log files that are created by `nfslogd`. The default value is 0137.

Summary of NFS Commands, Files, and Daemons

The main commands and files used on both the server and client systems are summarized in Table 6-2.

Table 6-2 Summary of NFS Commands, Files, and Daemons

	NFS Server	NFS Client
Commands	<code>share resource</code> <code>unshare resource</code> <code>shareall</code> <code>unshareall</code> <code>dfmounts</code> <code>/etc/init.d/nfs.server</code>	<code>mount server:directory \</code> <code>mount-point</code> <code>umount mount-point</code> <code>mountall -r</code> <code>umountall -r</code> <code>dfshares server</code> <code>/etc/init.d/nfs.client</code>
Files	<code>/etc/dfs/fstypes</code> <code>/etc/dfs/dfstab</code> <code>/etc/dfs/sharetab</code> <code>/etc/rmtab</code> <code>/etc/nfs/nfslog.conf</code> <code>/etc/default/nfslogd</code> <code>/etc/nfs/nfslogtab</code>	<code>/etc/dfs/fstypes</code> <code>/etc/vfstab</code> <code>/etc/mnttab</code>
Daemons	<code>/usr/lib/nfs/nfsd</code> <code>/usr/lib/nfs/mountd</code> <code>/usr/lib/nfs/statd</code> <code>/usr/lib/nfs/lockd</code> <code>/usr/lib/nfs/nfslogd</code>	<code>/usr/lib/nfs/statd</code> <code>/usr/lib/nfs/lockd</code>

Troubleshooting NFS Errors

You can discover most NFS problems through console messages or symptoms on a client.

rpcbind Failure Error

Error Message

```
nfs mount: server1:: RPC: Rpcbind failure
RPC: Timed Out
nfs mount: retrying: /mntpoint
```

This message is displayed on the client during the boot process or in response to an explicit mount request. It indicates a problem accessing the server. This error can occur due to the combination of an incorrect Internet address and a correct host or node name in the `hosts` database file supporting the client node.

This error can also occur whenever the `hosts` database file supporting the client is correctly specifying the server node, but the server node is extremely overloaded, temporarily stopped, or crashed.

Solution

Complete the following step:

1. If the server node is operational, determine if the server is out of critical resources (for example, memory, swap, or disk space).

Note – This example was caused by temporarily shutting down the server node and then attempting (through the command line) to have it service an NFS mount request from a client node.

Server Not Responding Error

Error Message

NFS server server2 not responding, still trying

This message is displayed during the boot process or in response to an explicit mount request and indicates a known server that is unreachable.

Solution

Complete the following steps:

1. Check to see if the network between the local system and the server is down by using the `ping` command (`ping server2`).
2. Check to see if the server (`server2`) is down.

NFS Client Fails a Reboot Error

Error Condition

An NFS client fails a reboot without producing an error message.

This error condition is encountered whenever an administrator attempts to restart an NFS client node using an `init 6` or `reboot` command. The client node correctly reboots up to the point where the system echoes:

Setting default interface for multicast: add net 224.0.0.0: gateway:
client_node_name.

The client node does not finish the proper boot sequence and does not generate any error messages.

These symptoms are consistent with a client requesting an NFS mount using an entry in the `/etc/vfstab` file, which specifies a hard mount in the foreground (the default option), to an NFS server that is not operational.

Solution

Complete the following step:

1. If the NFS is available and failing:
 - a. Reset the failed client node and boot it in single-user mode.
 - b. Once in single-user mode, edit the `/etc/vfstab` file so that you comment out the NFS mounts.
 - c. Continue with the boot cycle up to the default run level (normally 3) by pressing `Control-d`.
 - d. Using the information in the `/etc/vfstab` file, determine if all the NFS servers are operational and functioning properly.
 - e. After you have determined which NFS server(s) have failed, and you have resolved any outstanding problems with them, remove the comments placed in the `/etc/vfstab` file.

Note – An alternative to adding the comments to the `/etc/vfstab` file entries can be altering those entries to use the `soft` mount and background activation options.

Stopped Server Error

Error Message

```
nfs mount: dbserver: NFS: Service not responding
nfs mount: retrying: /mntpoint
```

This message is displayed during the boot process or in response to an explicit mount request and indicates a server that is reachable is not running the `nfsd` server daemons.

Solution

Complete the following steps:

1. Use the `who -r` command on the server to see if it is at run level 3. If it is not, change to run level 3 using the `init 3` command.
2. Use the `ps -e` command on the server to check whether the `nfsd` daemon and NFS server daemons are running. If they are not, start them with the `/etc/init.d/nfs.server` script and the `start` keyword.

Program Not Registered Error

Error Message

```
nfs mount: dbserver: RPC: Program not registered
nfs mount: retrying: /mntpoint
```

This message is displayed during the boot process or in response to an explicit mount request and indicates a server that is reachable is not running the `/usr/lib/nfs/mountd` server daemon.

Solution

Complete the following steps:

1. Use the `who -r` command on the server to see if it is at run level 3. If it is not, change to run level 3 using the `init 3` command.

Note – If you used the `shutdown` command to bring the system down to the single-user mode from run level 3, the `who -r` command might be disabled. Rebooting the system re-enables the command.

2. Use the `ps -e` command on the server to see if the `mount` daemon is running. If it is not, start it by invoking the `/etc/init.d/nfs.server` script first with a `stop` flag and then with a `start` flag.
3. Check or verify your `/etc/dfstab` entries.

Stale File Handle Error

Error Message

stale NFS file handle

This message is displayed when a process attempts to access a remote file resource and the file handle is out of date.

Solution

The file resource might have been moved on the server. Complete the following step:

1. Unmount and mount the resource again on the client.

Unknown Host Error

Error Message

nfs mount: sserver1:: RPC: Unknown host

This message indicates that the host name of the server on the client is missing, or not in the hosts table.

Solution

Complete the following step:

1. Determine if the host name in the hosts database supporting the client node is correctly specified.

Note – This example has the node name `server1` misspelled.

Mount Point Error

Error Message

mount: mount-point /DS9 does not exist.

This message is displayed during the boot process or in response to an explicit mount request and indicates a nonexistent mount point.

Solution

Complete the following step:

1. Check that the mount point exists on the client and is spelled correctly on the command line or in the `/etc/vfstab` file; or comment out the entry and reboot.

No Such File Error

Error Message

No such file or directory

This message is displayed during the boot process or in response to an explicit mount request and indicates an unknown file resource name on the server.

Solution

Complete the following step:

1. Check that the directory exists on the server and is spelled correctly on the command line or in the `/etc/vfstab` file.

Exercise: Configuring the NFS Environment



Exercise objective – In this lab, you configure an NFS server and client to share and mount `/usr/share/man`.

Preparation

Choose a partner for this lab, and determine which system will be configured as the NFS server and which will serve as the NFS client. Verify that entries for both systems exist in the `/etc/hosts` file of both systems. Refer to the lecture notes as necessary to perform the steps listed.

Task Summary

In this exercise, you accomplish the following:

- Select a system to act as an NFS server, and perform the steps required to share the `/usr/share/man` directory. Use the following commands to verify that the directory is shared and that no NFS mounts are present on the server:
 - ▼ `share`
 - ▼ `dfshares`
 - ▼ `dfmounts`
- On the NFS client system, rename the `/usr/share/man` directory to `/usr/share/man.orig`. Verify that the man pages are not available. Create a mount point called `/usr/share/man`. Mount the `/usr/share/man` directory from the NFS server. Verify that the man pages are now available.
- On the NFS client, record the default options used for this NFS mount. Verify the list of mounts that the server is providing. Unmount `/usr/share/man`, and again verify the list of remote mounts the server is providing.

- On the NFS server, unshare /usr/share/man. In /etc/dfs/dfstab, change the entry for this directory so that it uses the options -o rw=bogus. Share everything listed in the dfstab file.
- On the NFS client, attempt to mount /usr/share/man from the NFS server. Record what happens.
- On the NFS server, unshare /usr/share/man and remove the entry for it from /etc/dfs/dfstab.
- On the NFS client, return /usr/share/man to its original configuration.

Tasks

Complete the following steps.

On the NFS Server

1. Edit the /etc/dfs/dfstab file. Add an entry to share the directory that holds the manual pages.
`share /usr/share/man`
2. Start the NFS server daemons.
`# /etc/init.d/nfs.server start`
3. Use the following commands to verify that /usr/share/man is shared, and that no NFS mounts are present:
`# share`
`# dfshares`
`# dfmounts`

On the NFS Client

4. Rename the `/usr/share/man` directory so that you can no longer access the manual pages installed on the client system. Verify that the man pages are unavailable.

```
# cd /usr/share
# mv man man.orig
# man ls
```

What message does the `man` command report?

5. Create a new `man` directory for use as a mount point.

```
# mkdir man
```

6. Mount `/usr/share/man` from the server.

```
# mount server:/usr/share/man /usr/share/man
```

7. Verify that the man pages are now available.

```
# man ls
```

Did it work? _____

8. Verify and record the default options used for this mount.

```
# mount
```

9. Verify the list of mounts that the server is providing.

```
# dfmounts server
```

10. Unmount `/usr/share/man`, and again verify the list of remote mounts provided by the server.

```
# umountall -r
# dfmounts server
```

On the NFS Server

11. Unshare the /usr/share/man directory.

```
# unshareall
```

12. Change the share statement in /etc/dfs/dfstab for /usr/share/man so it reads:

```
share -o rw=bogus /usr/share/man
```

13. Share the /usr/share/man directory.

```
# shareall
```

On the NFS Client

14. Attempt to mount /usr/share/man again.

```
# mount server:/usr/share/man /usr/share/man
```

What happens?

On the NFS Server

15. Unshare the /usr/share/man directory.

```
# unshareall
```

16. Edit /etc/dfs/dfstab to remove the entry for the /usr/share/man directory.

On the NFS Client

17. Return /usr/share/man to its original configuration. Verify that the man pages are available again.

```
# cd /usr/share
# rmdir man
# mv man.orig man
# man ls
```

Exercise Summary



Discussion – Take a few minutes to discuss the experiences, issues, or discoveries that you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise: Configuring the NFS Environment

Task Solutions

On the NFS Server

1. Edit the `/etc/dfs/dfstab` file. Add an entry to share the directory that holds manual pages.

```
share /usr/share/man
```

2. Start the NFS server daemons.

```
# /etc/init.d/nfs.server start
```

3. Use the following commands to verify that `/usr/share/man` is shared, and that no NFS mounts are present:

```
# share
```

```
- /usr/share/man ro ""
```

```
# dfshares
```

RESOURCE	SERVER	ACCESS	TRANSPORT
server:/usr/share/man	server	-	-

```
# dfmounts
```

```
#
```

No output for `dfmounts`.

On the NFS Client

1. Rename the `/usr/share/man` directory so that you can no longer access the manual pages installed on the client system. Verify that the man pages are unavailable.

What message does the `man` command report?

No manual entry for `ls`.

2. Create a new `man` directory for use as a mount point.

```
# mkdir man
```

3. Mount /usr/share/man from the server.

```
# mount server:/usr/share/man /usr/share/man
```

4. Verify that the man pages are now available.

```
# man ls
```

Did it work? Yes

5. Verify and record the default options used for this mount.

Read and write (rw)

6. Verify the list of mounts that the server is providing.

```
# dfmounts server
RESOURCE      SERVER  PATHNAME      CLIENTS
-             server  /usr/share/man  client
```

7. Unmount /usr/share/man, and again verify the list of remote mounts provided by the server.

```
# umountall -r
# dfmounts server
#
```

No output from the dfmounts command indicates that there are no clients mounting file systems from the server.

On the NFS Server

8. Unshare the /usr/share/man directory.

```
# unshareall
```

9. Change the share statement in /etc/dfs/dfstab for /usr/share/man so it reads:

```
share -o rw=bogus /usr/share/man
```

10. Share the /usr/share/man directory.

```
# shareall
```


On the NFS Client

1. Attempt to mount /usr/share/man again.

What happens?

The client reports the error message:

```
nfs mount: server:/usr/share/man: Permission denied
```

On the NFS Server

2. Unshare the /usr/share/man directory.
unshareall
3. Edit /etc/dfs/dfstab to remove the entry for the /usr/share/man directory.

On the NFS Client

4. Return /usr/share/man to its original configuration. Verify that the man pages are available again.

```
# cd /usr/share  
# rmdir man  
# mv man.orig man  
# man ls
```

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ Describe the functions of an NFS server and an NFS client
- ☐ Make resources available and unavailable for mounting
- ☐ Edit the `/etc/dfs/dfstab` file on an NFS server to enable automatic sharing of resources
- ☐ Display a server's available resources for mounting
- ☐ Mount a resource from another system
- ☐ Edit the `/etc/vfstab` file to mount resources on an NFS client
- ☐ Describe the function of these commands: `mountall`, `umountall`, `shareall`, and `unshareall`
- ☐ Describe and configure NFS logging

Objectives

Upon completion of this module, you should be able to:

- List the benefits of using the automount utility
- Describe the purpose of each of the types of automount maps
- Configure the `auto_master` map to specify which direct, indirect, and special maps the `automountd` daemon reads
- Create the `auto_direct` map with the full path names and mount options for automatically mounting remote file resources
- Modify the `auto_home` map as an example of an indirect map providing a consistent view of home directories across the network, regardless of where the user is logged in

Additional Resources



Additional resources – The following references provide additional details on the topics discussed in this module:

- *System Administration Guide, Volume I*, Sun Part Number 805-7228-10
- *System Administration Guide, Volume II*, Sun Part Number 805-7229-10
- *System Administration Guide, Volume III*, Sun Part Number 806-0916-10

AutoFS Overview

The AutoFS file system enables you to do the following:

- Mount file systems on demand
- Unmount file systems automatically.

Automounted resources remain mounted for as long as they are being used. If no files or directories within the file system are accessed within a specified time-out period, a file system unmount automatically occurs.

- Centralize the administration of AutoFS mounts through the use of a name service.
- Have multiple mount resources for read-write file systems.

AutoFS Components

AutoFS contains three components that work together on the client to accomplish automatic mounting. These components include an AutoFS file system, the automount command, and the automountd daemon.

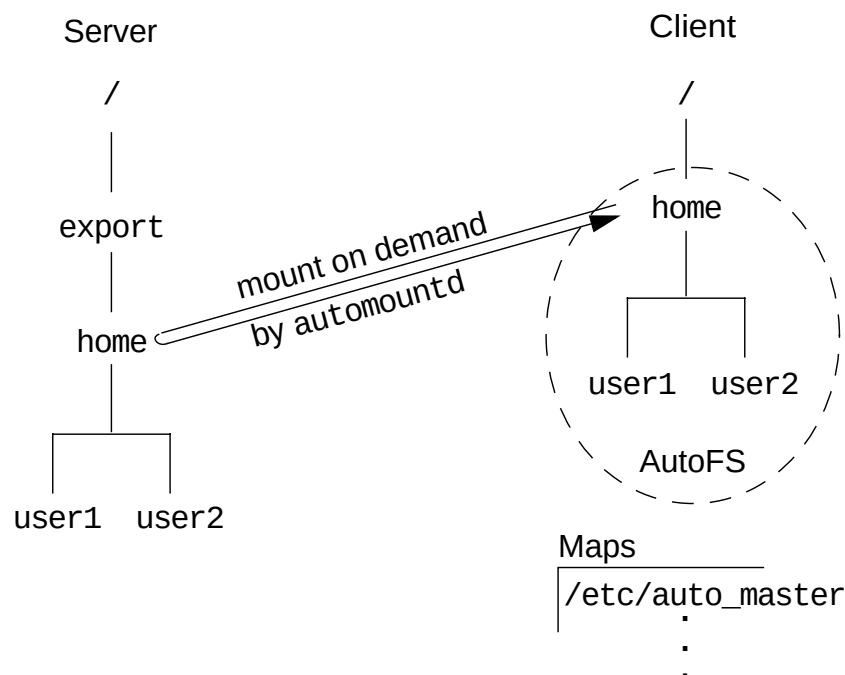


Figure 7-1 AutoFS Components

The following list describes the components:

- The **autofs** file system – An **autofs** file system's mount points are defined in automount maps located in the `/etc` directory on the client system. Once the **autofs** mount points are set up, activity under the mount points can trigger file systems to be mounted under them. If automount is configured, the **autofs** file system monitors mount requests made on the client. If a mount request is made for an **autofs** resource not currently mounted, **autofs** calls the **automountd** daemon, which actually mounts the requested resource.
- The **automount** command – The **automount** command, called at system startup time, reads the master map file `/etc/auto_master` to create the initial set of **autofs** mounts. These **autofs** mounts are not automatically mounted at startup time. They are points under which file systems are mounted on demand.

- The automountd daemon – The automountd daemon is started at boot time from the `/etc/init.d/autofs` script and mounts file systems on demand.

The automountd daemon is completely independent from the `automount` command. Because of this separation, you can add, delete, or change map information without having to stop and start the automountd daemon process. However, the process might have to reread the maps.

Automount Maps

The `autofs` files, called maps, identify the file system resources to be automatically mounted. These map types include:

- Master map – Read by the `automount` command during boot up. This map lists the other maps used for establishing the `autofs` file system.
- Direct map – Lists the mount points as absolute path names. This map explicitly indicates the mount point on the client.
- Indirect map – Lists the mount points as relative path names. This map uses a relative path to establish the mount point on the client.
- Special – Provides access to entries in `/etc/hosts` or the Federated Naming Service (FNS).

`automount` maps contain ASCII data files or NIS or NIS+ database files. Together, these maps describe information similar to the information specified in the `/etc/vfstab` file for remote file resources.

Master Maps

The `auto_master` file associates a mount point with a map. It is a master list specifying all of the maps that `autofs` should check. Names of direct and indirect maps listed here refer to files in `/etc`. The following example shows what an `auto_master` file could contain:

```
# cat /etc/auto_master
# Master map for automounter
#
+auto_master
/net          -hosts          -nosuid,nobrowse
/-           auto_direct
/home        auto_home      -nobrowse
/xfn         -xfn
```

The following describes the fields in this example of a master map file:

<i>mount_point</i>	The full path name of a directory. If the directory does not exist, <code>autofs</code> creates one if possible.
<i>map_name</i>	The name of a direct or indirect map. These maps are directions to mounting information.
<i>mount-options</i>	The general options for the map. The mount options are the same as those for standard NFS mounts.

Note – The plus (+) symbol at the beginning of the `+auto_master` line in this file directs the automounter program to look at the NIS or NIS+ databases. If this line is commented out, only the local files are used.

Special Maps

There are two mount points for special maps listed in this `/etc/auto_master` file. They identify the following:

- The `-hosts` map

This special map provides access to all resources shared by each NFS server listed in the hosts database. You can obtain this information from `/etc/inet/hosts`, NIS, NIS+, or DNS. This is a default entry. Shared resources associated with this map are mounted below `/net/hostname`.

- The `-xfn` map

This special map provides access to resources available through the X/Open Federated Naming Service. Resources associated with this service mount below `/xfn`.

Direct Map Entries

The `/-` entry in the example master map defines a mount point for direct maps. This mount point is a pointer that informs the automounter program that the full path names are defined in the file specified by *map_name* (`/etc/auto_direct` in this example).

Note – This is not an entry in the default master map. It has been added here as an example. The other entries in this example already exist in the `auto_master` file.

Note – A NIS or NIS+ `auto_master` map can have only one direct map entry. An `auto_master` map that is a local file can have any number of entries.

Indirect Map Entries

The `/net`, `/home`, and `/xfs` entries define mount points for indirect maps. The maps `-hosts`, `auto_home`, and `-xfs` list relative path names only. Indirect maps get the initial path of the mount point from the master map.

The Solaris 2.6 Operating Environment and later releases support browsing of indirect maps (and special maps) with the `-browse` option. This allows all of the potential mount points to be visible, regardless of whether they are mounted. The `-nobrowse` option disables the browsing of indirect maps. Therefore, in this example, the `/home` automount point provides no browser functions for any directory other than those that are currently mounted. The default for this option is `-browse`.

Direct Maps

Direct maps specify the absolute path name of the mount point, the specific options for this mount, and the shared resource to mount. For example:

```
# cat /etc/auto_direct
# Superuser-created direct map for automounter
#
/apps/frame      -ro,soft  server1:/export/framemaker,v4.0
/opt/local       -ro,soft  server2:/export/unbundled
/usr/share/man    -ro,soft  server3,server4,server5:/usr/share/man
```

The following describes the syntax for direct maps:

key [*mount-options*] *location*

- *key* – The full path name of the mount point for direct maps.
- *options* – The specific options for a given entry.
- *location* – The location of the file resource specified in *server:pathname* notation.

The following direct map entry specifies that the client mounts the `/usr/share/man` directory as read only from the servers `server3`, `server4`, or `server5`, as available:

```
/usr/share/man  -ro          server3,server4,server5:/usr/share/man
```

This entry uses a special notation, a comma-separated list of servers, to specify a powerful automounter feature—multiple locations for a file resource.

Note – The comma-separated list of servers automount feature works *only* with servers that are sharing files as read-only.

The `automountd` command automatically mounts `/usr/share/man` as needed, from `server3`, `server4`, or `server5`, with server proximity being the factor on server selection. If the nearest server fails to respond within the prescribed time-out period, the server with the next nearest proximity is selected.

Indirect Maps

Indirect maps have relative paths in the key field. The first part of the path name is specified in the master map. Indirect maps are useful when you want to mount many remote file resources below a common directory.

The auto_home Indirect Map

The `auto_home` indirect map provides a consistent view of home directories across the network, regardless of which system a user is currently logged in to. For example:

```
# cat /etc/auto_home
# Home directory map for automounter
#
+auto_home
stevenu      host5:/export/home/stevenu
johnnyd      host6:/export/home/johnnyd
wkd          server1:/export/home/wkd
```

Note – The plus (+) symbol at the beginning of the `+auto_home` line in this file directs the automounter to look at the NIS or NIS+ databases. If this line is commented out, only the local files are used.

The following describes the syntax for indirect maps:

key [*mount-options*] *location*

- *key* – The path name of the mount point relative to the beginning of the path name specified in the `/etc/auto_master` file.
- *options* – The specific options for a given entry.
- *location* – The location of the file resource specified in *server:pathname* notation.

The Substitution String for an Indirect Map

The following entry reduces the `auto_home` file to a single line. The use of substitution characters specifies that for every login ID, the client remotely mounts the `/export/home/loginID` directory from the NFS server `server1` onto the local mount point `/home/loginID`.

*

`server1:/export/home/&`

This entry uses the wildcard character (`*`) to match any key; the substitution character (`&`) at the end of the location specification is replaced with the matched key field. This works only when all home directories are on a single server (in this example, `server1`).

The automount Command

When making changes to the master map or creating a direct map, make the change effective by running the `automount` command.

Command Format

```
automount [ -t duration ] [ -v ]
```

The following describes the `automount` command options.

- `-t duration`

Specifies a time, in seconds, that the file system remains mounted when not in use. The default is 10 minutes.

- `-v`

Displays output as the `automount` command executes

You do not have to stop and restart the `automountd` daemon after making changes to existing entries in either a direct or indirect map because it is stateless. You can modify existing entries in both direct and indirect maps at any time. The new information is used when the `automountd` daemon next uses the map entry to perform a mount.

You can modify the master map entries or add entries for new maps. However, you must run the `automount` command to make these changes take effect.

A modification is a change to options or resources. A change to the key (the mount point) or a completely new line is considered to be an added or deleted entry or both.

Use Table 7-1 to determine whether you should run (or re-run) the automount command.

Table 7-1 Using Automount Maps

Automount	Run if Entry is Added or Deleted	Run if Entry is Modified
auto_master	Yes	Yes
direct map	Yes	No
indirect map	No	No

Note – There is no harm in running the automount command to rescan the maps, even if it is not required.

The Client autofs File System

The autofs file system is a kernel file system that supports automatic mounting and unmounting. When you make a request to access a file system at an autofs mount point, the following occurs:

1. The autofs file system intercepts the request.
2. The autofs file system sends a message to the automountd daemon for the requested file system to be mounted.
3. The automountd daemon locates the file system information in a map and performs the mount.
4. The autofs file system allows the intercepted request to proceed.
5. The automountd daemon unmounts the file system after a period of inactivity.

Note – Mounts managed through the autofs service should not be manually mounted or unmounted. Even if the operation is successful, the autofs file system does not check that the object has been unmounted, resulting in possible inconsistency. A reboot clears all of the autofs mount points.

Multi-threaded autofs

The new autofs automount daemon is now fully multi-threaded. This enables concurrent servicing of multiple mount requests and increases reliability.

Automount Administration

The following procedure describes how to set up remote access to a resource in the Solaris Operating Environment that is located on an NFS server, using the automountd daemon.

Setting up a Direct Map

This example demonstrates how to set up remote access to the man pages located on an NFS server.

1. Edit the `/etc/auto_master` file to add a direct map entry.

```
# Master map for automounter
#
+auto_master
/net          -hosts          -nosuid,nobrowse
/home        auto_home        -nobrowse
/-           auto_direct
/xfn         -xfn
```

2. Create a new file called `/etc/auto_direct` and add the following entry for the directory you want to automount. Replace *server* with the host name of your server.

```
/usr/share/man    -ro          server:/usr/share/man
```

3. Make the changes effective.

```
# automount -v
automount: /usr/share/man mounted
automount: no unmounts
```


Setting up an Indirect Map

Complete the following steps to set up an indirect map:

1. Edit the `/etc/auto_master` file. Add the patch directory and map.

```
# Master map for automounter
#
+auto_master
/net          -hosts          -nosuid,nobrowse
/home        auto_home       -nobrowse
/services    auto_patch
/xfn         -xfn
```

2. Create an `/etc/auto_patch` map. Enter the patch directory name on the client and the *server:pathname*. The following example illustrates the indirect map content:

```
# Patch directory map for automounter
#
patch        server1:/export/patch
```

3. Make the changes effective.

```
# automount -v
```

Exercise: Using the Automounter



Exercise objective – In this lab, you use the automounter processes to automatically mount manual pages and a user's home directory.

Preparation

Choose a partner for this lab, and determine which system will be configured as the NFS server and which will serve as the NFS client. Verify that entries for both systems exist in the `/etc/hosts` file of each systems. Refer to the lecture notes as necessary to perform the steps listed.

Task Summary

In this exercise, you accomplish the following:

- On the server, perform the steps required to share `/usr/share/man`.
- On the client, rename `/usr/share/man` to `/usr/share/man.orig`, and create a new mount point for the `/usr/share/man` directory. Edit the master map so it calls for a direct map. Edit the direct map to mount `/usr/share/man` from the server. Use `automount` to update the automounter. Test that the man pages work and verify the mount that occurs.
- Create a new identical user on both the server and client that uses `/export/home/username` for the user's home directory. On both systems, make the changes required in `/etc/passwd` to set the home directory for this new user to the `/home/username` directory.
- On the server, perform the steps required to share the `/export/home` directory.

- On both systems, make the changes required in the `/etc/auto_home` to allow both systems to automatically mount `/export/home/username` when the new user calls for the `/home/username` directory. Test the new user login on both systems and verify the mounts that happen. Log in as `root` when finished.
- On the server, unshare `/export/home` and `/usr/share/man` and remove entries for these directories from the `/etc/dfs/dfstab` file. Stop the NFS server daemons.
- On the client, remove the direct map entry from `/etc/auto_master` and update the automounter with the change. Return `/usr/share/man` to its original configuration.

Tasks

Complete the following steps:

On the Server Host

1. Edit the `/etc/dfs/dfstab` file and add the following line to share the manual pages:

```
share /usr/share/man
```

2. Use `pgrep` to check if the `mountd` daemon is running.

```
# pgrep -xl mountd
```

If the `mountd` daemon is not running, start it.

```
# /etc/init.d/nfs.server start
```

If the `mountd` daemon was already running, share the new directory by using the following command:

```
# shareall
```

On the Client Host

3. Rename the `/usr/share/man` directory so you can no longer access the manual pages installed on the client system.

```
# cd /usr/share/  
# mv man man.orig
```

4. Make a new `man` directory for use as a mount point.

```
# mkdir man
```

5. Change to the `/etc` directory.

```
# cd /etc
```

6. Edit `/etc/auto_master` to add the following line for a direct map:

```
/-      auto_direct
```

7. Use `vi` to create a new file called `/etc/auto_direct` and add the following line to it:

```
/usr/share/man          server:/usr/share/man
```

8. Run the `automount` command to update the list of directories managed by the automounter.

```
# automount -v
```

9. Test the configuration and verify that a mount for `/usr/share/man` exists after accessing the manual pages.

```
# man ls
```

```
# mount | grep man
```

Did it work? _____

On the Server Host

10. Verify that the `/export/home` directory exists. If it does not, create it.

```
# ls /export/home
```

```
# mkdir /export/home
```

11. Run `Admintool`.

```
# admintool &
```

12. From the Edit menu, select Add. In the Add User form, complete the following information:

- a. User Name: Your choice
- b. User Id: 3001
- c. Primary Group: Leave as 10
- d. Secondary Groups: Leave blank
- e. Comment: Your choice
- f. Login Shell: Your choice
- g. Password: Select Normal Password: Set the password to **cangetin**

Skip the password aging lines.

- a. Create Home directory: Leave as selected (do not click on the box)
- b. Path: **/export/home/username**

13. Click OK.

This creates a user on this system along with a home directory for the user (physically on this system).

14. Quit Admintool.

On the Client Host

15. Verify that the /export/home directory exists. If it does not, create it.

```
# ls /export/home
# mkdir /export/home
```

16. Run Admintool.

```
# admintool&
```

17. From the Edit menu, select Add. In the Add User form, complete the following information:

- a. User Name: Same user
- b. User Id: Same UID as the server
- c. Primary Group: Same GID as the server
- d. Secondary Groups: Leave blank
- e. Comment: Your choice
- f. Login Shell: Same shell

Password: Select Normal Password Set the password to **cangetin**

Skip the password aging lines.

- g. Create Home directory: Deselect it (click on the box)
- h. Path: **/export/home/username**

18. Click OK.

You are warned that the home directory does not exist.

19. Click OK.

This adds a user to this system but does not create a home directory. This is correct. Eventually you will use the **automount** command to mount the home directory from the server system.

20. Quit Admintool.

On Both Systems

21. Edit the **/etc/passwd** file and change the home directory for the new user from **/export/home/username** to **/home/username**. Replace *username* with the name of your new user.

22. Edit the **/etc/auto_home** file. Add the following line and replace *username* with the name of your new user:

username **server:/export/home/username**

On the Server Host

23. Edit the `/etc/dfs/dfstab` file and add the following line to share the `/export/home` directory:

```
share /export/home
```

24. Use the `pgrep` command to check if the `mountd` daemon is running.

```
# pgrep -xl mountd
```

If the `mountd` daemon is not running, start it.

```
# /etc/init.d/nfs.server start
```

If the `mountd` daemon is running, share the new directory.

```
# share /export/home
```

On Both Systems

25. Log in as the new user.

Do both systems automatically mount the new user's home directory? _____

What directory is mounted, and what is the mount point:

▼ On the server?

▼ On the client?

26. If the user's home directory is not mounted automatically, as `root`, stop and restart the `automountd` daemon.

```
# /etc/init.d/autofs stop  
# /etc/init.d/autofs start
```

27. On both systems, log out from the current user session and log back in as `root`.

On the Server Host

28. Unshare all shared directories. Edit the `dfstab` file and remove the lines for the `/export/home` and `/usr/share/man` directories.

```
# unshareall
```

29. In preparation for subsequent labs, stop the NFS server daemons.

```
# /etc/init.d/nfs.server stop
```

On the Client Host

30. Edit the `/etc/auto_master` file and remove the line that reads:

```
/-    auto_direct
```

31. Delete the `/etc/auto_direct` file.

```
# rm /etc/auto_direct
```

32. Run the `automount` command to update the list of directories managed by the automounter.

```
# automount -v
```

33. Remove the `/usr/share/man` mount point.

```
# cd /usr/share  
# rmdir man
```

34. Rename the `/usr/share/man` directory so you again have access to the manual pages installed on the client system.

```
# mv man.orig man
```


Exercise Summary



Discussion – Take a few minutes to discuss the experiences, issues, or discoveries that you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Task Solutions

On the Server Host

1. Edit the `/etc/dfs/dfstab` file and add the following line in order to share the manual pages:

```
share /usr/share/man
```

2. Use `pgrep` to check if `mountd` is running.

```
# pgrep -xl mountd
```

If `mountd` is not running, start it.

```
# /etc/init.d/nfs.server start
```

If `mountd` was already running, share the new directory by using the following command:

```
# shareall
```

On the Client Host

3. Rename the `/usr/share/man` directory so you can no longer access the manual pages installed on the client system.

```
# cd /usr/share/  
# mv man man.orig
```

4. Make a new `man` directory for use as a mount point.

```
# mkdir man
```

5. Change to the `/etc` directory.

```
# cd /etc
```

6. Edit `/etc/auto_master` to add the following line for a direct map:

```
/-      auto_direct
```

7. Use `vi` to create a new file called `/etc/auto_direct` and add the following line to it:

```
/usr/share/man      server:/usr/share/man
```

8. Run the automount command to update the list of directories managed by the automounter.

```
# automount -v
```

9. Test the configuration and verify that a mount for /usr/share/man exists after accessing the manual pages.

```
# man ls
```

```
<-- output from man command -- >
```

```
# mount | grep man
```

```
/usr/share/man on server:/usr/share/man read/write/remote on Fri Aug 13  
16:56:14 1999
```

Did it work?

This should automatically mount the directory where the manuals are stored. The man command should work.

On the Server Host

10. Verify that the /export/home directory exists. If it does not, create it.

```
# ls /export/home  
# mkdir /export/home
```

The /export/home directory should have been created during installation of the operating system. If not, the second command applies.

11. Run Admintool.

```
# admintool &
```

The admintool window is displayed.

12. From the Edit menu, select Add. In the Add User form, complete the following information:
 - a. User Name: Your choice
 - b. User Id: 3001
 - c. Primary Group: Leave as 10
 - d. Secondary Groups: Leave blank
 - e. Comment: Your choice
 - f. Login Shell: Your choice
 - g. Password: Select Normal Password: Set the password to **cangetin**

Skip the password aging lines.

 - h. Create Home directory: Leave as selected (do not click on the box)
 - i. Path: **/export/home/username**

13. Click OK.

This creates a user on this system along with a home directory for the user (physically on this system).

14. Quit Admintool.

On the Client Host

15. Verify that the /export/home directory exists. If it does not, create it.

```
# ls /export/home
# mkdir /export/home
```

The /export/home directory should have been created during installation of the operating system. If not, the second command applies.

16. Run Admintool.

```
# admintool &
```

17. From the Edit menu, select Add. In the Add User form, complete the following information:

- a. User Name: Same user
- b. User Id: Same UID as the server
- c. Primary Group: Same GID as the server
- d. Secondary Groups: Leave blank
- e. Comment: Your choice
- f. Login Shell: Same shell
- g. Password: Select Normal Password: Set the password to **cangetin**

Skip the password aging lines.

- a. Create Home directory: Deselect it (click on the box)
- b. Path: **/export/home/username**

On the Client Host

18. Click OK.

You are warned that the home directory does not exist.

19. Click OK.

This adds a user to this system but does not create a home directory. This is correct. Eventually you will use the **automount** command to mount the home directory from the server system.

20. Quit Admintool.

On Both Systems

21. Edit the **/etc/passwd** file and change the home directory for the new user from **/export/home/username** to **/home/username**. Replace *username* with the name of your new user.

22. Edit the **/etc/auto_home** file. Add the following line and replace *username* with the name of your new user:

username **server:/export/home/username**

On the Server Host

23. Edit the `/etc/dfs/dfstab` file and add the following line to share the `/export/home` directory:

```
share /export/home
```

24. Use the `pgrep` command to check if the `mountd` daemon is running.

```
# pgrep -xl mountd
391 mountd
```

If the `mountd` daemon is not running, start it.

```
# /etc/init.d/nfs.server start
```

Once the `mountd` daemon is running, share the new directory.

```
# share /export/home
#
```

On Both Systems

25. Try to log in as the new user.

Do both systems automatically mount the new user's home directory?

Yes, this should work.

What directory is mounted, and what is the mount point:

- ▼ On the server?

```
/home/username on /export/home/username
```

- ▼ On the client?

```
/home/username on server:/export/home/username
```

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ List the benefits of using the automount utility
- ☐ Describe the purpose of each of the types of automount maps
- ☐ Configure the `auto_master` map to specify which direct, indirect, and special maps the `automountd` daemon reads.
- ☐ Create the `auto_direct` map with the full path names and mount options for automatically mounting remote file resources.
- ☐ Modify the `auto_home` map as an example of an indirect map providing a consistent view of `home` directories across the network, regardless of where the user is logged in.

Objectives

Upon completion of this module, you should be able to:

- Describe the CacheFS file system
- Use the appropriate commands to configure the CacheFS file system
- Use the appropriate commands to check the status and consistency of the CacheFS file system
- Set up CacheFS file system logging
- Describe the steps necessary to perform a check of the CacheFS file system
- List the steps to dismantle and delete a CacheFS file system

Additional Resources



Additional resources – The following references provide additional details on the topics discussed in this module:

- *System Administration Guide, Volume I*, Sun Part Number 805-7228-10
- *System Administration Guide, Volume II*, Sun Part Number 805-7229-10
- *System Administration Guide, Volume III*, Sun Part Number 806-0916-10

CacheFS File System

You can use the CacheFS file system to improve the performance of remote file systems (such as NFS), or slow devices, such as CD-ROM drives.

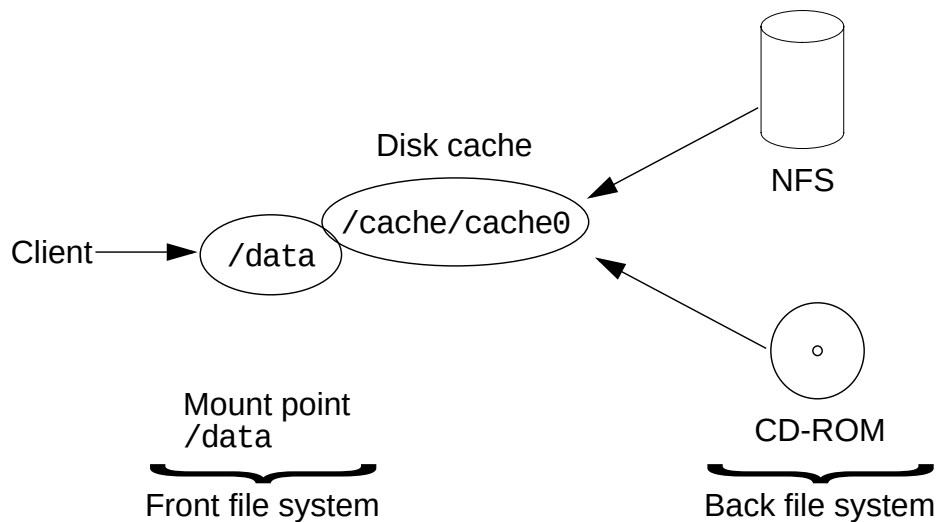


Figure 8-1 CacheFS Diagram

When you enable the CacheFS file system, the data read from the remote file system or CD-ROM is stored in a disk-based cache on the local system. Subsequent read requests to the same data are fulfilled by the local cache, which improves read performance.

Note – This has no effect on the NFS server; it affects only the client.

Using CacheFS Terminology

The following terms are used when discussing the CacheFS file system:

- Back file system – The original disk-based, network-based, or CD-ROM-based file system that is mounted as a CacheFS file system and cached.
- Front file system – The mounted file system that is cached and accessed by the user through the local mount point.
- Consistency – Refers to the state of synchronization between the back and front file systems.

Using CacheFS File System Commands

You have the following commands available for administering a CacheFS file system:

- `cfsadmin` – Administers the disk space for the cached file system. This includes creating, deleting, and listing the contents of the cache.
- `cachefsstat` – Provides statistics on cache usage.
- `cachefslog` – Establishes a login procedure for the cache.
- `cachefswssize` – Helps determine the working set sizes for CacheFS file systems so that the cache area can be properly sized. This functions only if CacheFS logging is enabled.

Creating a CacheFS File System

Setting up a CacheFS file system is basically a three-step procedure, illustrated in Figure 8-2.

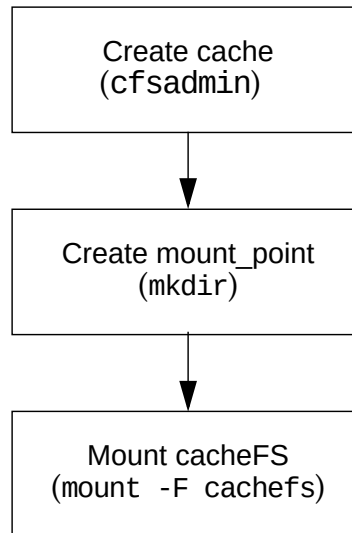


Figure 8-2 Setting up a CacheFS File System

This example procedure assumes that a file system (`/export/data`) on a remote system (`host1`) is available to the local system. The local system mounts the resource and caches it.

Note – All commands are executed on the local system.

To set up a CacheFS file system, perform the following steps:

1. Create a cache using the following command:

```
# cfsadmin -c /cache/cache0
```

2. If one does not already exist, create a local mount point in preparation for mounting the remote file system.

```
# mkdir /data
```

3. Mount the remote file system and implement a CacheFS file system.

```
# mount -F cachefs -o backfstype=nfs,cachedir=/cache/cache0,\
cacheid=data_cache host1:/export/data /data
```

- ▼ The remote resource is mounted as a CacheFS file system.
- ▼ The source file system type (backfstype) is nfs.
- ▼ You must specify the cachedir.
- ▼ The cacheid is optional, but it can provide a convenient user-defined label to identify this CacheFS mount for subsequent administration commands.
- ▼ The remote resource is host1:/export/data and local users access it through the /data mount point.

4. Use the mount command to verify the mount.

```
# mount
/ on /dev/dsk/c0t0d0s0 read/write/setuid/largefiles on Thu May 11
15:55:34 2000
/proc on /proc read/write/setuid on Thu May 11 15:55:34 2000
/dev/fd on fd read/write/setuid on Thu May 11 15:55:34 2000
/tmp on swap read/write on Thu May 11 15:55:35 2000

/data
on /cache/cache0/.cfs_mnt_points/host1:_export_data backfstype=nfs/cached
ir=/cache/cache0/cacheid=data_cache on Thu May 11 06:38:43 2000
```

You could automate this mount by adding a line similar to the following to the /etc/vfstab file:

```
host1:/export/data - /data cachefs - yes
backfstype=nfs,cachedir=/cache/cache0,cacheid=data_cache
```

The cache is now used when local users access the resource through the local /data mount point.

CacheFS Cache Directory Details

Figure 8-3 illustrates areas of the underlying cache directory hierarchy:

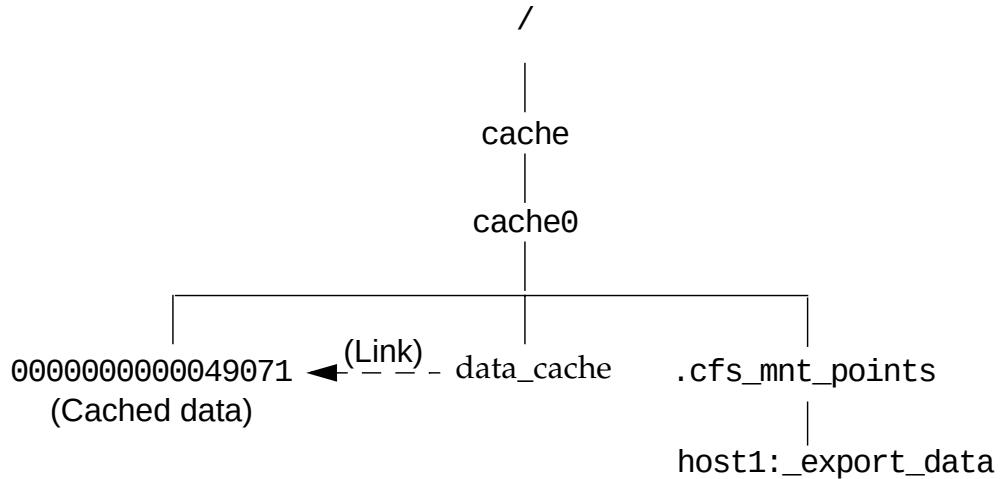


Figure 8-3 CacheFS Cache Directory Structure

The following describes the cache directory hierarchy:

- For each CacheFS file system being cached in the cache directory, an entry is made in the `.cfs_mnt_points` directory.
- If you specify a cache ID string when the CacheFS file system is mounted, this string becomes a symbolic link to the cache data for that file system.
- Local users access the cached data through the local `/data` mount point.

CacheFS Statistics and Consistency Checking

To view CacheFS file system statistics you can use the `cachefsstat` command. To check consistency, use the `cfsadmin` command.

The cachefsstat Command

You use `cachefsstat` command to display cache statistics. It displays information that describes the effectiveness of your cache.

The following output shows statistics for a newly created cache:

```
# cachefsstat /data
```

```
/data
cache hit rate:      100% ( 8 hits, 0 misses)
consistency checks:  24 (24 pass, 0 fail)
modifies:           0
garbage collection:  0
```

By default, automatic cache consistency is enabled. The files in the cache are checked against the originals in the back file system (on the server) and updates are performed on the client's front file system. The `pass` value (24 in this example) indicates the number of consistency checks performed. The `fail` value (0 here) indicates the number of updates that have been performed.

The value in the `hit` rate indicates the efficiency of the cache. The `hits` indicate the instances when the cached file was used and access to the original file avoided. The `miss` value indicates when there was not a cached copy of the file and the back file system copy was accessed.

To collect status over a specific period of time, you can first zero the `cachefs` counters. To zero all `cachefs` counters, use the following command:

```
# cachefsstat -z
```

The demandconst Option

You can disable the automatic consistency checks by using the `demandconst` option for the `mount` command (see the `mount_cacheefs(1)` man page for more information). This should be done only when the back file system is static or the back and front file systems do not need to be synchronized. For example, if the back file system is a read-only file system on a CD-ROM, there is no need to enable consistency checking.

The cfsadmin Command

If automatic consistency checking is disabled by using the `demandconst` option, the following `cfsadmin` command manually invokes a consistency check and performs any necessary updates:

```
# cfsadmin -s /data
```


Enhancing CacheFS File System Caching

You can have additional control of the caching mechanism for CacheFS file systems by doing the following:

- Set the number of data blocks used by the cache as a percentage of the front file system. Refer to the `cfsadmin(1M)` man page for details.

Note – These percentages can be enforced only if the CacheFS file subsystem is given exclusive access to the front file system.

- Set the minimum and maximum number of files that the CacheFS file system can use as a percentage of the files in the front file system.

The following display of cache statistics shows the default values:

```
# cfsadmin -l /cache/cache0
cfsadmin: list cache FS information
maxblocks      90%
minblocks      0%
threshblocks   85%
maxfiles       90%
minfiles       0%
threshfiles    85%
maxfilesize    3MB
data_cache
```

where:

- ▼ **maxblocks** – Maximum amount of storage space that CacheFS can use, expressed as a percentage of the total number of blocks in the front file system.
- ▼ **minblocks** – Minimum amount of storage space (expressed as a percentage of the total number of blocks in the front file system) that CacheFS is always allowed to use without limitation by its internal control mechanisms.
- ▼ **threshblocks** – A percentage of the total blocks in the front file system beyond which CacheFS cannot claim resources once its block usage has reached the level specified by **minblocks**.

- ▼ **maxfiles** – Maximum number of files that CacheFS can use, expressed as a percentage of the total number of inodes in the front file system.
- ▼ **minfiles** – Minimum number of files (expressed as a percentage of the total number of inodes in the front file system) that CacheFS is always allowed to use without limitation by its internal control mechanisms.
- ▼ **threshfiles** – A percentage of the total inodes in the front file system beyond which CacheFS cannot claim inodes once its usage has reached the level specified by **minfiles**.
- ▼ **maxfilesize** – Largest file size (expressed in Mbytes) that CacheFS is allowed to cache.

These parameters are specified with the **-o** option, with multiple parameters separated by commas. Refer to the **cfsadmin (1M)** man page for more details.

You can change these parameters only when you create the cache.

Sizing the Cache

You use the `cachefswssize` command to determine the current size of the data in the cache. This includes the amount of cache space needed for each file system that was mounted under the cache, as well as a total. Before using the `cachefswssize` command, you must enable `cachefs` logging. Before you enable the `cachefs` logging, you must create the directory for the log files.

```
# mkdir /var/cache/logs
```

The following command creates and begins a `cachefs` log:

```
# cachefslog -f /var/cache/logs/data.log /data  
/var/cache/logs/data.log: /data
```

In the previous command, a cache log called `/var/cache/logs/data.log` was created for the CacheFS file system mounted locally as `/data`.

You can change the cache log at any time. The following command is an example of changing the log file to:
`/var/cache/logs/data_new.log`:

```
# cachefslog -f /var/cache/logs/data_new_062100.log /data  
/var/cache/logs/data_new_062100.log: /data
```

At any time, you can determine the current log file. The following is a sample command that describes how to do this:

```
# cachefslog /data  
/var/cache/logs/data_new_062100.log: /data
```

You can stop and verify logging using the following commands:

```
# cachefslog -h /data
not logged: /data
# cachefslog /data
not logged: /data
```

Once you enable logging, you can check the size of the cache.

```
# cachefswwsize /var/cache/logs/data.log
```

```
total for cache
initial size:      4256k
end size:          511k
high water size:   511k
```

CacheFS File System Integrity

You use the `fsck(1M)` command to check and repair the integrity of file systems.

To check the integrity of the CacheFS file system, perform the following steps:

1. Unmount the CacheFS file system before invoking the `fsck` command.

```
# umount /data
```

2. Use the following command to check and repair the CacheFS file system:

```
# fsck -F cacheefs -o noclean /cache/cache0
```

The `-F` option informs the `fsck` command that the type of file system to check is the CacheFS file system type. The `-o noclean` option is used to force `fsck` to perform a check even if it determines that a check is not necessary.

3. Use the `mount` command to enable access to the repaired CacheFS file system.

```
# mount -F cacheefs -o backfstype=nfs,cachedir=/cache/cache0,\  
cacheid=data_cache host1:/export/data /data
```

Dismantling a CacheFS File System

Implementing a CacheFS file system can be an interim measure for enhancing performance. You can delete a CacheFS file system and recreate it at a later time.

Deleting a CacheFS file system (that is, deleting its cached copy) has no effect on the original back file system.

You can mount more than one back file system as a CacheFS file system and cache it in the same caching directory. You can dismantle one CacheFS file system, leaving others intact. (You can also dismantle all the CacheFS file systems in the caching directory.)

To dismantle a CacheFS file system, perform the following steps:

1. If necessary, warn users that their access to the CacheFS file system will be interrupted.
2. Determine the cache ID for the CacheFS file system you intend to delete. The ID string is located in the last line of the output of the following command. The remainder of the output is covered later in this module.

```
# cfsadmin -l /cache/cache0
cfsadmin: list cache FS information
maxblocks      90%
minblocks      0%
threshblocks   85%
maxfiles       90%
minfiles       0%
threshfiles    85%
maxfilesize    3MB
data_cache
```

3. Unmount all CacheFS file systems that share the same cache directory with the one you intend to delete. The examples in this module use only one cache directory, so the command is:

```
# umount /data
```

4. Delete the CacheFS file system.

```
# cfsadmin -d data_cache /cache/cache0
```

Note – To delete all CacheFS file systems in the cache directory, use the `cfsadmin -d all` command.

5. If some CacheFS file systems remain after others are deleted, use the `fsck` command to correct the resource counts in the cache directory.

```
# fsck -F cachefs -o noclean /cache/cache0
```

6. Remount the remaining CacheFS file systems.

Exercise: Configuring the CacheFS File System



Exercise objective – In this lab, you mount `/usr/share/man` to a CacheFS file system.

Preparation

Designate a machine in the lab that will store the man pages for clients to mount on the CacheFS file system.

Task Summary

In this exercise, you accomplish the following:

- Make a backup copy of `/usr/share/man` and then remove the original directory.
- Create a cache directory.
- Mount the `/usr/share/man` NFS file system to your CacheFS file system and list the files created in your cache directory.
- Check the status of the CacheFS file system and look at the size of the cache directory using the `df` command.
- Execute the `snoop` command from a terminal window to capture traffic between the server and the CacheFS system and then run the command `man ls` and note the network traffic.
- Perform a consistency check and note the hit rate and other information.
- Find the size of the cache directory using the `df` command and note the differences from the previous output.
- Remove the CacheFS file system and reinstall the original man pages from the backup copy or CD-ROM.

Tasks

Complete the following steps:

1. Move the man pages to another directory.
mv /usr/share/man /usr/share/man.save
2. Create a cache directory.
cfsadmin -c /export/cache_dir
3. Create a mount point for your CacheFS mount.
mkdir /usr/share/man
4. Mount the /usr/share/man NFS file system from the designated server to your CacheFS file system.

```
# mount -F cachefs -o \
backfstype=nfs,cachedir=/export/cache_dir,cacheid=007man,demandconst \
servername:/usr/share/man /usr/share/man
```

Note – You use the demandconst option to disable the consistency check features.

5. Use the mount command to identify the mount that succeeded.
mount | grep man
6. List the files created in the cache directory and note the cache ID number.
cd /export/cache_dir
ls -al
7. List the contents of the .cfs_mnt_points directory and note the mount points created.
ls -al .cfs_mnt_points
8. Check the status of the CacheFS file system.
cachefsstat /usr/share/man

9. Display file system size of the cache directory using the `df` command.

```
# df -k /export/cache_dir
```

10. Open another terminal window and run the following `snoop` command:

```
# snoop servername clientname
```

11. Access the `/usr/share/man` directory using the following command:

```
# man ls
```

Note – In the `snoop` window, take note of the network traffic that is generated.

12. Use the same `man` command again and note the network traffic generated. Why is there a difference?

13. Use the following commands to perform a manual consistency check. Note the hit rate and consistency checks.

```
# cfsadmin -s /usr/share/man
# cacheostat /usr/share/man
```

How have the numbers changed from the output of the previous (step 8) `cacheostat` command?

14. Display the file system size of the cache directory again using the `df` command.

```
# df -k /export/cache_dir
```

Note – Compare the size to that output in Step 9.

15. Run the `cfsadmin` command to determine the maximum amount of space in the `/export` directory that the cache can use.

```
# cfsadmin -l /export/cache_dir
```

16. Unmount the `/usr/share/man` directory and remove the CacheFS file system.

```
# umount /usr/share/man  
# cfsadmin -d 007man /export/cache_dir  
#
```

17. Remove the directory you created for the CacheFS file system.

```
# rmdir /usr/share/man
```

18. If the system is still running the `snoop` command from Step 10, press Control-C to terminate it.

19. Reinstall the original man pages

```
# mv /usr/share/man.save /usr/share/man
```

Exercise Summary



Discussion – Take a few minutes to discuss the experiences, issues, or discoveries that you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Task Solutions

1. Move the man pages to another directory.

```
# mv /usr/share/man /usr/share/man.save
```

2. Create a cache directory.

```
# cfsadmin -c /export/cache_dir
```

3. Create a mount point for your CacheFS mount.

```
# mkdir /usr/share/man
```

4. Mount the /usr/share/man NFS file system from the designated server to your CacheFS file system.

```
# mount -F cachefs -o \
backfstype=nfs,cachedir=/export/cache_dir,cacheid=007man,demandconst \
servername:/usr/share/man /usr/share/man
```

The demandconst option is used to disable the consistency check features.

5. Use the mount command to identify the mount that succeeded.

```
# mount | grep man
/usr/share/man on
/export/cache_dir/.cfs_mnt_points/servername:_usr_share_man
backfstype=nfs/cachedir=/export/cache_dir/cacheid=007man/demandconst on
Fri Aug 13 10:27:20 1999
```

6. List the files created in the cache directory and note the cache ID number.

```
# cd /export/cache_dir
# ls -al
total 33924
d----- 5 root      other      512 Aug 13 10:27 .
drwxr-xr-x 8 root      root       512 Aug 13 10:24 ..
-rw----- 1 root      other       48 Aug 13 10:24 .cfs_label
-rw----- 1 root      other       48 Aug 13 10:24 .cfs_label.dup
-rwx----- 1 root      other        0 Aug 13 10:24 .cfs_lock
drwx----- 3 root      other      512 Aug 13 10:27 .cfs_mnt_points
-rw----- 1 root      other    17334272 Aug 13 10:27 .cfs_resource
-rw-r--r-- 1 root      other        4 Aug 13 10:24 .cfs_unmnt
-rw-r--r-- 1 root      other       41 Aug 13 10:24 .nsr
drwxrwxrwx 10 root      root       512 Aug 13 10:27 0000000000004e000
lrwxrwxrwx 1 root      root       16 Aug 13 10:27 007man ->
0000000000004e000
drwx----- 2 root      other      512 Aug 13 10:24 lost+found
```

7. List the contents of the .cfs_mnt_points directory and note the mount points created.

```
# ls -al .cfs_mnt_points
drwx----- 3 root      other      512 Aug 13 10:27 .
d----- 5 root      other      512 Aug 13 10:27 ..
drwxr-xr-x 81 bin       bin      1536 Feb  8 1999
server:_usr_share_man
```

8. Check the status of the CacheFS file system.

```
# cachefsstat /usr/share/man

/usr/share/man
      cache hit rate:    100% (0 hits, 0 misses)
    consistency checks:    0 (0 pass, 0 fail)
              modifies:    0
    garbage collection:    0
```

9. Display file system size of the cache directory using the `df` command.

```
# df -k /export/cache_dir
```

Filesystem	kbytes	used	avail	capacity	Mounted on
/dev/dsk/c0t0d0s3	2828824	326557	2445691	12%	/usr/share/man

10. Open another terminal window and run the following `snoop` command, and leave it running.

```
# snoop servername clientname
Using device /dev/hme (promiscuous mode)
```

11. Access the `/usr/share/man` directory using the following command:

```
# man ls
```

Note – In the `snoop` window, take note of the network traffic that is generated.

```
clientname -> servername NFS C GETATTR3 FH=0082
servername -> clientname NFS R GETATTR3 OK
clientname -> servername NFS C ACCESS3 FH=0082lookup (lookup)
servername -> clientname NFS R ACCESS3 OK (lookup)
clientname -> servername NFS C GETATTR3 FH=40EE
servername -> clientname NFS R GETATTR3 OK
clientname -> servername NFS C ACCESS3 FH=40EEread (read)
servername -> clientname NFS R ACCESS3 OK (read)
clientname -> servername TCP D=2049 S=677 Ack=13938960
Seq=2251034193 Len=0 Win=8760
clientname -> servername NFS C GETATTR3 FH=5331
servername -> clientname NFS R GETATTR3 OK
clientname -> servername NFS C ACCESS3 FH=5331read (read)
servername -> clientname NFS R ACCESS3 OK (read)
clientname -> servername NFS C GETATTR3 FH=5331
< output deleted >
```

12. Use the same `man` command again and note the network traffic generated. Why is there a difference?

The output from the previous command is now accessed from the cache.

13. Use the following commands to perform a manual consistency check. Note the hit rate and consistency checks.

```
# cfsadmin -s /usr/share/man
# cachefsstat /usr/share/man
```

```
/usr/share/man
```

```
cache hit rate: 75% (12 hits, 3 misses)
consistency checks: 2 (2 pass, 0 fail)
modifies: 0
garbage collection: 0
```

How have the numbers changed from the output of the previous (step 8) `cachefsstat` command?

The cache hit rate should be less than 100 percent.

14. Display the file system size of the cache directory again using the `df` command.

```
# df -k /export/cache_dir
```

```
Filesystem      kbytes    used  avail capacity  Mounted on
/dev/dsk/c0t0d0s3 2828824 327456 2444792    12%    /export
```

Compare the size to that output in step 9.

The used column shows almost a 1 Mbyte increase (327456 - 326557=899 Kbytes).

15. Run the `cfsadmin` command to determine the maximum amount of space in the `/export` directory that the cache can use.

```
# cfsadmin -l /export/cache_dir
```

```
cfsadmin: list cache FS information
```

```
maxblocks      90%
minblocks      0%
threshblocks   85%
maxfiles       90%
minfiles       0%
threshfiles    85%
maxfilesize    3MB
```

```
007man
```


16. Unmount the `/usr/share/man` directory and remove the CacheFS file system.

```
# umount /usr/share/man  
# cfsadmin -d 007man /export/cache_dir
```

17. Remove the directory you created for the CacheFS file system.

```
# rmdir /usr/share/man
```

18. If the system is still running the `snoop` command from Step 10, press Control-C to terminate it.

19. Reinstall the original man pages.

```
# mv /usr/share/man.save /usr/share/man
```

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

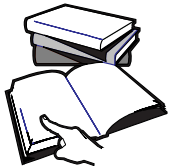
- ☐ Describe the CacheFS file system
- ☐ Use the appropriate commands to configure the CacheFS file system
- ☐ Use the appropriate commands to check the status and consistency of the CacheFS file system
- ☐ Set up CacheFS file system logging
- ☐ Describe the steps necessary to perform a check of the CacheFS file system
- ☐ List the steps to dismantle and delete a CacheFS file system

Objectives

Upon completion of this module, you should be able to:

- Build an association between users and roles with authorizations and execution profiles
- Define authorizations and their attributes
- List a profile's assigned authorizations
- Define the privileged operations that are assigned to a profile
- Identify help files that are associated with profiles and authorizations
- Configure a user's execution profile to allow access to a specified subset of system administrator privileges

Additional Resources



Additional resources – The following references provide additional details on the topics discussed in this module:

- *System Administration Guide, Volume I*, Sun Part Number 805-7228-10
- *System Administration Guide, Volume II*, Sun Part Number 805-7229-10

Role-Based Access Control

Role-based access control (RBAC) provides an alternative to the all-or-nothing security model of traditional superuser-based systems. The problem with the traditional model is not just that the superuser is so powerful, but that other users are not powerful enough to fix their own problems. RBAC enables you to assign subsets of superuser privileges to user accounts. This enables users to solve some of their own problems.

You can still control the capabilities of the superuser by dividing those capabilities into several packages and assigning them separately to individuals sharing administrative responsibilities.

RBAC includes the following features:

- Authorization – A right that is used to grant access to a restricted function
- Execution profile (or simply profile) – A bundling mechanism for grouping authorizations and commands with special attributes; for example, user and group IDs
- Role – A special type of user account intended for performing a specific set of administrative tasks



Caution – You can use RBAC to provide superuser access to administrative roles within the system. However, you must exercise caution to avoid creating security lapses when providing access to administrative accounts using these features.

Components

RBAC relies on four databases to provide users access to privileged operations:

- `/etc/user_attr` (extended user attributes database) – In addition to the `/etc/passwd`, `/etc/group`, and `/etc/shadow` files, this database associates users and roles with authorizations and execution profiles.
- `/etc/security/auth_attr` (authorization attributes database) – This database defines authorizations and their attributes and identifies the associated help file.
- `/etc/security/prof_attr` (execution profile attributes database) – This database defines profiles, lists the profile's assigned authorizations, and identifies the associated help file. Profiles can be logically named based on what the profile is designed to do.
- `/etc/security/exec_attr` (profile execution attributes database) – This database defines the privileged operations assigned to a profile.

Delimiters

Understanding the RBAC databases is simplified with the use of a common set of delimiters. These delimiters function as follows:

- Colon (:) – Used as a field separator within each database; for example,

`name:qualifier:res1:res2:attribute`

- Semicolon (;) – Used to separate key-value pairs within attribute fields; for example,

`...:attribute_type=value;attribute_profile=value;attribute_auth=value`

- Comma (,) – Used to separate an ordered list within a specific attribute key; for example,

`...;attribute_profile=profile_access1,profile_access2,profile_access3;...`

- period (.) – Used to separate the prefix from suffixes within authorization names to define execution profiles with finer granularity; for example,

`solaris.system.date::Set Date & Time::help=SysDate.html`

illustrates the interdependencies between the RBAC components. Databases are shown in boxes while the arrows indicate relationships between databases. The entities assigned in the relationships appear next to the arrows.

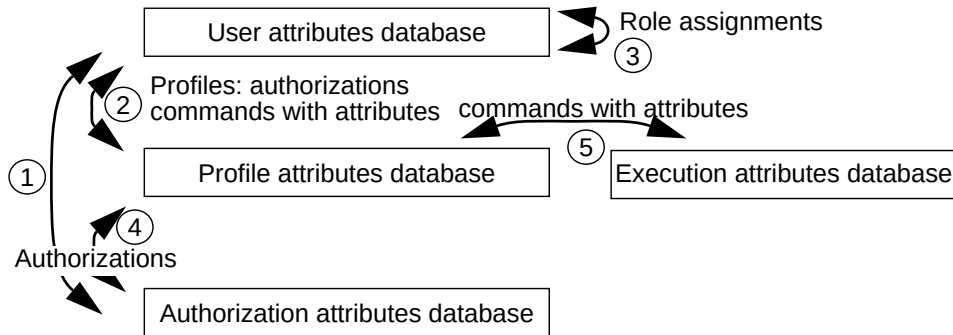


Figure 9-1 Component Interaction within RBAC

You can assign authorizations (1) and profiles (2) to users in the `user_attr` database; this is a direct assignment of privileged operations. You can also assign the user to a role (3), which gives the user access to any privileged operations associated with that role. Profiles are defined in the `prof_attr` database and can include authorizations (4) defined in `auth_attr` and commands with attributes (5) defined for that profile in `exec_attr`.

Note – illustrates that dependencies exist; the remainder of this module expands on these dependencies.

Commands that are assigned to profiles are run in special shells called *profile* shells. The profile shells are `pfsh`, `pfcsch`, and `pfksh`, and they correspond to the Bourne shell (`sh`), C shell (`csh`), and Korn shell (`ksh`) respectively.

Extended User Attributes Database (user_attr)

The `/etc/user_attr` database supplements the `passwd` and `shadow` databases. It contains extended user attributes, such as authorizations and execution profiles. It also allows you to assign roles to a user.

A *role* is a special type of user account that is intended for performing a set of administrative tasks. It is similar to a normal user account in most respects except that users gain access to it only through the `su` command. Users cannot access it using their normal procedures; for example, through the CDE login window. From a role account, a user can access commands with special attributes, typically `root` user ID, which are not available to users in normal accounts.

The fields in the `user_attr` database are separated by colons, as follows:

`user:qualifier:res1:res2:attr`

Table 9-1 describes the fields in the `user_attr` database.

Table 9-1 `user_attr` Database

Field Name	Description
<code>user</code>	The name of the user, as specified in the <code>passwd(4)</code> database.
<code>qualifier</code>	Reserved for future use.
<code>res1</code>	Reserved for future use.
<code>res2</code>	Reserved for future use.
<code>attr</code>	An optional list of semicolon-separated (;) key-value pairs that describe the security attributes to be applied when the user runs commands. There are four valid keys: <code>auths</code>, <code>profiles</code>, <code>roles</code>, and <code>type</code>. • <code>auths</code> specifies a comma-separated list of authorization names chosen from names defined in the <code>auth_attr(4)</code> database. Authorization names might include the asterisk (*) character as a wildcard. For example, <code>solaris.device.*</code> means all of the Solaris Operating Environment device authorizations. • <code>profiles</code> contains an ordered, comma-separated list of profile names chosen from <code>prof_attr(4)</code>. A profile determines which commands a user can execute and with which command attributes. At minimum, each user in <code>user_attr</code> should have the <code>All</code> profile, which makes all commands available but without any attributes. The order of profiles is important; it works similarly to UNIX search paths. The first profile in the list that contains the command to be executed defines which (if any) attributes are to be applied to the command. • <code>roles</code> can be assigned to the user using a comma-separated list of role names. Roles are defined in the same <code>user_attr</code> database. They are indicated by setting the <code>type</code> value to <code>role</code>. You cannot assign roles to other roles. • <code>type</code> can be set to <code>normal</code>, if this account is for a normal user, or to <code>role</code>, if this account is for a role. A role is assumed by a normal user after the user has logged in.

The following is an example of a `user_attr` database, with typical values:

```
root::::type=normal;auth=solaris.*,solaris.grant;profiles=All
sysadmin::::type=role;profiles=Device Management,Filesystem
Management,All
johndoe::::type=normal;auths=solaris.system.date;roles=sysadmin;profiles=
All
```

Repeating this typical role assignment (and emphasizing the details) you can see that the `sysadmin` role has been assigned to the user `johndoe`. When assuming the `sysadmin` role, `johndoe` has access to specific profiles, such as `Device Management`, `Filesystem Management`, and the `All` profile.

```
root::::type=normal;auth=solaris.*,solaris.grant;profiles=All
sysadmin::::type=role;profiles=Device Management,Filesystem
Management,All
johndoe::::type=normal;auths=solaris.system.date;roles=sysadmin;profiles=
All
```

Authorizations

An authorization is a user right that grants access to a restricted function. It is a unique string that identifies what is being authorized as well as who created the authorization.

Authorizations are checked by certain privileged programs to determine whether users can execute restricted functionality. For example, the `solaris.jobs.admin` authorization is required for one user to edit another user's crontab file.

All authorizations are stored in the `auth_attr` database. You can assign authorizations directly to users (or roles) in which case they are entered in the `user_attr` database. You can also assign authorizations to execution profiles, which in turn are assigned to users.

The fields in the `auth_attr` database are separated by colons:

`authname:res1:res2:short_desc:long_desc:attr`

Table 9-2 describes the fields in the `auth_attr` database.

Table 9-2 `auth_attr` Database

Field Name	Description
<code>authname</code>	A unique character string used to identify the authorization in the format <code>prefix.[suffix]</code>. Authorizations for the Solaris Operating Environment use <code>solaris</code> as a prefix. All other authorizations should use a prefix that begins with the reverse-order Internet domain name of the organization that creates the authorization (for example, <code>com.xyzcompany</code>). The suffix indicates what is being authorized, typically the functional area and operation. When there is no suffix (that is, the <code>authname</code> consists of a prefix and functional area and ends with a period), the <code>authname</code> serves as a heading for use by applications in their GUIs rather than as an authorization. The <code>authname solaris.printmgr</code> is an example of a heading. When <code>authname</code> ends with the word <code>grant</code>, the <code>authname</code> serves as a grant authorization and lets the user delegate related authorizations (that is, authorizations with the same prefix and functional area) to other users. The <code>authname solaris.printmgr.grant</code> is an example of a grant authorization; it gives the user the right to delegate authorizations, such as <code>solaris.printmgr.admin</code> and <code>solaris.printmgr.nobanner</code> to other users.
<code>res1</code>	Reserved for future use.
<code>res2</code>	Reserved for future use.
<code>short_desc</code>	A terse name for the authorization suitable for displaying in user interfaces, such as a GUI.
<code>long_desc</code>	A long description. This field identifies the purpose of the authorization, the applications in which it is used, and the type of user interested in using it. You can display the long description in the help text of an application.
<code>attr</code>	An optional list of semicolon-separated (;) key-value pairs that describe the attributes of an authorization. You can specify zero or more keys. The help keyword identifies a help file in HTML. You can access help files from the <code>index.html</code> file in the <code>/usr/lib/help/auths/locale/C</code> directory.

The following is an example of an `auth_attr` database, with some typical values:

```
solaris.*:::Primary Administrator::help=PriAdmin.html
solaris..grant:::Grant All Rights::help=PriAdmin.html
...
solaris.device:::Device Allocation::help=DevAllocHeader.html
solaris.device.allocate:::Allocate Device::help=DevAllocate.html
solaris.device.config:::Configure Device Attributes::help=DevConfig.html
solaris.device.grant:::Delegate Device Administration::help=DevGrant.html
solaris.device.revoke:::Revoke or Reclaim Device::help=DevRevoke.html
```

The relationship between the `auth_attr` and the `user_attr` databases is illustrated in the following example. The `solaris.system.date` authorization, which is defined in the `auth_attr` database, is assigned to the user `johndoe` in the `user_attr` database.

```
root:::type=normal;auths=solaris.*,solaris.grant;profiles=All
...
johndoe:::type=normal;auths=solaris.system.date;roles=sysadmin;profiles=
All
...

solaris.*:::Primary Administrator::help=PriAdmin.html
...
solaris.system.date:::Set Date & Time::help=SysDate.html
...
```

Execution Profiles

An execution profile is a bundling mechanism for grouping authorizations and commands with special attributes, and assigning them to users or roles. The special attributes include real and effective UIDs and GIDs. The most common attribute is setting the real or effective UID to root. The definitions of execution profiles are stored in the `prof_attr` database.

The fields in the `prof_attr` database are separated by colons:

`profname:res1:res2:desc:attr`

Table 9-3 describes the fields in the `prof_attr` database.

Table 9-3 `prof_attr` Database

Field Name	Description
<code>profname</code>	The name of the profile. Profile names are case-sensitive.
<code>res1</code>	Reserved for future use.
<code>res2</code>	Reserved for future use.
<code>desc</code>	A long description. This field should explain the purpose of the profile, including what type of user would be interested in using it. The long description should be suitable for displaying in the help text of an application.
<code>attr</code>	An optional list of key-value pairs separated by semicolons (;) that describe the security attributes to apply to the object upon execution. You can specify zero or more keys. There are two valid keys, <code>help</code> and <code>auths</code>. • The keyword <code>help</code> identifies a help file in HTML. You can access help files from the <code>index.html</code> file in the <code>/usr/lib/help/auths/locale/C</code> directory. • The <code>auths</code> keyword specifies a comma-separated list of authorization names chosen from those names defined in the <code>auth_attr(4)</code> database. You can specify authorization names using the asterisk (*) character as a wildcard.

The following is an example of a `prof_attr` database, with some typical values:

```
All:::Standard Solaris user:help=All.html
...
Printer Management:::Manage print jobs:help=Printer.html
Device Management:::Control Access to Removable Media:
auths=solaris.device.*;help=DevMgmt.html
...
```

The following example illustrates the relationship between the `prof_attr` and the `user_attr` databases. The Device Management profile, which is defined in the `prof_attr` database, is assigned to the `sysadmin` role in the `user_attr` database.

```
root:::type=normal;auth=solaris.*,solaris.grant;profiles=All
sysadmin:::type=role;profile=Device Management,Printer Management
```

```
All:::Standard Solaris user:help=All.html
...
Printer Management:::Manage print jobs:help=Printer.html
Device Management:::Control Access to Removable Media:
auths=solaris.device.*;help=DevMgmt.html
```

The following example illustrates the relationship between the `prof_attr` and the `auth_attr` databases. The Device Management profile is defined in the `prof_attr` database as having all authorizations beginning with the `solaris.device.string` assigned to it. These authorizations are defined in the `auth_attr` database.

```
All:::Standard Solaris user;help=All.html
```

```
...
```

```
Device Management:::Control access to Removable Media:  
auths=solaris.device.*;help=DevMgmt.html
```

```
solaris.*::Primary Administrator::help=PriAdmin.html
```

```
solaris.grant::Grant All Rights::help=PriAdmin.html
```

```
...
```

```
solaris.device::Device Allocation::help=DevAllocHeader.html
```

```
solaris.device.allocate::Allocate Device::help=DevAllocate.html
```

```
solaris.device.config::Configure Device Attributes::help=Dev Config.html
```

```
solaris.device.grant::Delegate Device Administration::help=DevGrant.html
```

```
solaris.device.revoke::Revoke or Reclaim Device::help=DevRovoke.html
```

```
...
```


Execution Attributes

An execution attribute associated with a profile is a command (with special security attributes) that can be run by those users or roles to whom the profile is assigned. Special security attributes refer to attributes, such as UID, EUID, GID, EGID that can be added to a process when the command is run.

The definitions of the execution attributes are stored in the `exec_attr` database.

The fields in the `exec_attr` database are separated by colons:

`name:policy:type:res1:res2:id:attr`

Table 9-4 describes the fields in the `exec_attr` database.

Table 9-4 `exec_attr` Database

Field Name	Description
<code>name</code>	The name of the profile. Profile names are case-sensitive.
<code>policy</code>	The security policy associated with this entry. Currently, <code>suser</code> (the superuser policy model) is the only valid policy entry.
<code>type</code>	The type of entity whose attributes are specified. Currently, the only valid type is <code>cmd</code> (command).
<code>res1</code>	Reserved for future use.
<code>res2</code>	Reserved for future use.
<code>id</code>	A string identifying the entity. You can use the asterisk wild card. Commands should have the full path or a path with a wild card. To specify arguments, write a script with the arguments and point the <code>id</code> to the script.
<code>attr</code>	An optional list of semicolon (;) separated key-value pairs that describe the security attributes to apply to the entity upon execution. You can specify zero or more keys. The list of valid key words depends on the policy being enforced. There are four valid keys: <code>euid</code>, <code>uid</code>, <code>egid</code>, and <code>gid</code>. • <code>euid</code> and <code>uid</code> contain a single user name or a numeric user ID. Commands designated with <code>euid</code> run with the effective UID indicated, which is similar to setting the <code>setuid</code> bit on an executable file. Commands designated with <code>uid</code> run with both the real and effective UIDs. • <code>egid</code> and <code>gid</code> contain a single group name or numeric group ID. Commands designated with <code>egid</code> run with the effective GID indicated, which is similar to setting the <code>setgid</code> bit on an executable file. Commands designated with <code>gid</code> run with both the real and effective GIDs.

The following is an example of an `exec_attr` database, with some typical values:

```
All:suser:cmd::*:
...
Printer Management:suser:cmd:::/usr/lib/lp/lpsched:euid=0
Printer Management:suser:cmd:::/usr/lib/lp/lpshut:euid=0
Printer Management:suser:cmd:::/usr/lib/lp/lpmove:euid=0
Printer Management:suser:cmd:::/bin/lp:euid=0
Printer Management:suser:cmd:::/bin/lpadmin:euid=0
Printer Management:suser:cmd:::/usr/sbin/lpadmin:euid=0
Printer Management:suser:cmd:::/usr/bin/enable:euid=0
Printer Management:suser:cmd:::/usr/bin/disable:euid=0
Printer Management:suser:cmd:::/usr/sbin/accept:euid=0
Printer Management:suser:cmd:::/usr/sbin/reject:euid=0
Printer Management:suser:cmd:::/usr/sbin/lpsystem:euid=0
...
```

The following example illustrates the relationship between the `exec_attr` and the `prof_attr` databases. The Printer Management profile is defined in the `prof_attr` database. It has 13 execution attributes with the appropriate security attributes assigned in the `exec_attr` database.

```
All:::Standard Solaris user:help=Al.html
...
Printer Management::Manage print jobs:help=Printmgt.html
...
```

```
All:suser:cmd::*:
...
Printer Management:suser:cmd:::/etc/init.d/lp:euid=0
Printer Management:suser:cmd:::/usr/bin/cancel:euid=0
Printer Management:suser:cmd:::/usr/bin/lpset:egid=14
Printer Management:suser:cmd:::/usr/bin/enable:euid=lp
Printer Management:suser:cmd:::/usr/bin/disable:euid=lp
Printer Management:suser:cmd:::/usr/sbin/accept:euid=lp
Printer Management:suser:cmd:::/usr/sbin/reject:euid=lp
Printer Management:suser:cmd:::/usr/sbin/lpadmin:egid=14
Printer Management:suser:cmd:::/usr/sbin/lpfilter:euid=lp
Printer Management:suser:cmd:::/usr/sbin/lpforms:euid=lp
Printer Management:suser:cmd:::/usr/sbin/lpmove:euid=lp
Printer Management:suser:cmd:::/usr/sbin/lpshut:euid=lp
Printer Management:suser:cmd:::/usr/sbin/lpusers:euid=lp
...
```

Role-Based Access Control Overview

Defines users and roles with authorizations (auth_attr) and profiles (prof_attr).
Supplements shadow and passwd databases.

```
/etc/user_attr
USER:RES:RES:RES:ATTR=VAL, VAL;ATTR=VAL, VAL
root:::type=normal;auths=solaris.*,solaris.grant;profiles=All
```

USER = The user name – /etc/passwd
ATTR = auths, profile, roles, type
(role = special user account – /etc/user_attr)
(type = normal or role)
(normal = user account – /etc/passwd)
RES = Reserved

Defines profile name, authorizations (auth_attr) and specifies the help file.
Each profile is defined in exec_attr.

```
/etc/security/prof_attr
PROFNAME:RES:RES:DESC:ATTR=VAL, VAL;ATTR=VAL
All:::Standard Solaris user:help=All.html
Audit Review:::View the audit trail:auths=solaris.audit.read;help=AuditReview.html
Printer Management:::Control Access to Printer:help=PrinterMgmt.html
```

PROFNAME = The name of profile defined in exec_attr
ATTR = The security attributes (help or auths)
RES = Reserved

```
/etc/security/auth_attr
AUTHNAME:RES:RES:SHORT_DESC:LONG_DESC:ATTR=VAL
solaris.*:::Primary Administrator::help=PriAdmin.html
solaris.grant:::Grant All Rights::help=PriAdmin.html
solaris.audit.read:::Read Audit Trail::help=AuditRead.html
...
```

Defines what is being authorized and help files. Can be assigned directly to user_attr as a role. Can also be used in prof_attr.

AUTHNAME = *prefix.suffix*
ATTR = help (html file)
solaris = Authority for Solaris
grant = Allows a user to delegate authorization
* = All *prefix.xxxx*
RES = Reserved

Defines the privileged operations of the Profile Name (prof_attr).
UID, EUID, GID, and EGID for commands

```
/etc/security/exec_attr
NAME:POLICY:TYPE:RES:RES:ID:ATTR=VAL
All:suser:cmd:::
Audit Review:suser:cmd:::/usr/sbin/auditreduce:euid=0
Printer Manager:suser:cmd:::/etc/init.d/lp:euid=0
```

NAME = The name of profile
POLICY = *suser* (security policy)
TYPE = *cmd* (command)
ID = The full path of command or a script (* = all commands)
ATTR = UID, EUID, GID, EGID
RES = Reserved

Figure 9-2 Role-based Access Control

Assuming Role-Based Access Control

To assume a **role**, you must use the **su** command. You cannot log in to a role. For example:

```
% su my-role  
Password: my-role-password  
#
```

To use commands in the profile, type into a shell. For example:

```
# lpadmin -p myprinter options
```

The **lpadmin** command is executed with any process attributes, such as special UIDs or GIDs. You must have assigned these attributes to the **lpadmin** command in the profiles.

Tools for Managing Role-Based Access Control

In addition to editing the databases directly, a number of tools are available for managing role-based access control. The tools include:

- `roleadd` – Adds a role account on the system.
- `rolemod` – Modifies a role's login information.
- `useradd` – Adds a user account on the system.

The roleadd Command

The `roleadd` command adds a role entry to the `/etc/passwd`, `/etc/shadow`, and `/etc/user_attr` files. Some common options include:

- `-c comment`

Any text string. It is generally a short description of the role.

- `-d dir`

The home directory of the new role.

- `-m`

Create the new role's home directory if it does not already exist.

- The `-A` authorization and `-P` profile

These options respectively assign authorizations and profiles to the role.

```
roleadd -m -d /export/home/tarback -c "Privileged tar Backup Role" \  
-P "Backup and Restore" tarback
```

The `roleadd` command creates a new role called `tarback`, builds the required directory structures (including the home directory), and assigns the role with a profile of `Backup and Restore`.

Note – `Backup and Restore` should have been previously edited into the `exec_attr` file and the `prof_attr` file.

The rolemod Command

The `rolemod` command modifies a role's login information on the system. It changes the definition of the specified login and makes the appropriate login-related system file and file system changes.

- `-A authorization`

One or more comma separated authorizations, as defined in `/etc/security/auth_attr`.

- `-e expire`

Specifies the expiration date for a role.

- `-l new_logname`

Specifies the new login name for the role.

- `-P profile`

One or more comma separated authorizations, as defined in `/etc/security/auth_attr`.

- `-s shell`

Specifies the full pathname of the program that is used as the role's shell when logging in.

These options respectively modify authorizations and profiles to the role.

```
# rolemod -P auth1,auth2 role1
```

This is a generic example of the `rolemod` command that assigns authorizations `auth1` and `auth2` to the role named `role1`. These named authorizations must be previously defined in the `/etc/security/auth_attr` database.

The useradd Command

The `useradd` command adds a new user to the `/etc/passwd`, `/etc/shadow`, and `/etc/user_attr` files. Some common options include:

- `-c comment`

Any text string. It is generally a short description of the `login` command and is currently used as the field for the user's full name.

- `-d dir`

The home directory of the new user.

- `-m`

Creates the new user's home directory if it does not already exist.

- `-s shell`

Full pathname of the program used as the user's shell on `login`

- `-R role`

One or more comma-separated execution profiles defined in `user_attr(4)`.

```
# useradd -m -d /export/home/usera -c "User Account usera" -s  
/usr/bin/ksh -R datuser usera
```

The `useradd` command creates a new user account for a user named `usera`. The user has access to the role `datuser`, and the user's shell has been changed from the default Bourne shell to the Korn shell.

Additional Commands

Table 9-5 describes some additional commands that you can use with RBAC operations.

Table 9-5 RBAC Commands

Command	Description
auths(1)	Displays authorizations for a user.
makedbm(1M)	Makes a dbm file.
nscd(1M)	Identifies the name service cache daemon, which is useful for caching the user_attr, prof_attr, and exec_attr databases.
pam_roles(5)	Identifies the role account management module for the Password Authentication Module (PAM). Checks for authorization to assume role.
pfexec(1)	Identifies profile shells, used by profile shells to execute commands with attributes specified in the exec_attr database
policy.conf(4)	Identifies the configuration file for security policy. Lists granted authorizations.
profiles(1)	Displays profiles for a specified user.
roles(1)	Displays roles granted to a user.
roleadd(1M)	Adds a role account on the system.
roledel(1M)	Deletes a role's account from the system.
rolemod(1M)	Modifies a role's account information on the system.
useradd(1M)	Adds a user account on the system. The -R option assigns a role to a user's account.
userdel(1M)	Deletes a user's login from the system.
usermod(1M)	Modifies a user's account information on the system.

Note – Reference <http://docs.sun.com> for further information on these commands.

Creating a User and a Role

To create a user and a role, perform the following steps:

1. Create the role.

```
# roleadd -u 1000 -g 10 -d /export/home/minime -m username
# passwd minime
```

2. Create the profile.

```
# vi /etc/security/prof_attr
Shut:::Able to shutdown the system:
```

3. Add the profile to the role.

```
# rolemod -P Shut,All username
```

4. Verify that the changes have been made in the /etc/user_attr file.

```
# more /etc/user_attr
```

5. Create the user.

```
# useradd -u 1001 -g 10 -d /export/home/user1 -m -s /bin/ksh -R username
user1
# passwd user1
# more /etc/passwd /etc/user_attr
```

6. Assign commands to the profile:

```
# vi /etc/security/exec_attr
Shut:suser:cmd:::/usr/sbin/shutdown:uid=0
```

Testing the Configuration

To test the configuration, complete the following steps.

1. Log in as user1.
2. Use the su command to assume the role *username*.
3. Issue the following command:

```
# /usr/sbin/shutdown -i 6 -g 0
```

Exercise: Implementing System Security



Exercise objective – In this lab, you:

- Create an execute attribute
- Create a role-based profile
- Create a role identity
- Create a login identity that can make use of the role

Preparation

During the lab, you are directed to carry out commands that do not work in order to demonstrate how the RBAC facility must be used by login users.

Task Summary

In this exercise, you configure and test role-based access control.

Tasks

Creating a Role

You need to create an entry in the `/etc/security/exec_attr` file. This entry allows a user to execute the `date` command with an effective ID of 0 (the `root` user). This allows the user to set the system date and time even though that user did not log in as `root`.

1. Add the following line to the end of the `/etc/security/exec_attr` file:

Date Management:suser:cmd:::/usr/bin/date:euid=0

- ▼ The first field of data is a descriptive field name. This field's contents become the official name by which this attribute is known. Because this example includes a Space character, the field name must always be enclosed in quotes when used with either the `useradd` or `roleadd` (or user/role associated) commands.
- ▼ The second field value, `suser`, is required. This value is the only value currently supported by the Solaris 8 Operating Environment, although other values might be added in the future.
- ▼ The third field contains the word `cmd`. This is a required value that denotes that this attribute relates to a command.
- ▼ Two empty fields follow. These are currently not in use and should be left empty.
- ▼ The sixth field contains the absolute pathname of the command that is to be executed. In this instance, the command to be executed is the `/usr/bin/date` command.
- ▼ The final field states which effective user ID value will be assigned to the user when the command (`/usr/bin/date`) is executed. Entering a value of 0 gives the user the effective identity of the `root` user.

2. Save and exit from that file.

3. You must enter a profile attribute entry in the `/etc/security/prof_attr` file, as follows:

Date Management:::Date Setting:

- ▼ As with the `exec_attr` file, the first data field contains the official name for the attribute. This must exactly match the first field as used in the `/etc/security/exec_attr` file.
- ▼ The second, third and fourth fields are currently not used.
- ▼ The fifth field is a comment field and can contain any descriptive text that might be required.

4. Using the `roleadd` command, create a role entry:

```
# roleadd -m -d /export/home/datuser -c "RBAC Lab example" -s  
/usr/bin/pfksh -P "Date Management",All datuser  
# passwd datuser
```

- ▼ The words `Date` and `Management` must be enclosed in quotes to be treated as a one-name entry.
- ▼ The word `All` does not require the quotes because it consists of just one word. The word *All* relates to a predefined profile that should exist in the files at the time of installation of the Solaris 8 Operating Environment. This profile allows a user to execute any valid UNIX command while functioning in a role-based capacity.

Note – One of the lab exercises that follows asks you to remove this from the appropriate file and then test whether the role-based user can execute commands, such as the `ls` command.

- ▼ The shell that is being used by the user is `/usr/bin/pfksh`. This is a special version of the Korn shell that allows you to use the RBAC profiles. If the user had been created to use a standard shell (such as `/usr/bin/ksh`), then that user would not be able to be assigned a role.

5. Two profiles have been used for the role called `datuser`. You can view these profiles by viewing the contents of the `/etc/user_attr` file, as shown in the following example:

```
# cat /etc/user_attr
# Copyright (c) 1999 by Sun Microsystems, Inc. All rights reserved.:
#:::
# /etc/user_attr:::
#:::
# user attributes. see user_attr(4):::
#:::
#pragma ident    "@(#)user_attr  1.2      99/07/14 SMI":
#:::
root:::type=normal;auths=solaris.*,solaris.grant;profiles=All
datuser:::type=role;profiles=Date Management,All
```

The `type` field contains the value `role`. This designates that the name `datuser` can be used only for role-assignment and cannot be used as a valid login name.

You must create a user that can make use of the `datuser` attribute.

You create this user by issuing the following command:

```
# useradd -m -d /export/home/userb -c "Role user (userb)" -s /usr/bin/ksh
-R datuser userb
# passwd userb
```

Note – Provide an appropriate password for the role user (`userb`).

After you have added the user, the following line should appear in the `/etc/user_attr` file:

```
userb:::type=normal;roles=datuser
```

Note – The `type` field contains the value `normal` instead of `role`. This indicates that the name `userb` is a valid login name.

You have created the profile and execute attributes, and you have created a user who can make use of these attributes. You should test that the user called `userb` can now set the system's date and time.

Test Role

Complete the following steps:

1. Log in to the system as `userb`. Attempt to execute the following commands.

The output of the commands, listed below, is shown for example only. Your output will be different for some of the commands issued.

```
$ who
userb console      May  4 15:23
$ ls
local.cshrc  local.login  local.profile
$ id
uid=102(userb) gid=1(other)
$ pwd
/export/home/userb
$
```

2. Check the current date and time using the `date` command.

```
$ date
Thu May  4 15:23:39 BST 2000
```

3. Add two minutes to the current time, and attempt to update the system time with that new value.

In this example, as the date format is `MMDDhhmm` and the current values are `05041523`, the new date and time value that will be used is `05041525`.

```
$ date -u 05041525
```

You should receive the following error message:

```
date: Not owner
usage:  date [-u] mmddHHMM[[cc]yy][.SS]
        date [-u] [+format]
        date -a [-]sss[.fff]
```

This is because you do not have the appropriate authority to change the system date and time.

If you assume the role of `datuser`, you are granted that authority.

4. Use the `su` command to assume the `datuser` role, by issuing the following command:

```
$ su datuser
```

```
Password: <enter the password for datuser here>
```

5. Validate the current login and session identities, using the following commands:

```
$ who
```

```
userb console      May  4 15:23
```

```
$ id
```

```
uid=103(datuser) gid=1(other)
```

6. Attempt to execute some of the standard UNIX commands:

```
$ ls
```

```
local.cshrc  local.login  local.profile
```

```
$ pwd
```

```
/export/home/userb
```

7. Run the `date` command to assess the current date and time details; for example:

```
$ date
```

```
Thu May  4 15:24:43 BST 2000
```

8. As before, try to add two minutes to the current time and see if you can reset the system's date and time details.

```
$ date -u 05041527
```

```
Thu May  4 15:27:00 GMT 2000
```

You should be successful because you have assumed the role of `datuser`.

9. Exit from the `su` session, using the `exit` command, and then log off as the user (`userb`).
10. Log in to the system as the `root` user.

11. Edit the `/etc/user_attr` file and modify the `datuser` line, as follows:

The line currently reads:

```
datuser::::type=role;profiles=Date Management,All
```


Remove the comma and the word **All** from the end of the line so that the line now reads:

datuser::::type=role;profiles=Date Management

12. Save and exit from the file and then log out as the **root** user.

13. Log in as the user called **userb**.

14. Issue the following commands (output might differ on your system):

```
$ id
uid=102(userb) gid=1(other)
$ who
userb console      May  4 16:43
$ pwd
/export/home/userb
$ ls
local.cshrc  local.login  local.profile
$ date
Thu May  4 16:43:27 BST 2000
```

15. Switch to the **datuser** role, using the **su** command.

```
$ su datuser
Password: <enter the appropriate password>
```

16. Execute a range of standard UNIX commands:

```
$ date
Thu May  4 16:43:38 BST 2000
$ who
pfksh: who:  not found
$ ls
pfksh: ls:   not found
$ id
pfksh: id:   not found
```

Only the **date** command is currently valid, because it is the only **exec** authority that applies to the **datuser** role

17. Exit from the **datuser** session and then log off as the user (**userb**).

Modify Roles

Complete the following steps.

1. Log in to the system as the root user.
2. Make the following amendments to the files:
 - a. Add the following line to the end of the /etc/security/auth_attr file:

```
solaris.backup.:::Backup and Restore::help=index.html
```

- b. Add the following line to the end of the /etc/security/exec_attr file:

```
Backup and Restore:user:cmd::/usr/sbin/tar:uid=0
```

- c. Add the following line to the end of the /etc/security/prof_attr file:

```
Backup and Restore:::Control Backup and restore using tar:
```

3. Create the role, using the roleadd command.

```
# roleadd -m -d /export/home/tarback -c "Privileged tar Backup Role" \  
-P "Backup and Restore,All" tarback
```

4. Assign a password to the tarback role.

```
# passwd tarback  
New password: tarback  
Re-enter new password: tarback  
passwd (SYSTEM): passwd successfully changed for tarback
```

5. Edit the /etc/user_attr file and modify the entry for userb as follows. The line should currently read:

```
userb.:::type=normal;roles=datuser
```

Edit the entry so that it reads:

```
userb.:::type=normal;roles=datuser, tarback
```

This adds the role of a privileged tar user to the user called userb.

6. Save and exit from the file.
7. Log out as the root user.

8. Log in as `userb`, and issue the following commands.

Note – The output of these commands will differ depending on your system.

```
$ id
uid=102(userb) gid=1(other)
$ cd /etc
$ cat /etc/shadow
cat: cannot open /etc/shadow
$
$ tar cvf /tmp/shadowfile.backup ./shadow
tar: ./shadow: Permission denied
$
```

9. Switch to the `tarback` role, using the `su` command, as shown below:

```
$ su tarback
Password: <...enter the appropriate password>
```

10. You should still be situated in the `/etc` directory but will not be able to use most UNIX commands because of the earlier stage of the exercise where the `All` profile was removed. Test to see that you are in the `/etc` directory using the following command:

```
$ pwd
/etc
```

11. Test to see if the role, `tarback`, has access to the contents of the `shadow` file, using the following commands:

```
$ tar cvf /tmp/shadowfile.backup ./shadow
a ./shadow 1K
$
$ tar tvf /tmp/shadowfile.backup
tar: blocksize = 4
-r-----  0/3      375 May  4 17:11 2000 ./shadow
$
```

12. Exit from the `tarback` role by issuing the `exit` command.

```
$ exit
```

13. Because the backup file was effectively created by the `userb`, working in the `tarback` role, the resultant backup file is owned by the `userb`. To validate this, issue the following command:

```
$ ls -l /tmp/shadowfile.backup
-rw-r--r--  1 userb other          2048 May  4 17:16
/tmp/shadowfile.backup
```

14. Change the directory back to the `userb` `$HOME` directory.

```
$ cd
$ pwd
/export/home/userb
```

15. Because `userb` owns the backup file, `userb` should be able to extract from that file into a file in the `userb` `$HOME` directory.

```
$ tar xvf /tmp/shadowfile.backup
tar: blocksize = 4
x ./shadow, 375 bytes, 1 tape blocks
```

Note – This was achieved only because the original backup was made using a relative pathname for the `/etc/shadow` file. If an absolute pathname had been used, the only way the file could be restored from a backup would be to restore it to its absolute-pathname position.

16. Ensure the file that has been restored.

```
$ cat shadow
root:X2ApPcp5SERlg:6445::::::
daemon:NP:6445::::::
bin:NP:6445::::::
sys:NP:6445::::::
adm:NP:6445::::::
lp:NP:6445::::::
uucp:NP:6445::::::
nuucp:NP:6445::::::
listen:*LK*::::::
nobody:NP:6445::::::
noaccess:NP:6445::::::
nobody4:NP:6445::::::
liz:a0PJDaxoxpGbM:11081::::::
userb:KTW978x91tLnc:11081::::::
datuser:1zSiCkDZmkejw:11081::::::
tarback:.dRSpVHAqXNnU:11081::::::
```

17. Validate that your personal copy of the `shadow` file is identical in size to the original version of the file, as stored in the `/etc` directory, using the following commands:

```
$ ls -l shadow
-r----- 1 userb other      375 May  4 17:11
shadow
$ ls -l /etc/shadow
-r----- 1 root      sys      375 May  4 17:11
/etc/shadow
$ pwd
/export/home/userb
```

18. Log out as the user (`userb`).

Exercise Summary



Discussion – Take a few minutes to discuss the experiences, issues, or discoveries that you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ Build an association between users and roles with authorizations and execution profiles
- ☐ Define authorizations and their attributes
- ☐ List a profile's assigned authorizations
- ☐ Define the privileged operations that are assigned to a profile
- ☐ Identify help files that are associated with profiles and authorizations
- ☐ Configure a user's execution profile to allow access to a specified subset of system administrator privileges

Solaris Management Console™ and Solaris AdminSuite

10 

Objectives

Upon completion of this module, you should be able to:

- List the benefits of the Solaris Management Console™
- Install the Solaris Management Console software
- Add an application to the Solaris Management Console
- List the features of Solaris AdminSuite
- Install the Solaris AdminSuite software
- Create and modify user accounts using the Users feature of Solaris AdminSuite
- Add hosts to the server using the Computers/Networks feature of Solaris AdminSuite
- Manipulate mount states on existing file systems using the Mounts/Shares feature of Solaris AdminSuite
- Configure serial ports using the Serial Ports feature of Solaris AdminSuite

Additional Resources



Additional resources – The following references provide additional details on the topics discussed in this module:

- Solaris Management Console Help
- Solaris AdminSuite 3.0.1 Help

The Solaris Management Console

The Solaris Management Console (SMC), also called the Console, is a Java™ technology-based tool for administration of the servers. It provides a central integration point for important applications and services.

SMC software simplifies the job of configuring and administering servers. With point-and-click GUI tools, SMC makes the Solaris Operating Environment easy to administer, especially for administrators not familiar with the UNIX environment.

The Console enables users and administrators to register other SMC servers and applications on the network they wish to administer. When you access the Console, it dynamically configures tree views of those registered hosts and services. By pointing and clicking, you can remotely invoke an application on a selected SMC server and view the application's GUI on the local display.

The Benefits of Using the Console

The benefits of using the Console include:

- The network features of the SMC reduce the number of remote logins required to do administration, including rebooting systems, performing backups, and so on.
- The SMC simplifies user administration by bringing all the tools together in one location.
- You can manage all SMC servers from one location.
- Once a new application is added to an SMC server, anyone can run that application from that server.
- The SMC gives the user a graphical representation of the administration components available and the level of user privilege required to run them.

Installation Requirements

The SMC client is an application written entirely in Java programming language. The client package does not include a Java runtime environment, like the Solaris 2.6 and Solaris 7 Operating Environments. The supported Java environments are JDK™ 1.1.5 and JDK 1.1.6. The distribution CD-ROM (SEAS 2.0) has JDK1.1.6 on it if you need to upgrade your system. You can also download the JDK for the Solaris Operating Environment from SUN's web site, <http://www.sun.com/solaris/java>.

Download Procedure

Installing the Solaris Operating Environment requires root privilege on the destination machine. It involves adding the SMC client software package and providing system-wide usage of the client for all users.

Use the `ftp` command to download the `Solaris_Management_Console.shar` and `AdminSuite.shar` files from the classroom server. The following is an example of the `ftp` exchange that takes place:

```
# ftp classroom_server
Connected to classroom_server.
220 classroom_server FTP server (SunOS 5.8) ready.
Name (classroom_server:root):
331 Password required for root.
Password:
230 User root logged in.
ftp> cd pub
250 CWD command successful.
ftp> bin
200 Type set to I.
ftp> get Solaris_Management_Console.shar
200 PORT command successful.
150 Binary data connection for Solaris_Management_Console.shar
(129.147.11.62,34935) (20065485 bytes).
226 Binary Transfer complete.
ftp> get AdminSuite.shar
150 Binary data connection for AdminSuite.shar (192.9.200.9,33727)
(10254380 bytes).
226 Binary Transfer complete.
ftp> bye
```

Installing SMC

To install the SMC on your system (installation steps for Solaris AdminSuite start on page 10-15), perform the following steps:

1. Log in to the CDE desktop environment as `root`, and, with your shell located in the download directory, begin the installation the SMC client with this command:

```
# sh Solaris_management_Console.shar
```

This starts an installation wizard as shown in Figure 10-1.

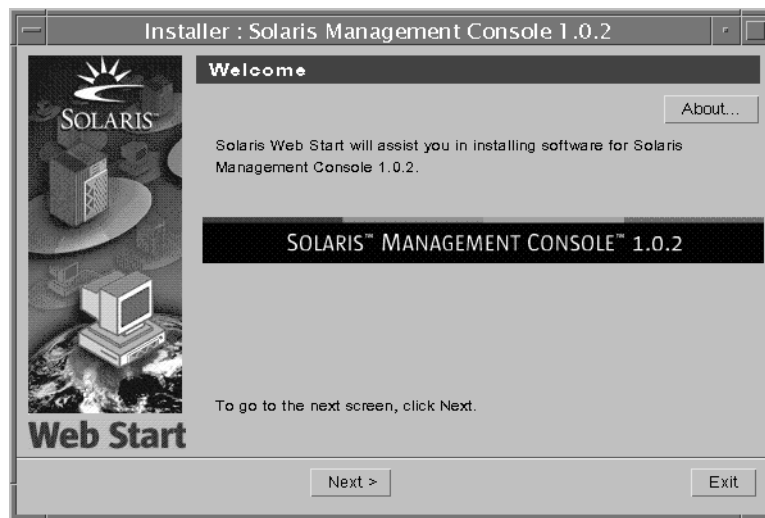


Figure 10-1 Solaris Management Console Install Wizard Window

2. Click Next to continue the installation.

A screen enabling the selection of either the Default or Custom Install is displayed.

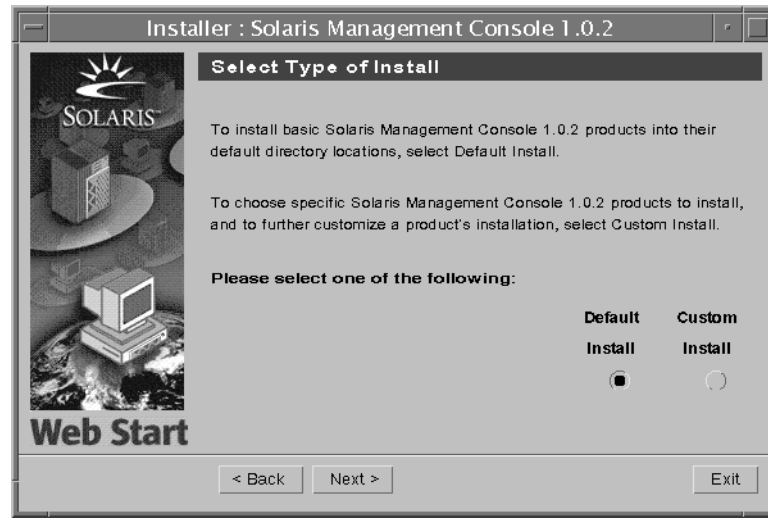


Figure 10-2 SMC Install Wizard - Select Type of Install Window

3. For the purpose of this module, select Custom Install to see all installation choices possible.

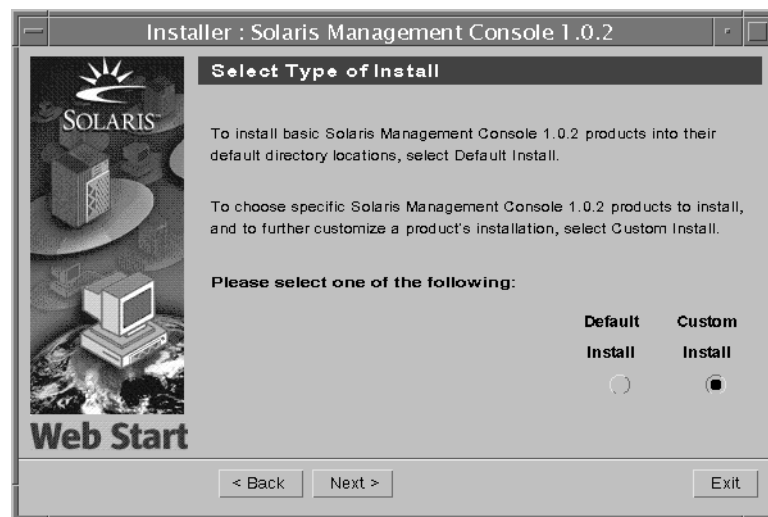


Figure 10-3 SMC Install Wizard - Select Type of Install Window

The Locale Selection window is displayed to enable the selection of other languages, in addition to the default English:

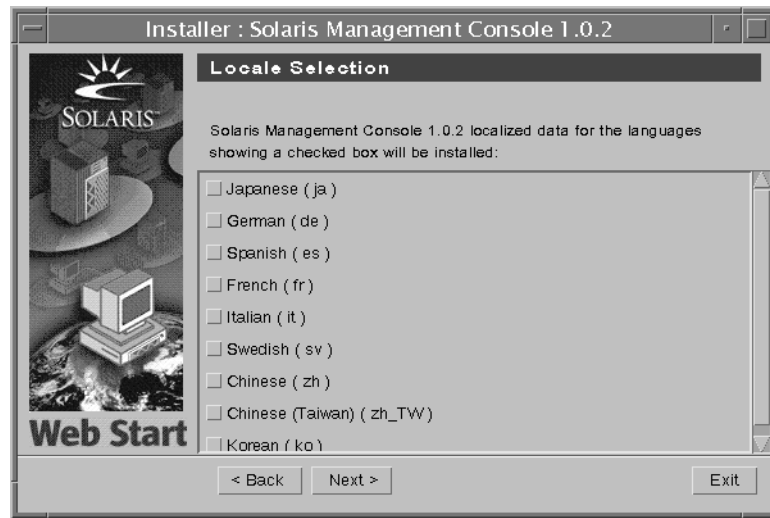


Figure 10-4 SMC Install Wizard – Locale Selection Window

4. Select the appropriate language(s) for your locale.
5. Click Next.

The Select Install Directory window is displayed.

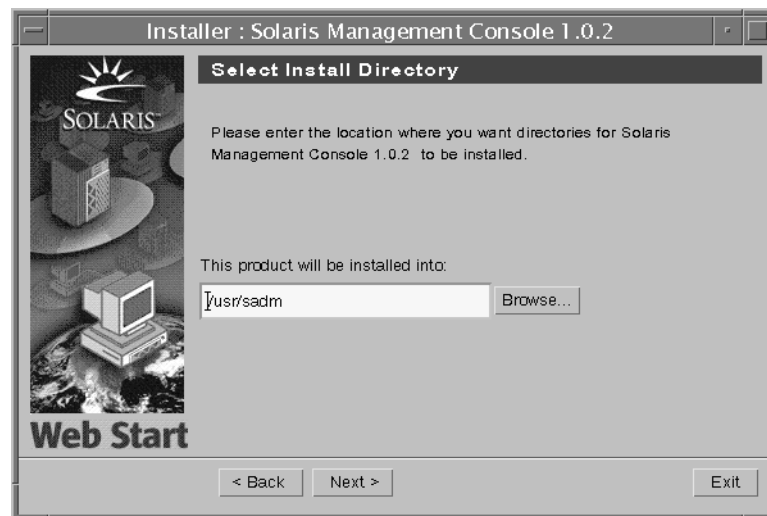


Figure 10-5 SMC Install Wizard – Select Install Directory Window

6. Accept the default install directory location of /usr/sadm.
7. Click Next.

This displays the Component Selection screen.

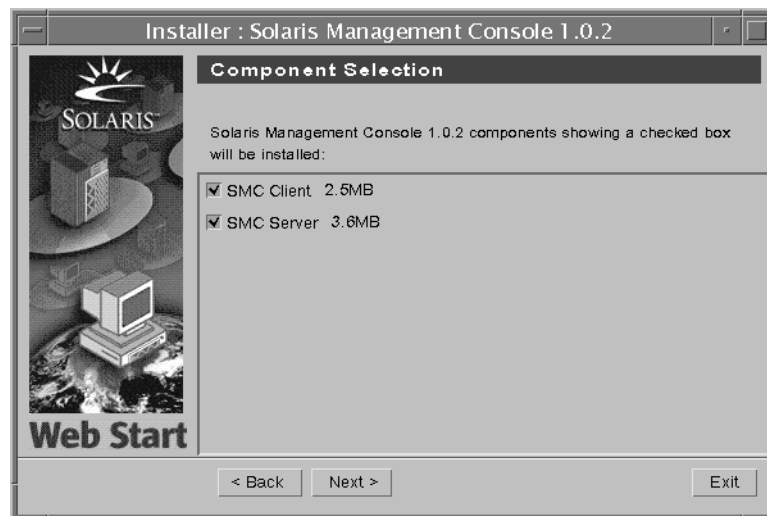


Figure 10-6 SMC Install Wizard – Component Selection Window

8. Accept the default selections of both the SMC client and server.
9. Click Next.

This displays the Ready to Install screen.

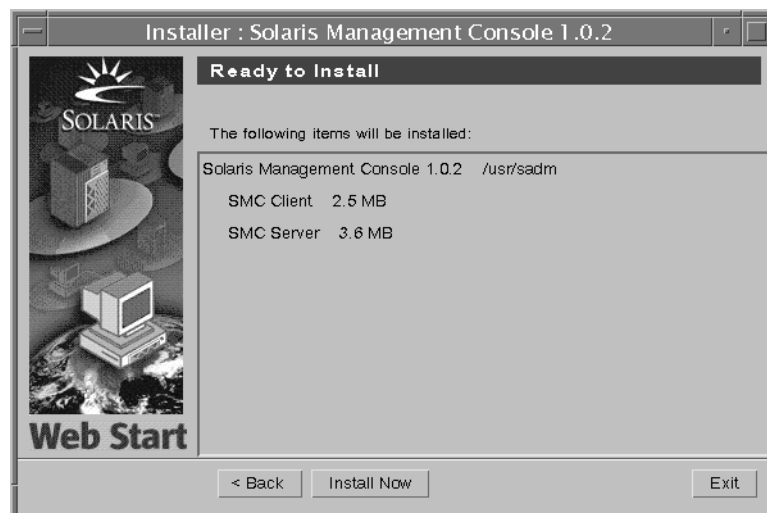


Figure 10-7 SMC Install Wizard – Ready to Install Window

10. Ensure that both the SMC Client and Server items are selected for installation.
11. Click Install Now.

The Installing window is displayed, and it shows the install progress as it takes place.

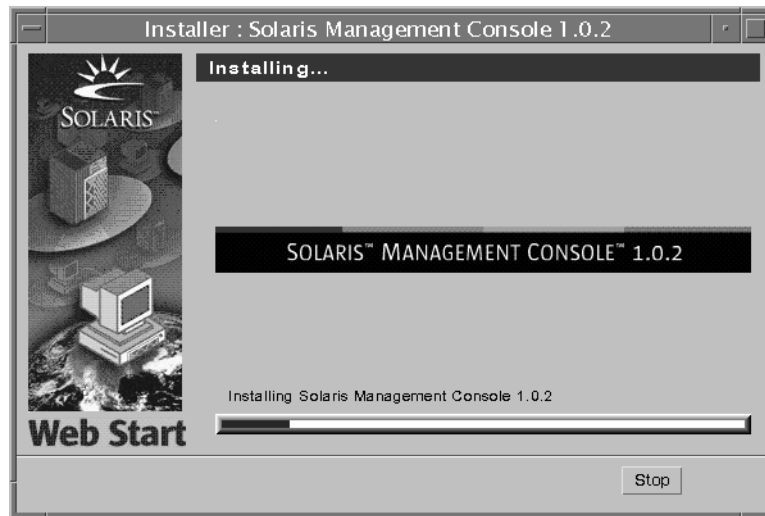


Figure 10-8 SMC Install Wizard – Installing Window

Once installation is complete, the install wizard displays the Installation Summary window, which confirms the success of the installation.

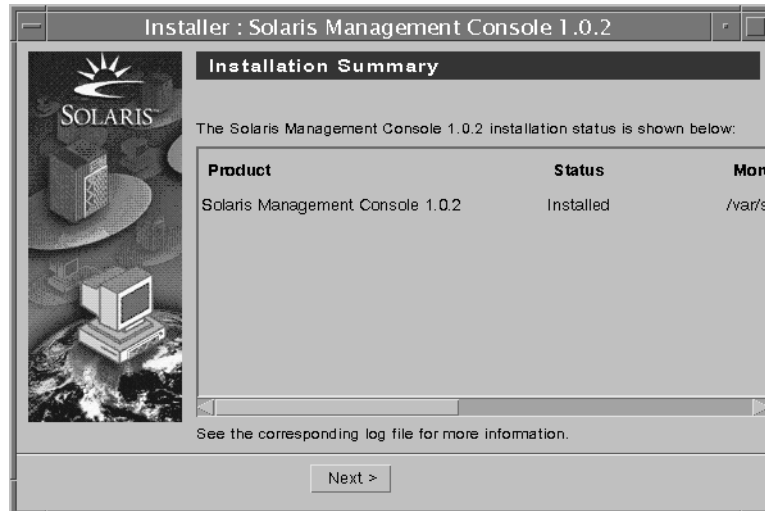


Figure 10-9 SMC Install Wizard – Installation Summary Window

12. Click Next to continue.

The Additional Information screen is displayed and provides a brief explanation of the Solaris Product Registry and the `/usr/bin/prodreg` command, which enables installation and removal of installed products.



Warning – Use the `/usr/bin/prodreg` command to remove any products (such as the SMC) that are added to the Solaris Operating Environment using installation wizards. Using this command ensures a complete removal. You can use this same command to install additional software.

Note – This window also references the URL location for AnswerBook2™ documentation, <http://docs.sun.com>.

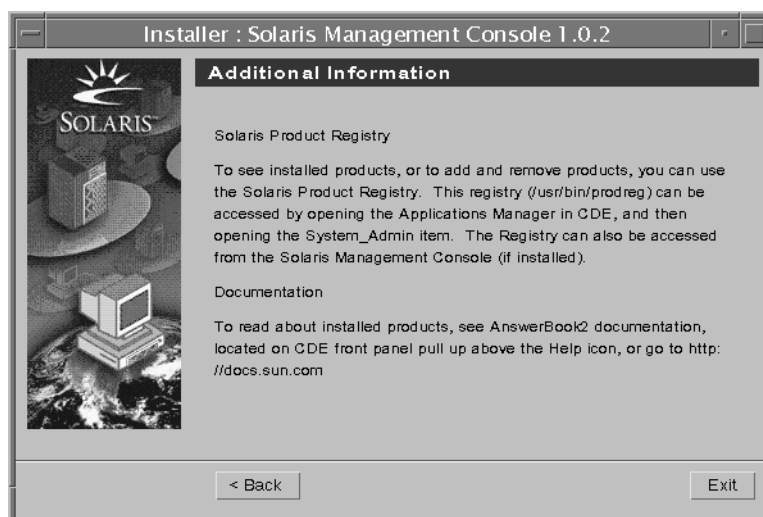


Figure 10-10 SMC Install Wizard – Additional Information Window

13. Click Exit to complete the install process.

14. Create a new user account.

This account is used for access to the SMC application and in the Solaris AdminSuite exercise.

Running the SMC Application

After you have installed SMC client software on a machine, perform the following steps to start the Console:

1. Type `smc` from the command line.

```
$ smc &
```

After a few moments, the initial SMC window is displayed with the name of the current SMC server in the Server field.

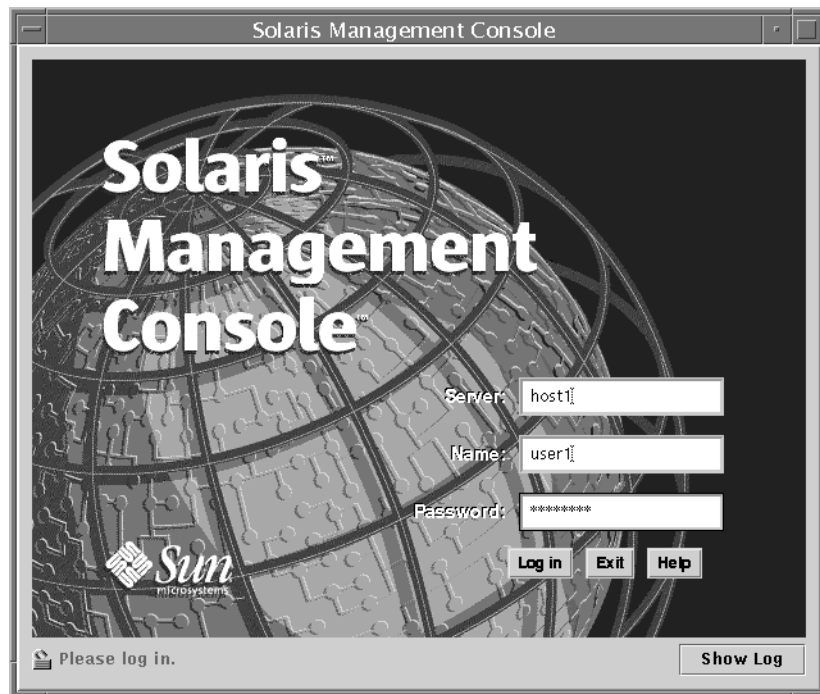


Figure 10-11 SMC Initial Console Window

2. Enter the server name if it is not already displayed.

To log in to a machine other than the current SMC server, replace the name in the Server field. The server name is limited to 40 characters.

3. Log in as a user (*not* root).

Once you are running the Console, if root or another user ID is necessary for the application you want to run, the Authorization Required dialog box is displayed.

4. Enter the password for the logged in user.

5. Click the Log in button.
6. After a short delay, the Console is displayed, in Applications View, listing a number of categories within which software applications are available on the current server.

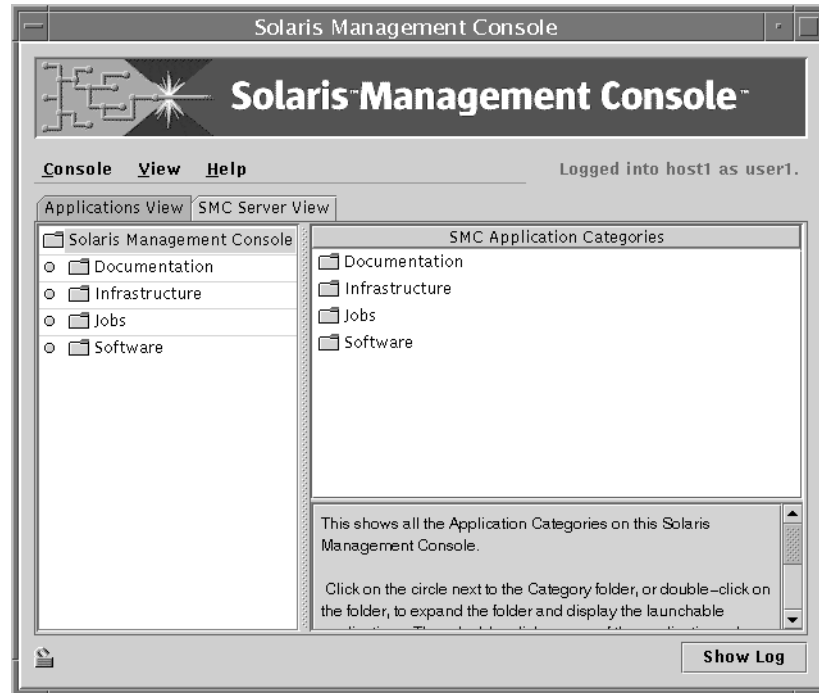


Figure 10-12 SMC Applications View Console Window

When you start the Console for the first time on a particular host, it is displayed in Applications View, with a number of service categories listed. The applications available to be managed by the Console are those applications that have been registered as SMC applications for the current host. In Applications View, a number of SMC application categories (4 in the example in Figure 10-12 are listed: Documentation, Infrastructure, Jobs, and Software).

To expand the application folders shown on the left-most frame, click the small circle just to the left of the category. In the following example, with the Documentation folder circle selected, the display opens to show the AnswerBook application:

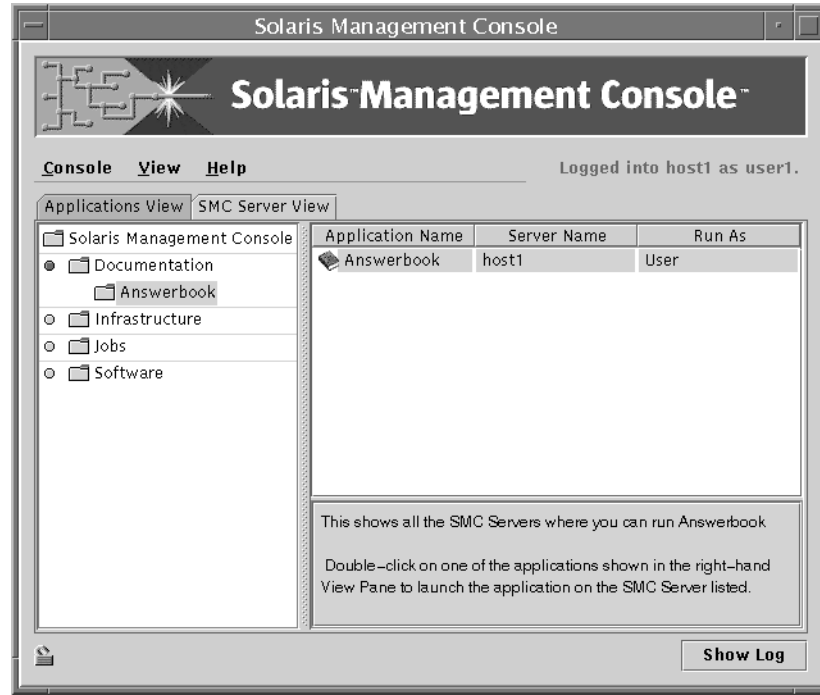


Figure 10-13 SMC View of the AnswerBook Application

You can start the AnswerBook application by double-clicking on the icon by that name in the right-most frame. Any of the other folders can be expanded and their respective applications started in the same way.

Solaris AdminSuite

Solaris AdminSuite contains GUI features that automate:

- User administration
- Group management
- Host administration
- File system manipulation
- Serial ports configuration

Each of the previously mentioned features is launched from the Solaris Operating Environment AdminSuite graphical user interface (GUI).

User Manager

The User Manager enables easy setup and maintenance of user accounts. This includes adding new users, removing users, and updating user information. Some of the defaults that you can set up include:

- Login shell
- Password policy
- The user's mail server

Group Manager

You can use the Group Manager to add, display, modify, and delete groups in the NIS or NIS+ name service environment or on a local or remote system (/etc files).

Host Manager

The Host Manager is used for connecting client systems to the network as well as modifying and deleting them. Supported client types include standalone, diskless, AutoClient, and JavaStation. The Host Manager also enables you to add operating environment services and set up remote installation services.

Mount/Share Manager

There are three components of the Mount/Share Manager: a File System Manager, Mounted File System Monitor, and Shared File System Monitor. The File System Manager manages mounted file systems on a server.

Serial Port Manager

The Serial Port Manager is used for adding and maintaining port services for terminals and modems. It can display serial port information and facilitate port set up, modification, and deletion. It also provides templates for common terminal and modem configurations.

Installation Procedure

To install the AdminSuite application on your system, perform the following steps:

1. Log in to the CDE desktop environment as root, (with your shell located in the download directory).
2. Run the following commands to stop NIS daemons on the NIS master only, and then install the SMC client.

```
# /usr/lib/netsvc/yp/ypstop  
# sh AdminSuite.shar
```

This starts an install wizard as shown in Figure 10-14.

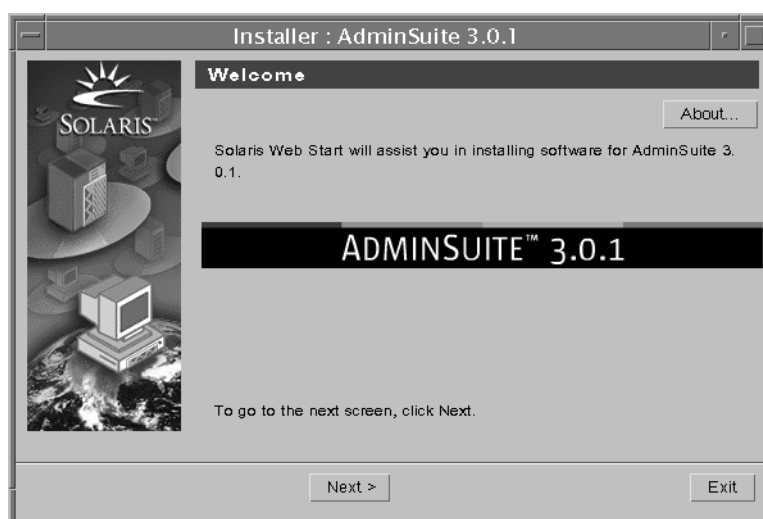


Figure 10-14 AdminSuite Application Install Wizard

3. Click Next to continue the installation.

This displays a screen enabling the selection of either the Default or Custom Install.



Figure 10-15 AdminSuite Install Wizard – Select Type of Install Window

4. For the purpose of this module, select Custom Install to see all installation choices possible.
5. Click Next

The Locale Selection window is displayed to enable the selection of other languages, in addition to the default English:



Figure 10-16 AdminSuite Install Wizard – Locale Selection Window

6. Select the appropriate languages for your locale.
7. Click Next.

The Select Install Directory window is displayed.



Figure 10-17 AdminSuite Install Wizard – Install Directory Selection Window

8. Accept the default install directory location of /opt.
9. Click Next.

The Component Selection window is displayed.

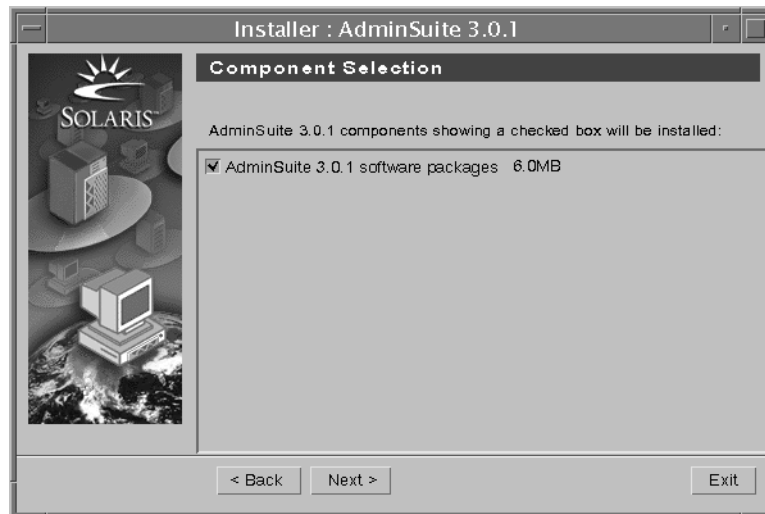


Figure 10-18 AdminSuite Install Wizard – Component Selection Window

10. Accept the default selections of AdminSuite 3.0.1 software packages.
11. Click Next.

The Primary Administrator window is displayed:.

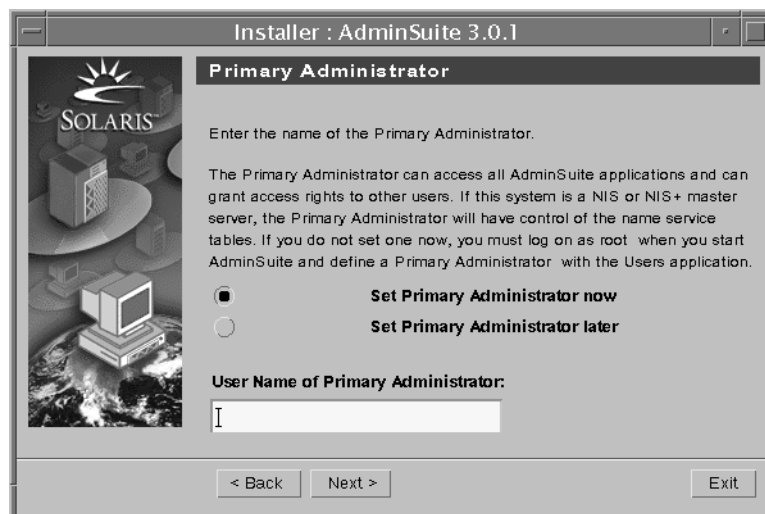


Figure 10-19 AdminSuite Install Wizard – Primary Administrator Window

12. Select Set Primary Administrator now.

13. Type your user name in the empty field; the example that follows indicates user1 set to be the primary administrator:

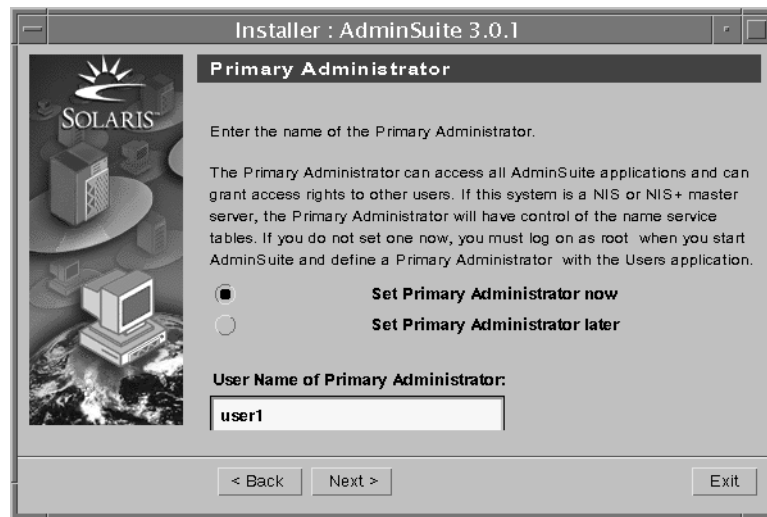


Figure 10-20 AdminSuite Install Wizard – Primary Administrator Window

14. Click Next.

The Ready to Install window is displayed.

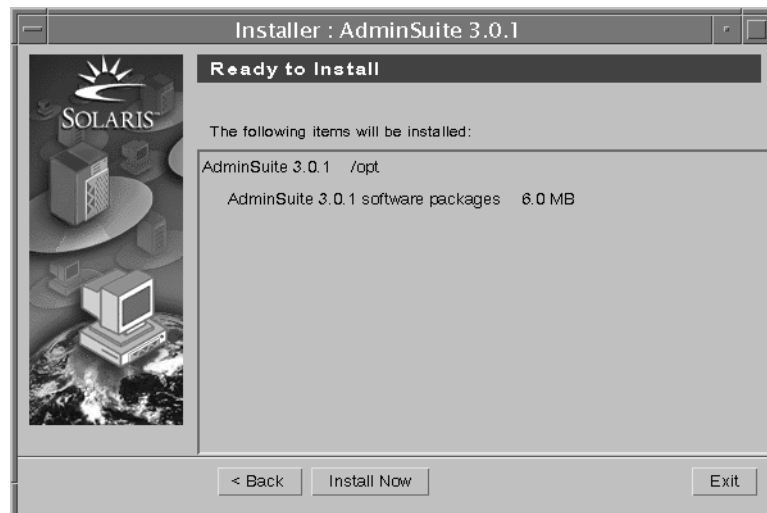


Figure 10-21 AdminSuite Install Wizard – Ready to Install Window

15. Ensure that the AdminSuite 3.0.1 item is shown.
16. Click Install Now.

The Installing window is displayed, and it shows the install progress as it takes place.



Figure 10-22 AdminSuite Install Wizard – Installing Window

After the installation is complete, the install wizard displays the Installation Summary window confirming installation success.:



Figure 10-23 AdminSuite Install Wizard – Installation Summary Window

17. Click Exit to complete the install process.

The Solaris AdminSuite GUI can be used by either the superuser or a user that belongs to Group 14, the administrative group.

Note – Unless privileges have been modified using RBAC, the `root` identity has privileges (to access and update data) only on the local system. Solaris AdminSuite permissions are granted to users who are members of the `sysadmin` group (group 14). This means that a user modifying administration data must be a member of the `sysadmin` group on the system where the task is being executed.

Before running the Solaris AdminSuite program, execute the following commands:

```
# /opt/SUNWseam/3_0/sbin/dirtbl_setup initial
# init 6
```

To run the Solaris AdminSuite tools, log in as `root` and type the command `/opt/SUNWseam/3_0/bin/admapp` at the command-line prompt. The Solaris AdminSuite window is displayed.

```
# /opt/SUNWseam/3_0/bin/admapp &
```

Starting Solaris AdminSuite

When you launch the Solaris AdminSuite application, the AdminSuite banner is displayed (see Figure 10-24).

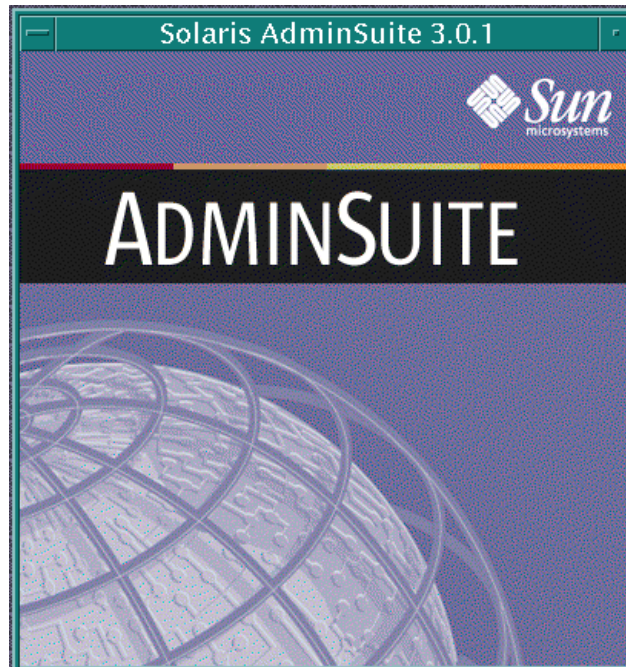


Figure 10-24 Solaris AdminSuite Banner Window

You must enter a login and password before using the Solaris AdminSuite application (see Figure 10-25).

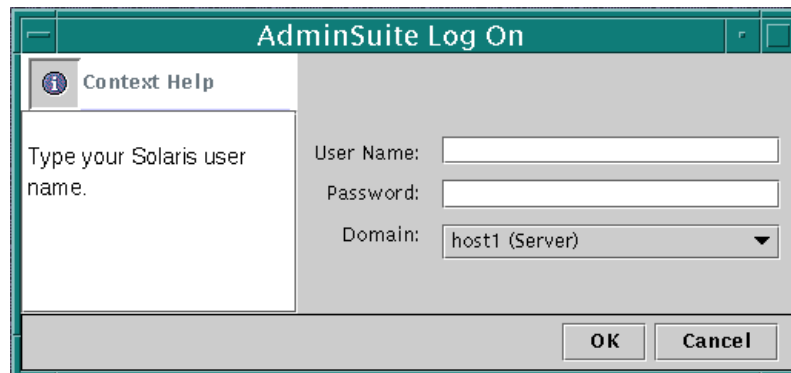


Figure 10-25 AdminSuite Log On Window

Selecting a Name Service

In earlier releases of the Solaris AdminSuite software, you had to select a name service when you launched the individual tools that comprised the Solaris AdminSuite package. In this release of the Solaris AdminSuite application, you select name service configuration using the Domain menu, as shown in Figure 10-26 on page 10-26:

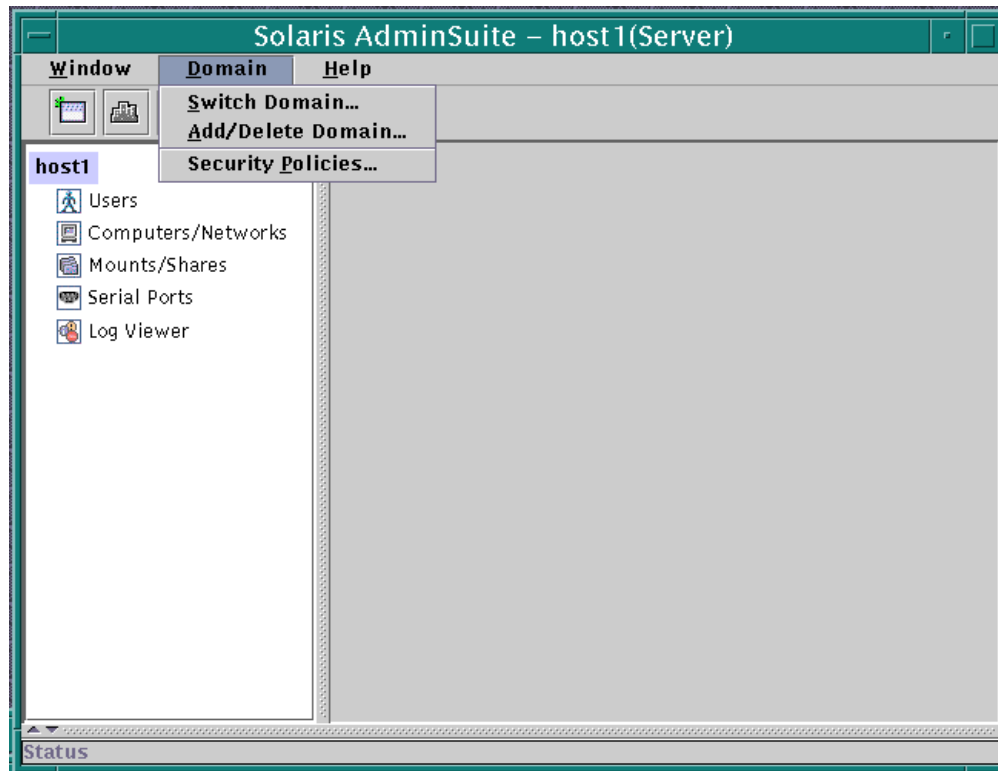


Figure 10-26 Solaris AdminSuite – Domain Menu

1. Click Domain►Security Policies (see Figure 10-26).

You are presented with the Name Service Selection display (see Figure 10-27 on page 10-27).

You must be a primary administrator for the server shown in the title bar, before you can set security policies for the server.

2. Select your desired name service configuration by clicking on the appropriate selections.

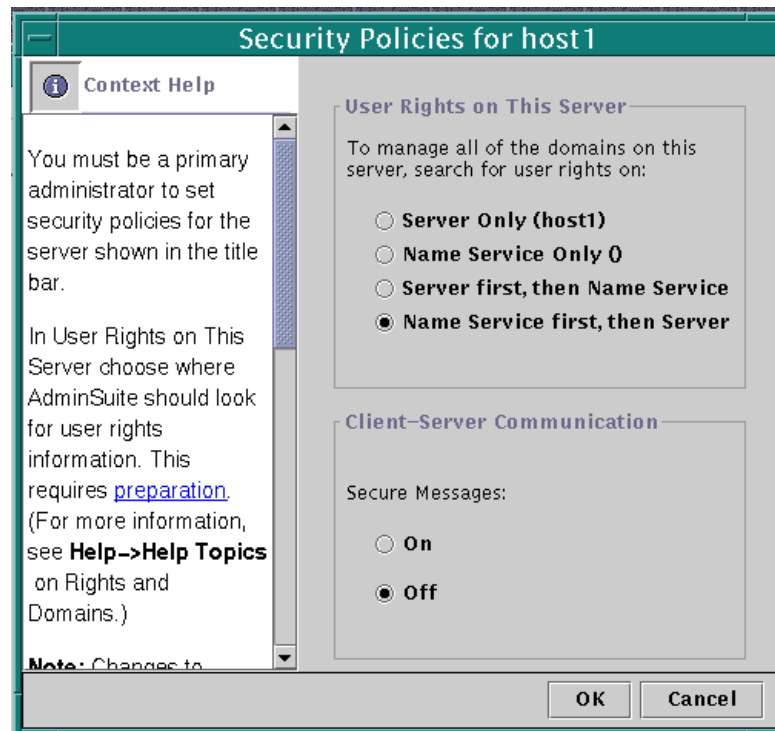


Figure 10-27 Security Policies for Host Window

3. Click OK.

Note – You must configure the selected name services before selecting them, or you will receive an error message alerting you to potential problems.

Solaris AdminSuite Components

The initial Solaris AdminSuite window is shown in Figure 10-28. The display can be divided into three segments, as follows:

- The menu segment in the upper left corner
- The Context Help segment across the bottom
- The current context workspace occupies the upper right segment

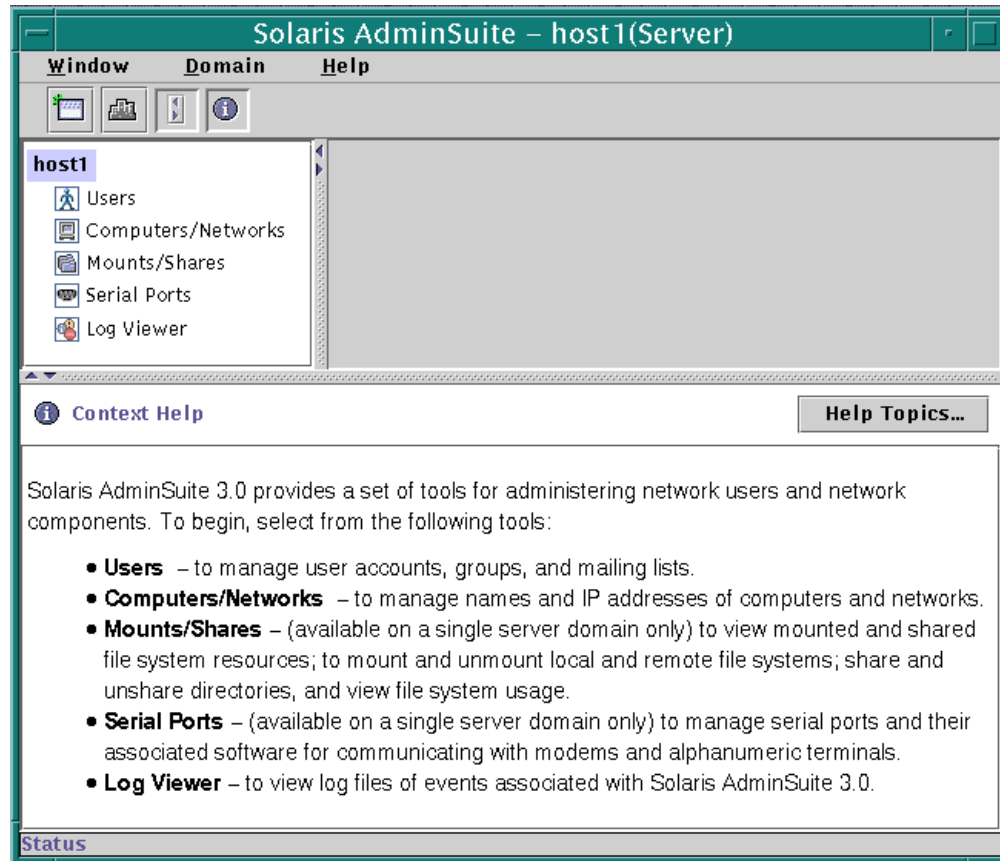


Figure 10-28 Solaris AdminSuite – Host Window

The current context workspace contains displays for the feature of Solaris AdminSuite that you are currently using. You can expand this workspace to absorb the menu segment space by clicking on the left arrow or right arrow on the slider bar or by sliding the slider bar itself. Similarly, the workspace can absorb the Context Help segment with its arrows or slider bar.

Users Feature

You can access the Users feature through the Users Menu, which expands this feature to configure users, configure groups, or configure mailing lists. Click the Users icon in the upper left segment (on Figure 10-29 on page 10-30) to display the desktop segment (or upper right segment).

Note – In Figure 10-29 on page 10-30, the (bottom) Context Help segment has been absorbed to provide a larger desktop (workspace) segment in the upper right.

Viewing Users

Double-clicking Users in the menu expands the menu selection to include its sub-menu selections, then double-click the User Accounts icon.

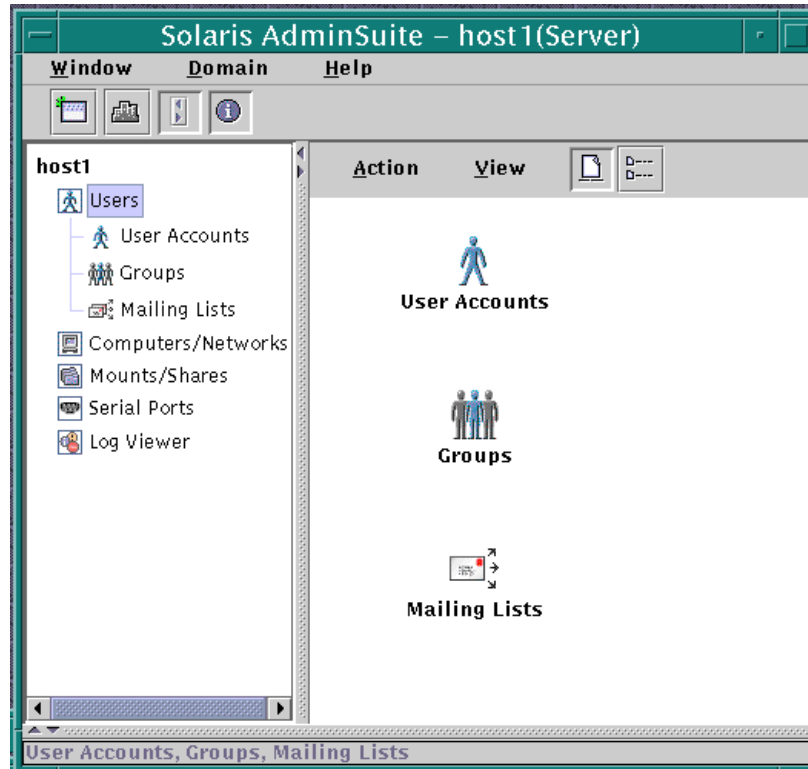


Figure 10-29 Solaris AdminSuite –Users Window

You are prompted to set the initial view for displaying the available user accounts.

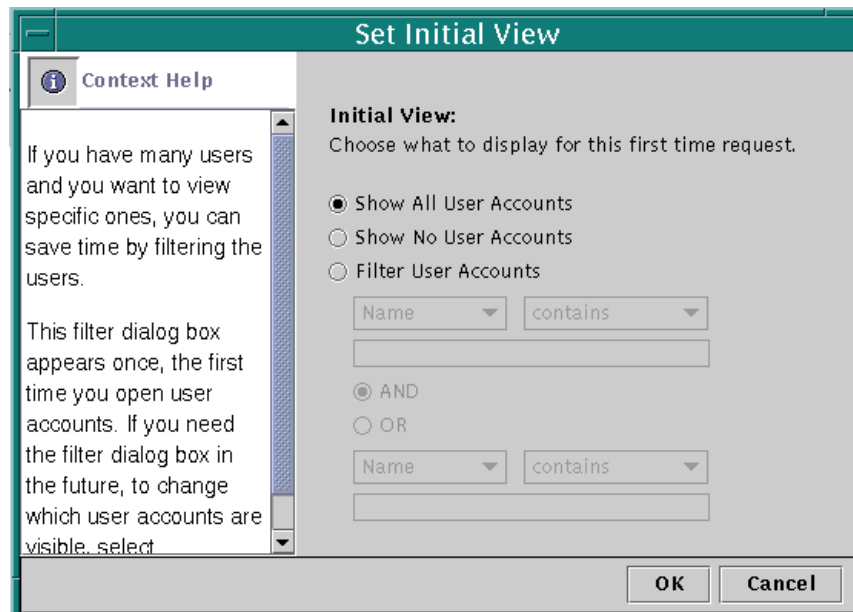


Figure 10-30 Solaris AdminSuite – Set Initial View Window

To manipulate user accounts, perform the following steps:

1. Click your desired view.
2. Click OK.

If you select the option to show all user accounts, you are presented with the iconized accounts, as illustrated in Figure 10-31.

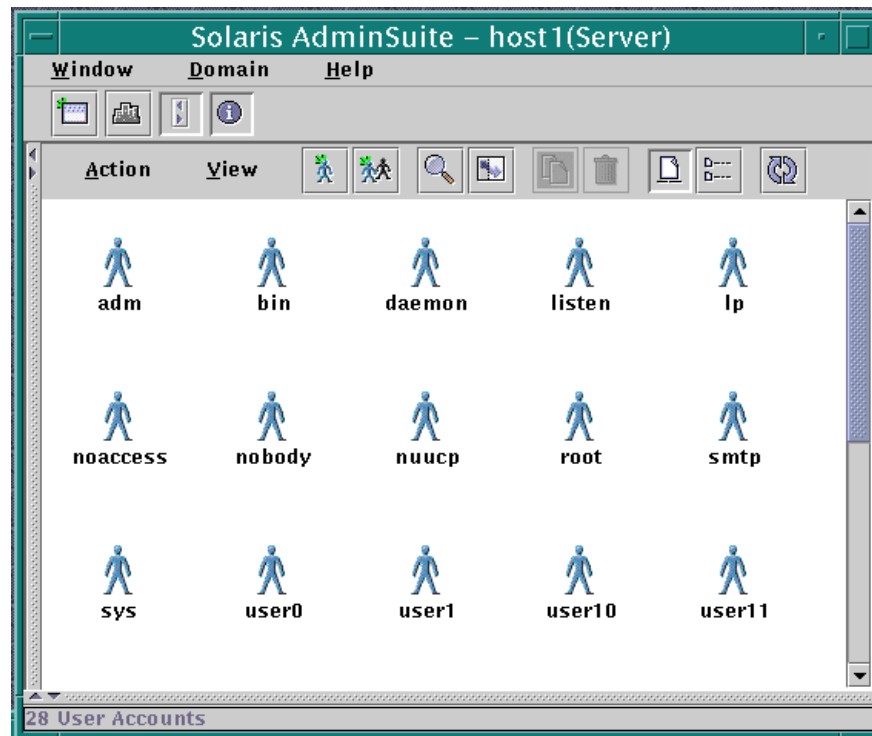


Figure 10-31 Solaris AdminSuite – All Users Window

The Menu/Icon bar changes to support the current desktop workspace environment. The View menu on this display shows the current configuration of the desktop. The icons show the user names, but you can display more information about each user by selecting a detailed view, as follows:

1. Click View►Details.

The new display contains a text-oriented listing of all current users that are visible to host1.

You can alter this display using the sort selection under the View menu, as follows:

2. Click View►Sort by►*sort_characteristic*.

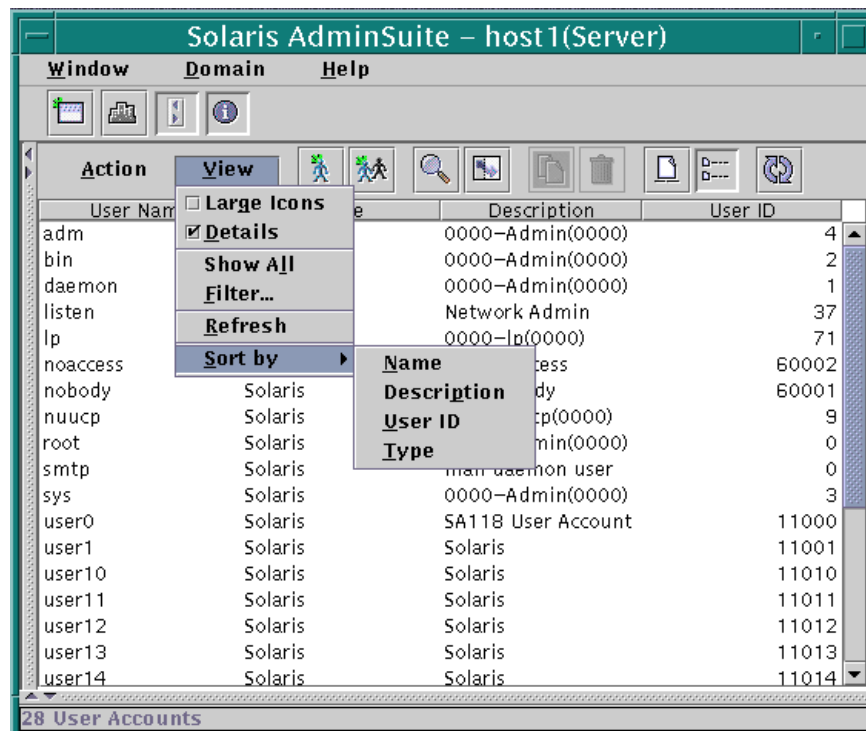


Figure 10-32 Solaris AdminSuite – Users Sort By Menu

Adding Users

Just as the View menu provides the ability to alter how the current users are displayed, an Action menu provides various methods of manipulating the User accounts feature.

To add a new user, perform the following steps:

1. Click Action►Add User.

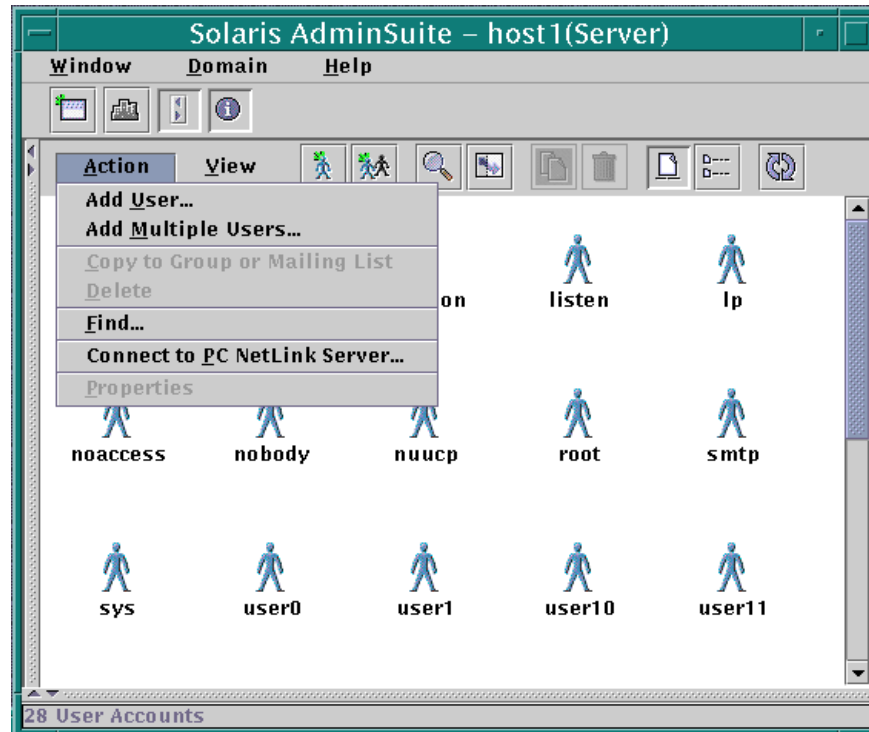


Figure 10-33 Solaris AdminSuite – Users Action Menu

A series of input windows are displayed that prompt for the information describing the new user account.

The left side of the display lists the steps that are required to accomplish the selected operation. Each step in the process displays an additional display, with the corresponding process step highlighted on that display.

2. Enter the new user name.

Note – The naming conventions for users differ according to organizational rules.

3. Enter a brief description of the user account.

Note – Descriptions are optional, but they are usually recommended.

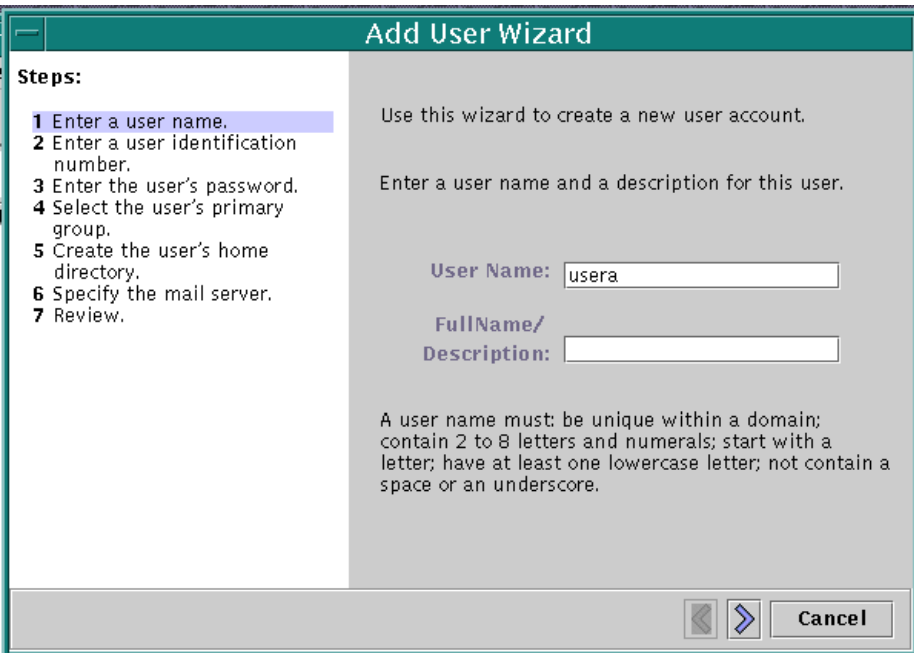


Figure 10-34 Add User Wizard

4. Click the ➤ symbol.

The user ID (UID) number is the identifier by which this user will be known to the system.

5. Add a unique user identification number to the account.

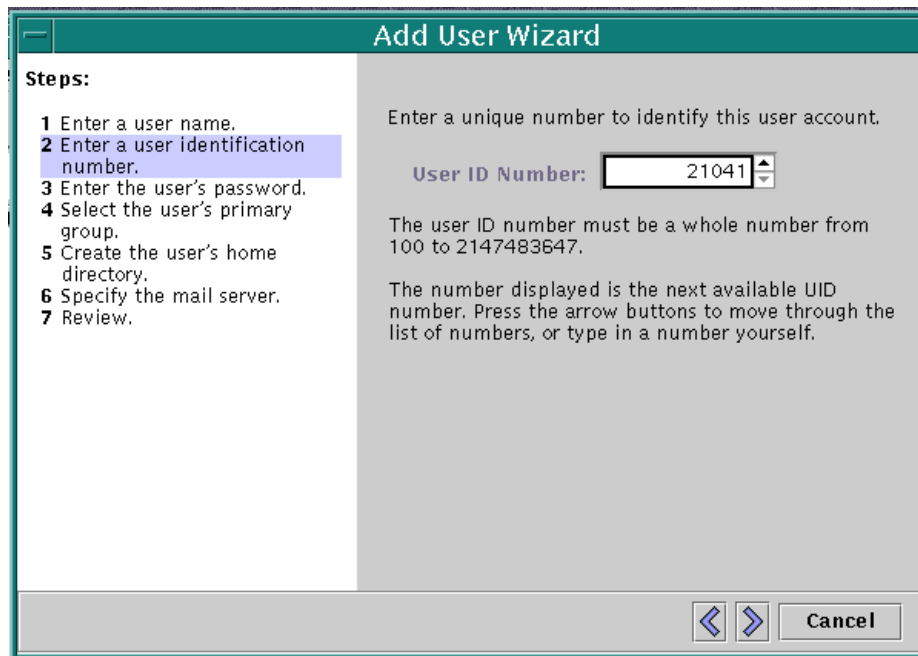


Figure 10-35 Add User Wizard – Add User ID

6. Click the ➤ symbol.

You can set the user's initial password within this display, or you can require the user to set their own password during their first login session.

7. Click User Must Use This Password At First Logon:
8. Enter a password in the Password field
9. Re-enter the password in the Confirm Password field

Note – The password displays as a series of asterisks for security purposes.



Add User Wizard

Steps:

- 1 Enter a user name.
- 2 Enter a user identification number.
- 3 Enter the user's password.
- 4 Select the user's primary group.
- 5 Create the user's home directory.
- 6 Specify the mail server.
- 7 Review.

You may choose to let the user set the password, or set it yourself now.

☐ User Must Set Password At First Logon

☒ User Must Use This Password At First Logon:

Password: *****

Confirm Password: *****

The password must contain a combination of 6 to 16 characters. Within the first 6, there must be at least 2 alphabetic characters and at least one number. The password is case sensitive.

< > Cancel

Figure 10-36 Add User Wizard – Add User Password

10. Click the ➤ symbol.

Select the primary group to which this user is to belong.

11. Click Primary Group.

12. Select the user's group from the drop-down list.

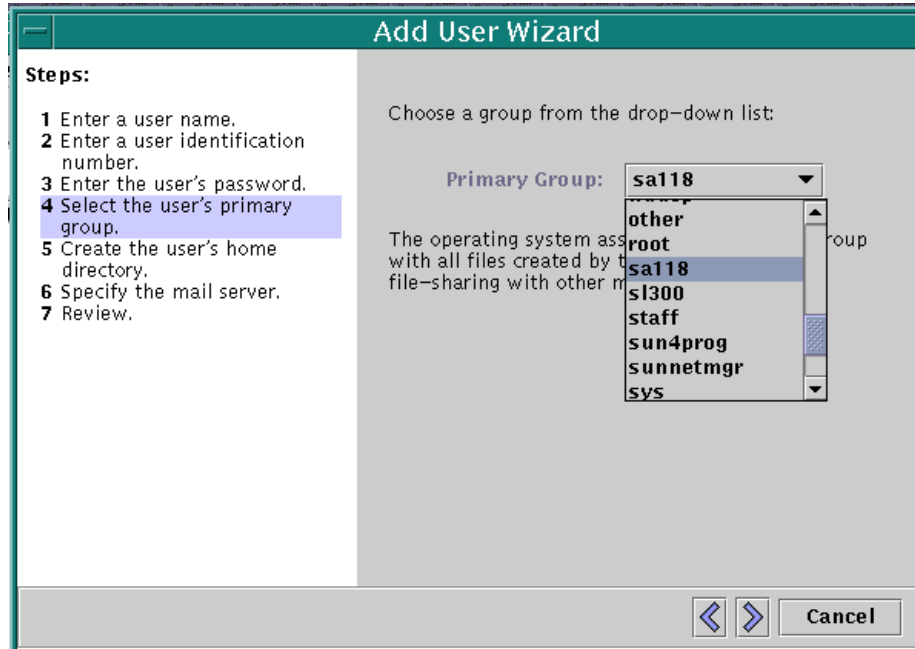


Figure 10-37 Add User Wizard – Add User Group Selection

13. Click the ➤ symbol.

Each user requires a home directory where the user is placed upon login. If the home directory server is other than this server, you must ensure Solaris AdminSuite 3.x is running on the home directory server.

14. Enter the home directory path minus the user name.

Note – The user name is appended to the end of the path before the directory is created.

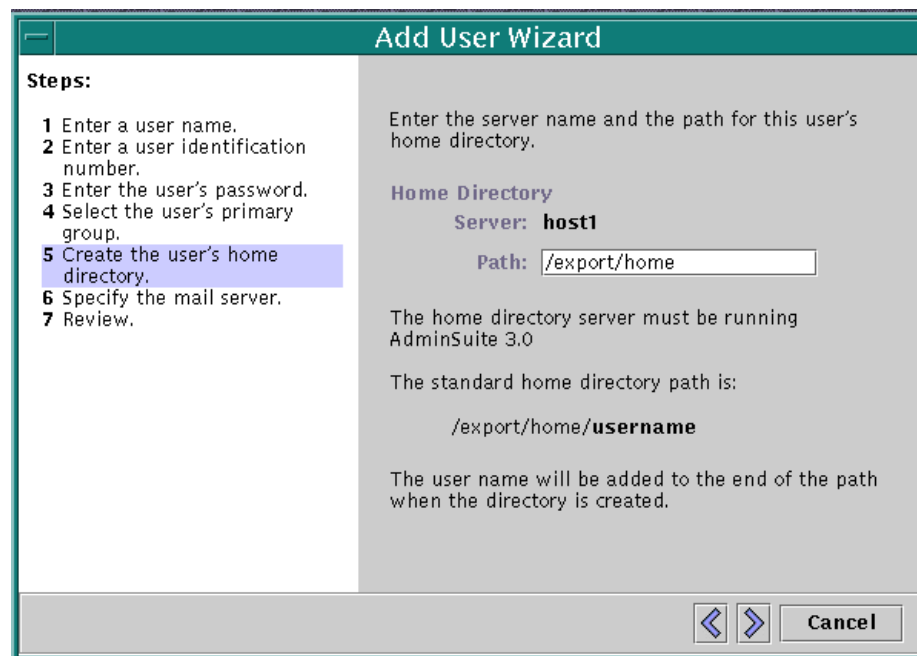


Figure 10-38 Add User Wizard – Add User Home Directory

15. Click the ➤ symbol.

This window is for informational purposes only.



Figure 10-39 Add User Wizard – Add User Mail Server

16. Click the ➤ symbol.

Before to actually creating the user account, Solaris AdminSuite re-displays your sections, and provides an opportunity to correct the errors.

17. If there are errors, return to the appropriate display to correct them.

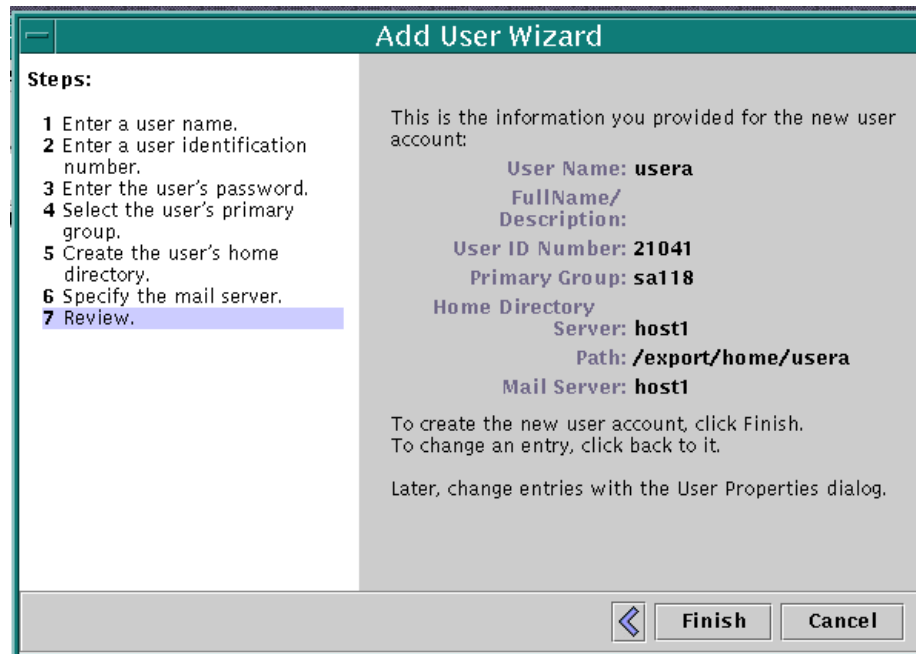


Figure 10-40 Add User Wizard – Add User Attributes Confirmation

18. If there are no errors, click Finish.

After you create the new user, you can confirm the operation was successful by listing the users.

Note – If your view option is set for Large Icons, your display might differ from this screen capture; however, the icon for `usera` should still be present.

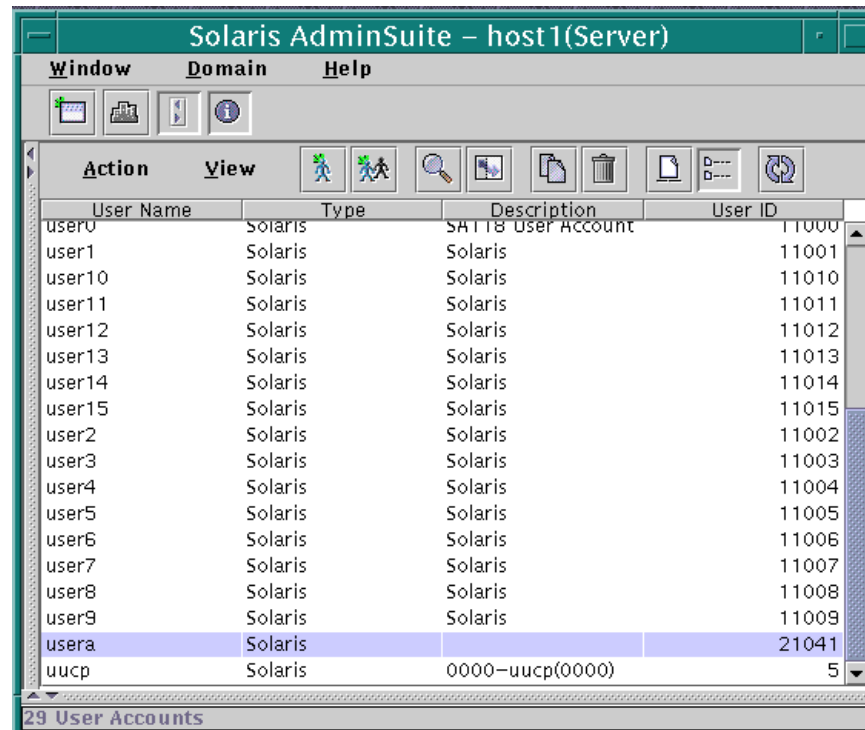


Figure 10-41 Solaris AdminSuite – New User Listing

You can now use the new `usera` login.

Viewing Groups

Viewing groups begins by either clicking the Groups icons on the desktop in the upper right segment, or by clicking on the Groups icon in the Users menu on the upper left segment of the Solaris AdminSuite display.

In this example, you would click the Groups icon in the Users menu on the upper left segment of the Solaris AdminSuite display, because the list of users (from the previous example) is displayed on the right.

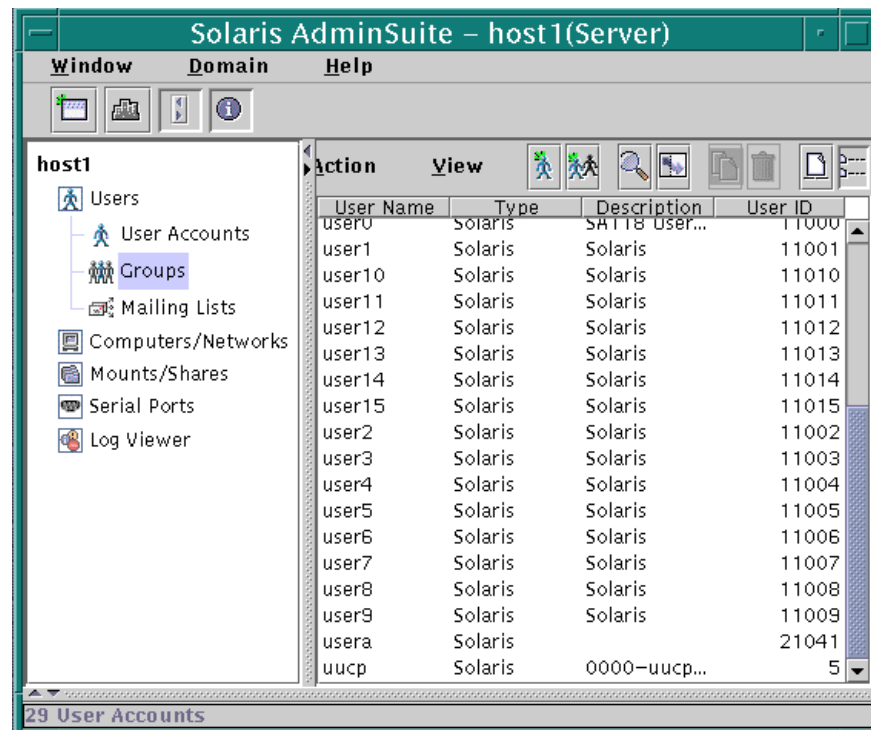


Figure 10-42 Solaris AdminSuite – Users Group Selection

To set the initial view for group accounts, perform the following steps:

1. Double-click the Groups Accounts icon.

You are prompted to set the initial view for displaying the available groups.

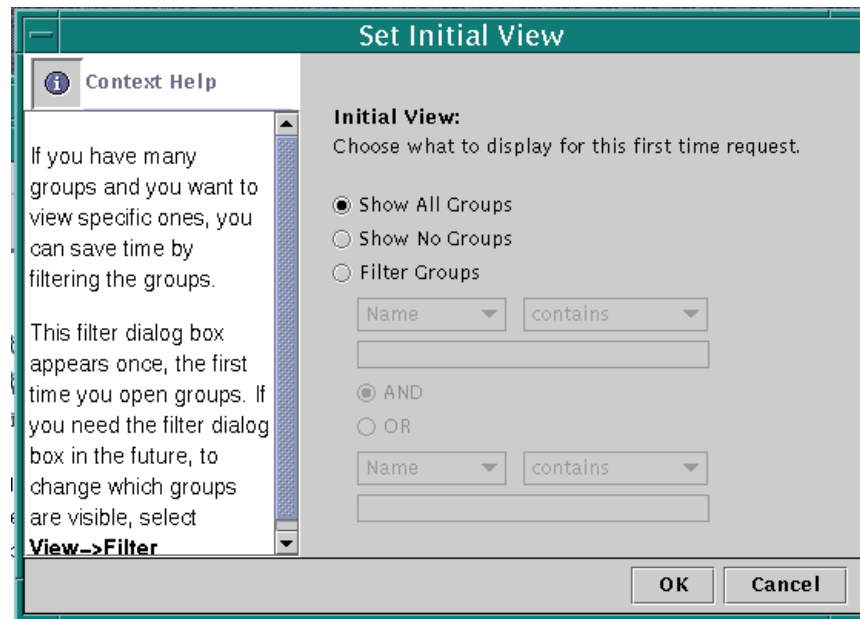


Figure 10-43 Set Initial View Window

2. Click your desired view.
3. Click OK.

The initial view of the available groups is displayed alphabetically by name.

Note – You must remember that your earlier selection of Domain ► Security Policies affects whether you search through local server files first or use the naming services.

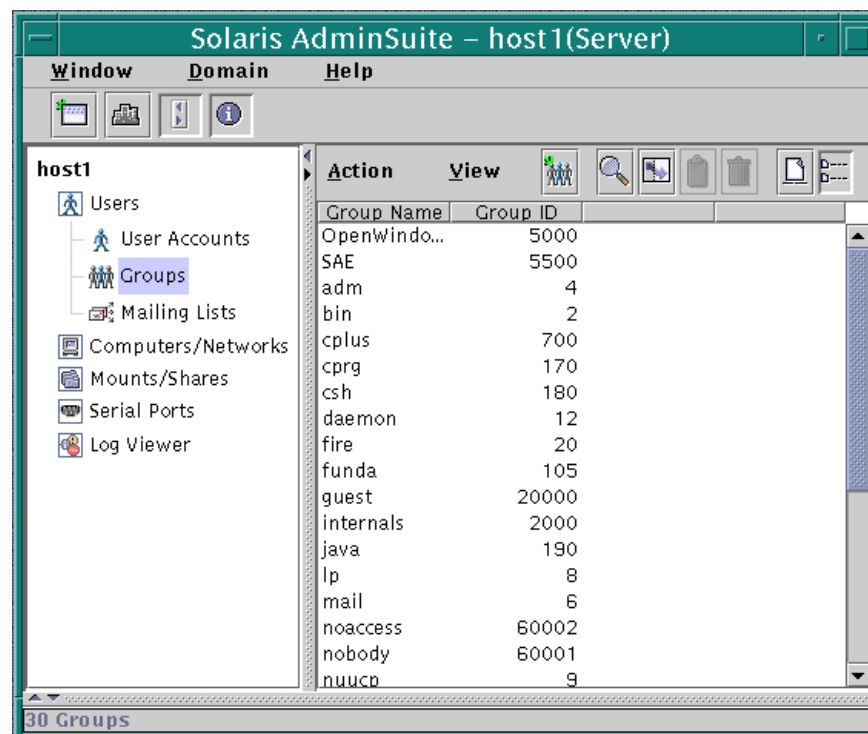


Figure 10-44 Solaris AdminSuite – Groups Window

Many administrators find a numerically sorted group file easier to read. You can sort the displayed groups by group number, as follows:

4. Click View►Sort by►Group ID.

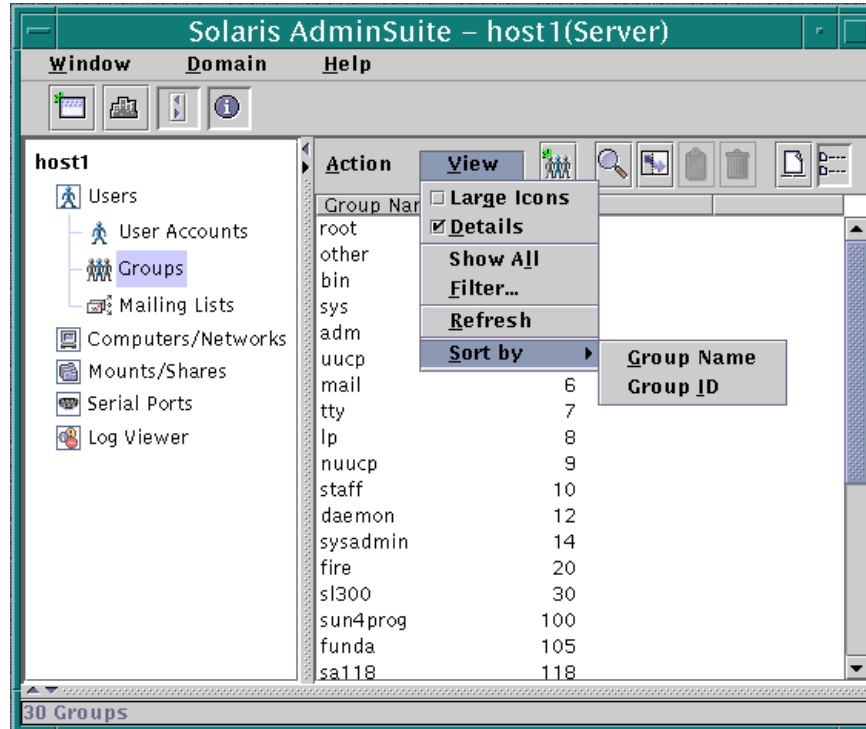


Figure 10-45 Solaris AdminSuite – Groups View Sort By Menu

The group display also contains an action menu. To make the menu item active, you must first select an existing group; otherwise, the only active menu item is the Add Group selection.

Adding Groups

To add a new group, perform the following steps:

1. Click Action►Add Group.

The Group Action menu is displayed.

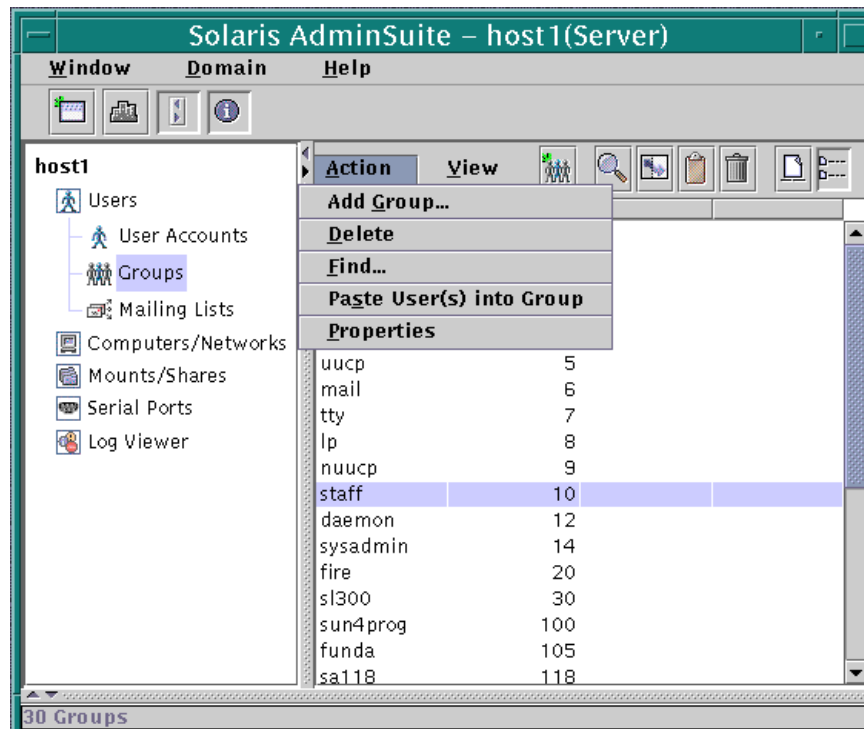


Figure 10-46 Solaris AdminSuite – Groups Action Menu

2. Enter the new group name.

You are presented with an available (unused) group ID number.

3. You can accept this group ID number or choose another.

Note – This group name and group ID number must be unique.

4. Enter the user names of the group members in the appropriate space.

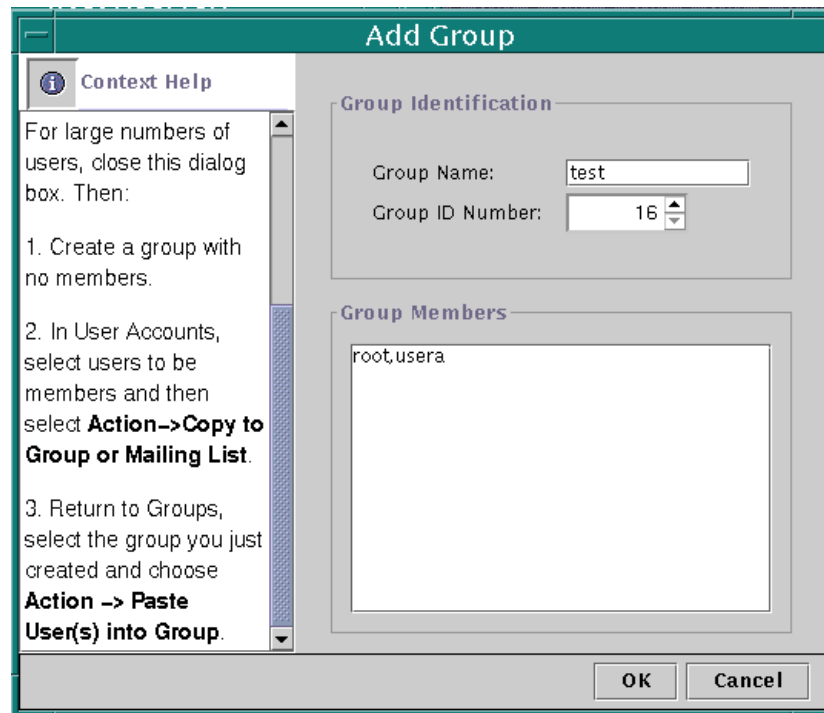


Figure 10-47 Add Group Window

5. Click OK.

To confirm the creation of the group,

6. Click the Groups icon in the upper-left window.

Note – When you reference the groups file using the menu icon, the display's view format returns to alphabetical.

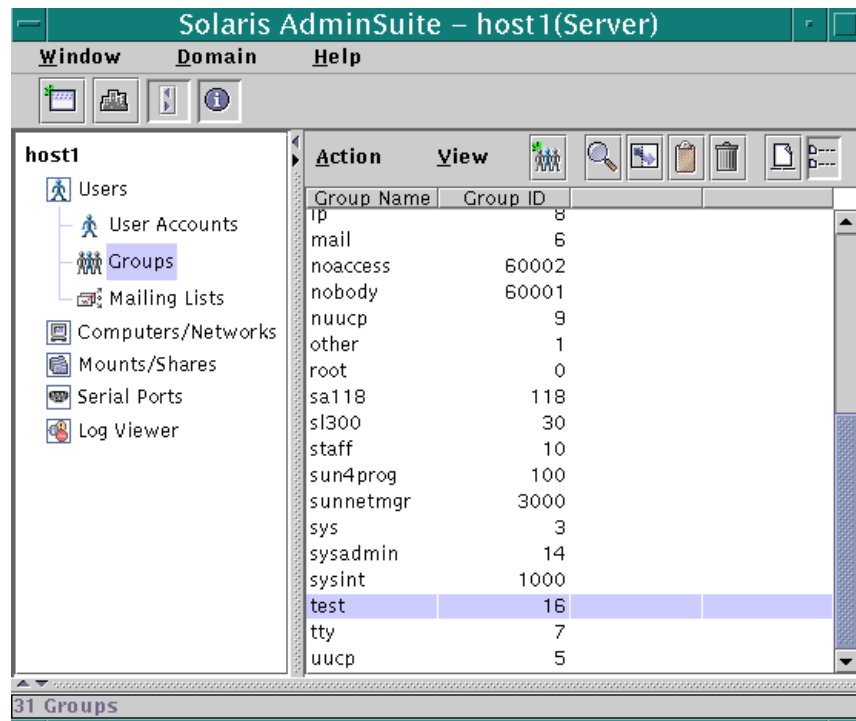


Figure 10-48 Solaris AdminSuite – New Group Listing

Modifying Groups

To add users to an existing group, perform the following steps:

1. Highlight the group in the groups display (select sysadmin).
2. Click Action►Properties.

The Group Properties window that corresponds to the highlighted group is displayed.

3. Enter the additional group members into the appropriate location.

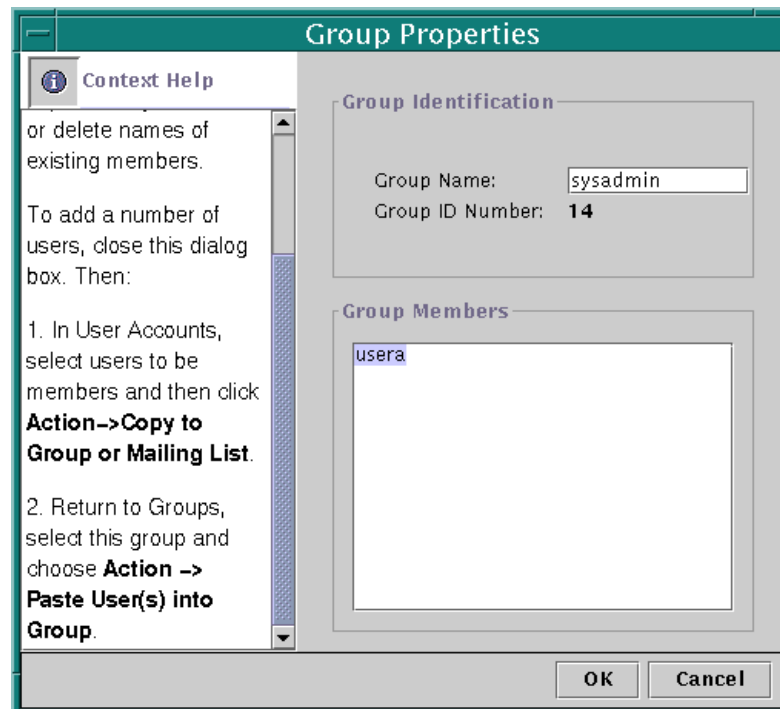


Figure 10-49 Group Properties Window

4. Click OK.

Computers/Networks Feature

Solaris AdminSuite has a feature for manipulating the Computers/Networks information. Local hosts are selected using the Computers selection on the sub-menu.

1. Click Computers/Networks to display the menu selections.
2. Click the Computers selection, and answer the prompts for the initial view.

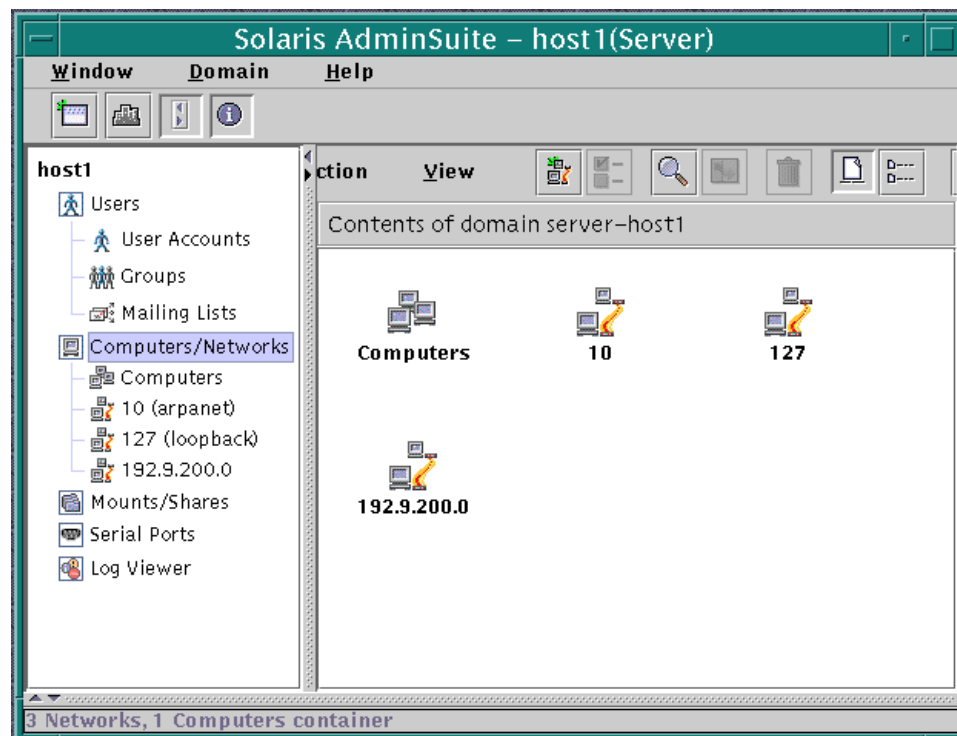


Figure 10-50 Solaris AdminSuite – Computers/Networks Selections

Depending on the size of your subnet, you can select to Show All Computers or some subset of the available computers on the subnet

To choose a subset of the net,

3. Click Filter Computers.
4. Enter the desired filter.

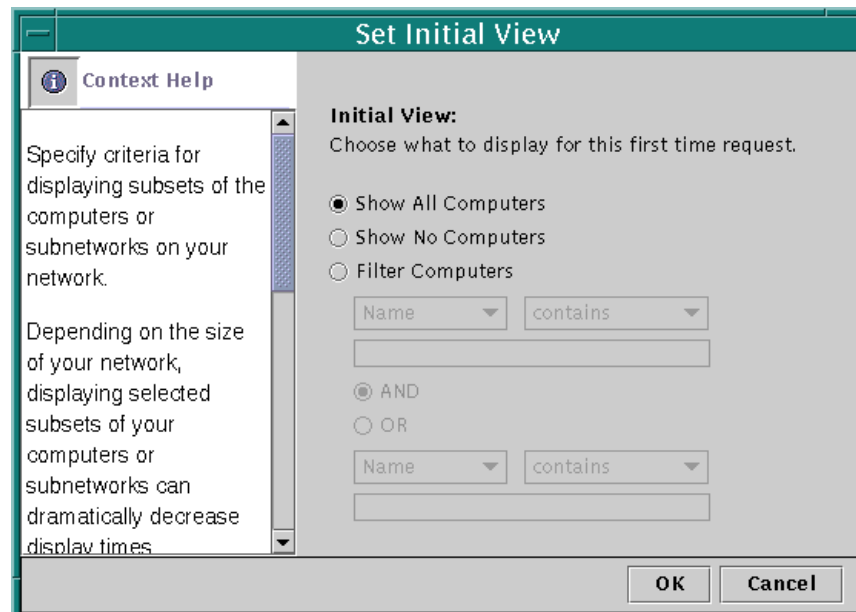


Figure 10-51 Set Initial View Window

5. Click OK.

The following display was generated from a complete listing of a small network.

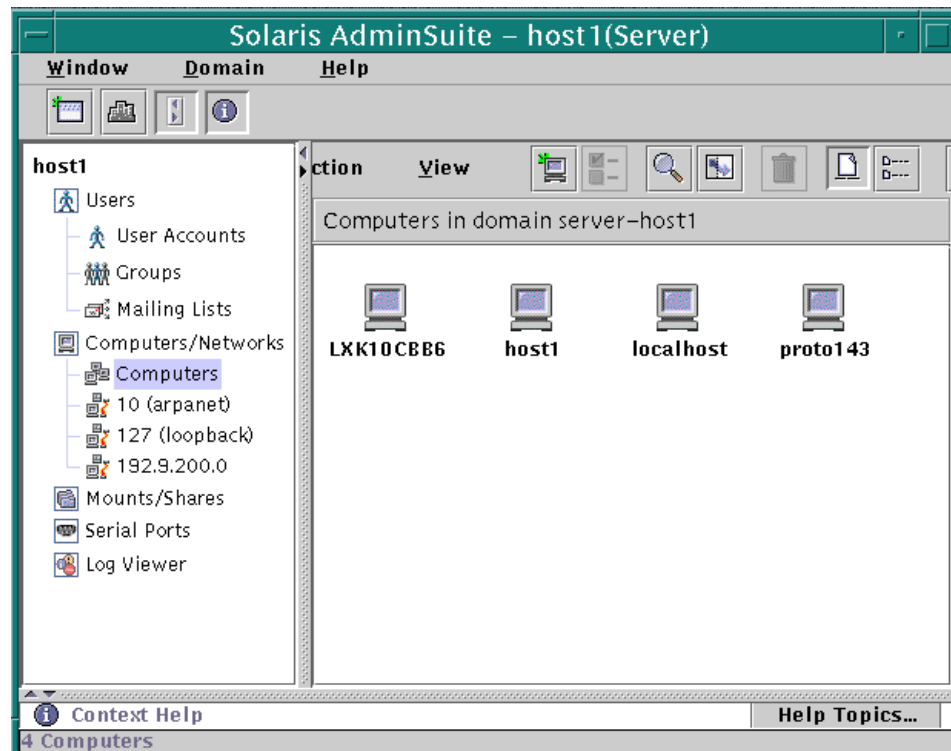


Figure 10-52 Solaris AdminSuite – Computers (Local Host) Window

6. Click View to Open the View menu.

The view menu for hosts shows this subnet is sorted by the name of the computer. Other possible sort categories include:

- ▼ IP addresses
- ▼ Descriptions
- ▼ Aliases
- ▼ Ethernet addresses

You can sort the members either in ascending or descending order.

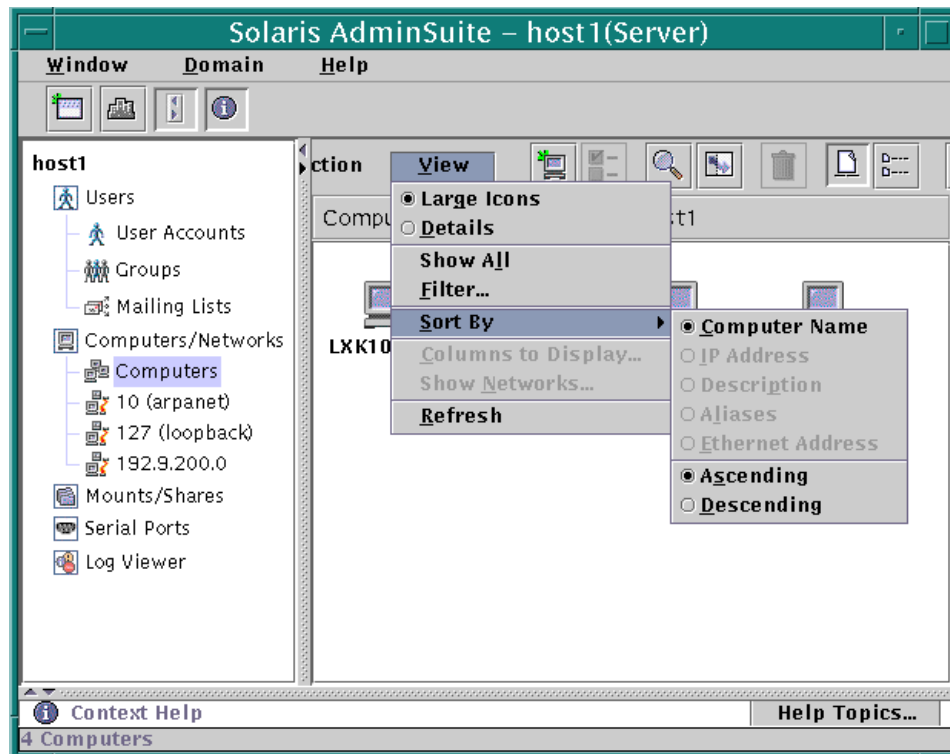


Figure 10-53 Solaris AdminSuite – Computers View Sort By Menu

Adding a Host

Similar to the other Solaris AdminSuite features, to add another host to the subnet, you need to use the Action menu. Perform the following steps:

1. Click Action►Add Computer.

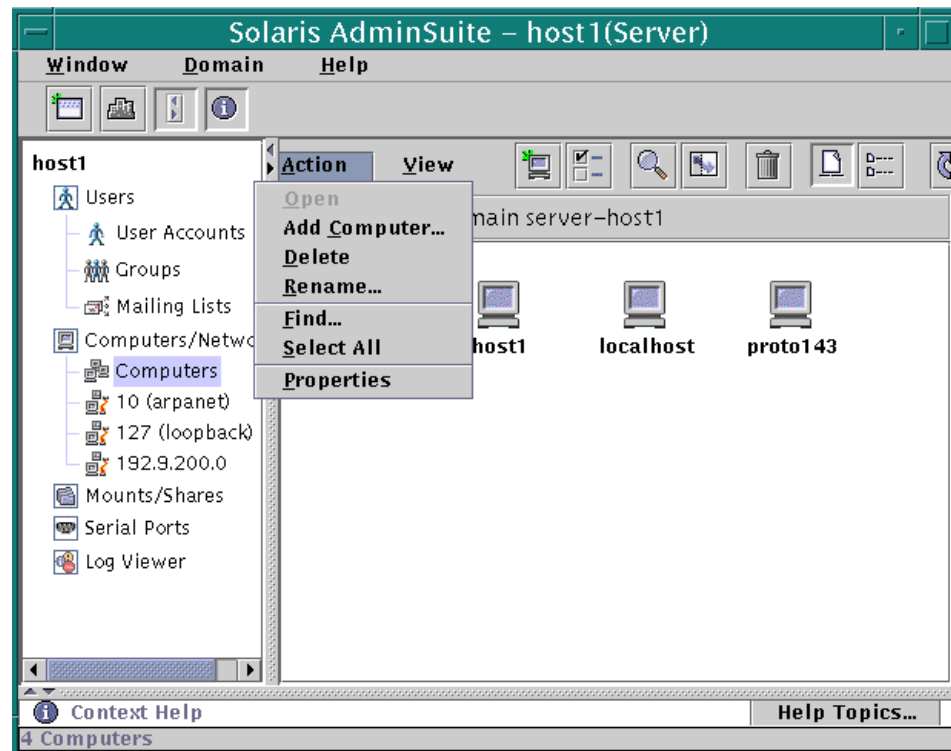


Figure 10-54 Solaris AdminSuite – Computers Action Menu

2. In the Name field, enter the new host name.
3. In the IP Address field, enter the Internet Protocol address.

Note – The additional information fields are optional. You can add aliases, descriptions, and Ethernet addresses. This information can make future administration exercises easier; however, the added information is not required.

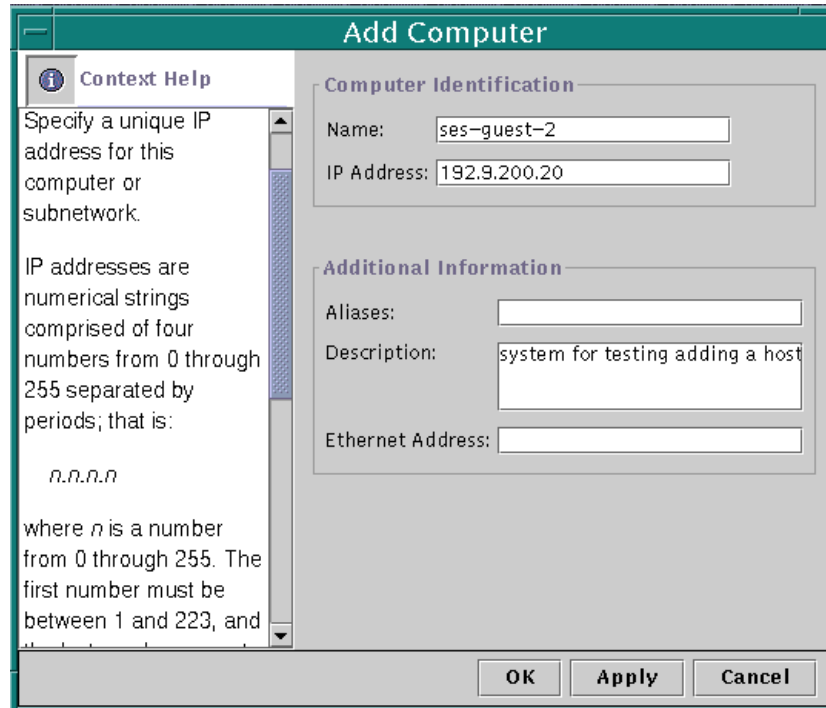


Figure 10-55 Add Computer Window

4. Click OK.

5. Verify that the new host was added by re-displaying the list of subnet hosts.

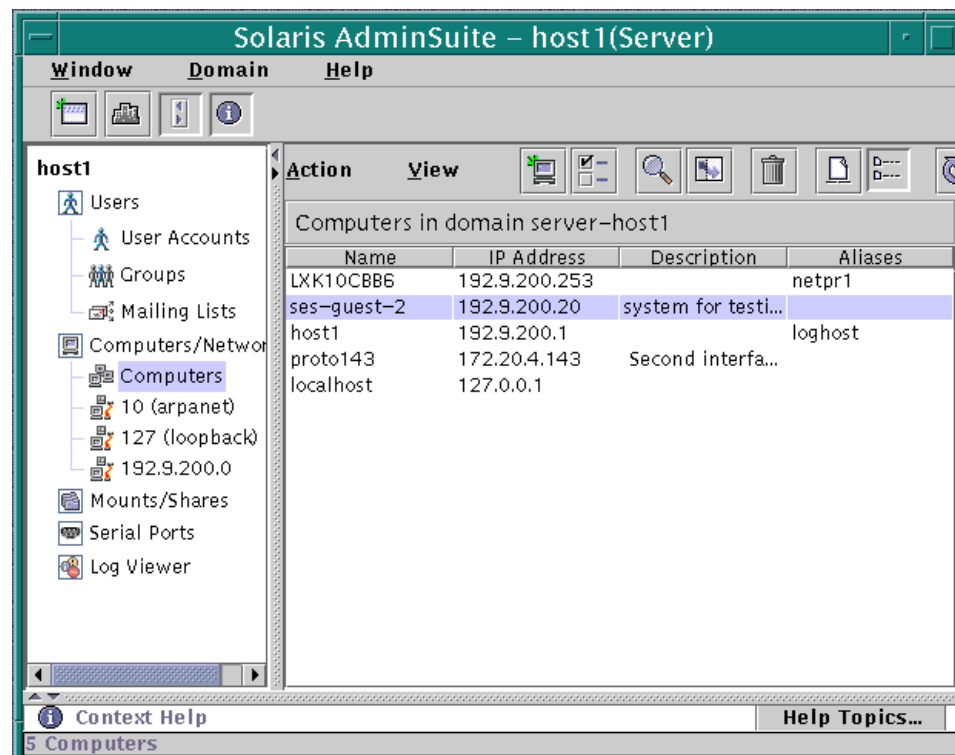
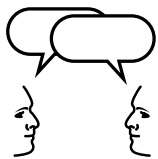


Figure 10-56 Solaris AdminSuite – New Computer Confirmation Window



Discussion – What view options were used to create the display in Figure 10-56?

Renaming a Host

To change the name of an existing host, perform the following steps:

1. In the Computers (host list) window, click the host that you want to rename.
2. Click Action►Rename.

The Rename Computer window is displayed as shown in Figure 10-57).

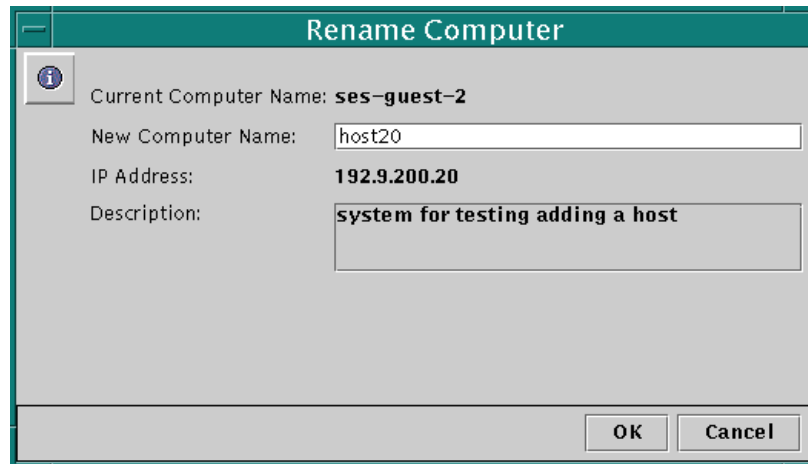


Figure 10-57 Rename Computer Window

3. In the New Computer Name field, enter the new host name.
4. Click OK.

5. Display the list of hosts (showing the modified host name).

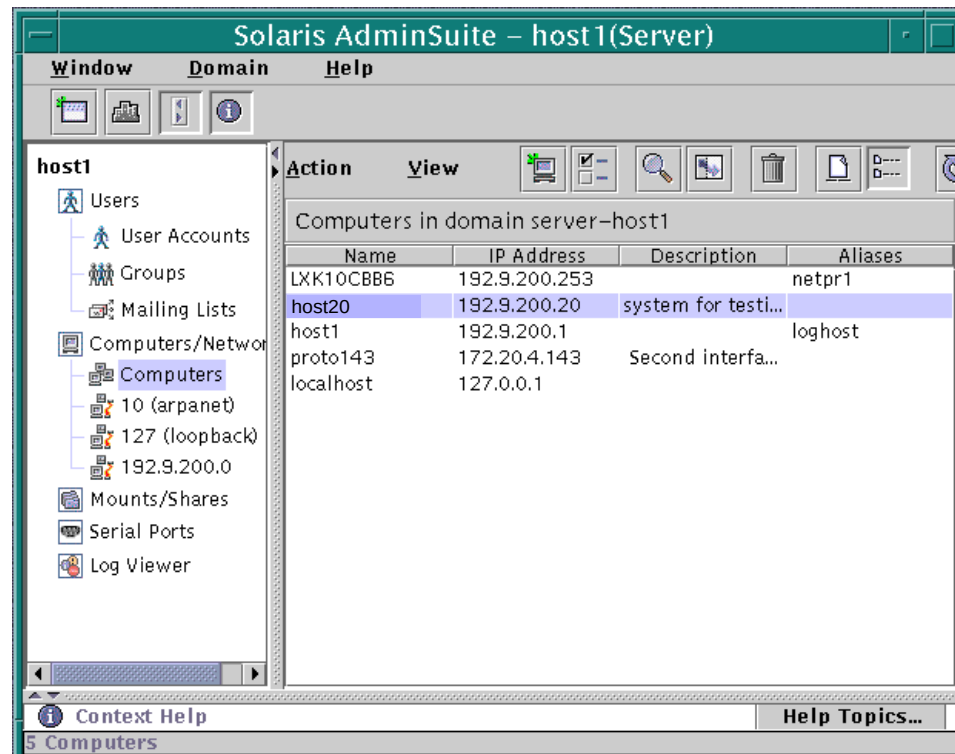


Figure 10-58 Solaris AdminSuite – Renamed Computer Confirmation Window

Mounts/Shares Feature

You can access the Solaris AdminSuite's Mounts/Shares feature from the main menu (the same as all of the other Solaris AdminSuite features). The Mounts/Shares feature includes the following:

- Mounts
- Shares
- File System Usage

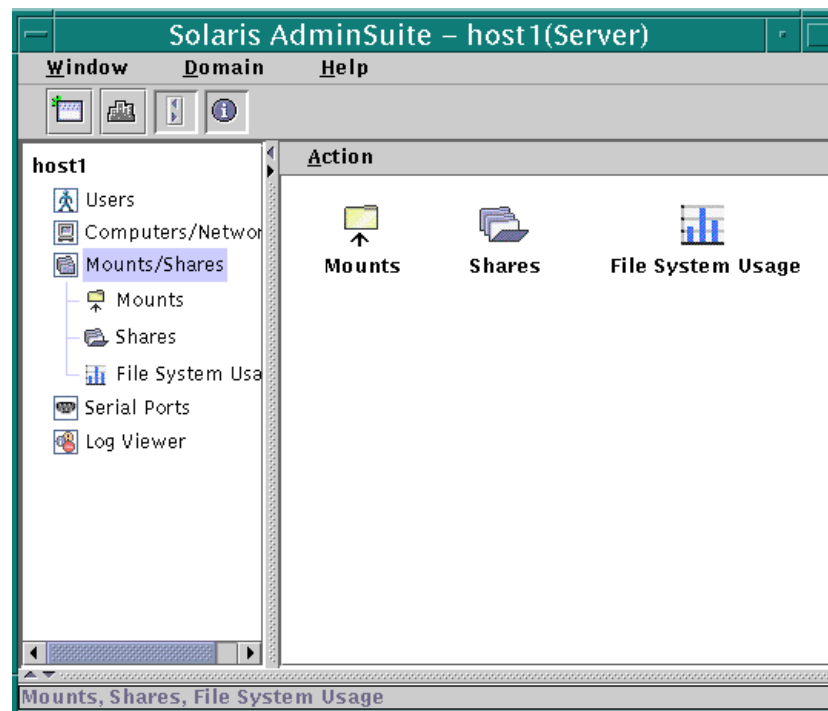


Figure 10-59 Solaris AdminSuite – Mounts/Shares Window

The Mounts feature displays all file system mounts that are visible to the server that is named in the title bar. To use the Mounts/Shares feature, perform the following steps:

1. To display currently mounted resources, click Mounts.

Note – The resources are initially displayed alphabetically by mount point directory.

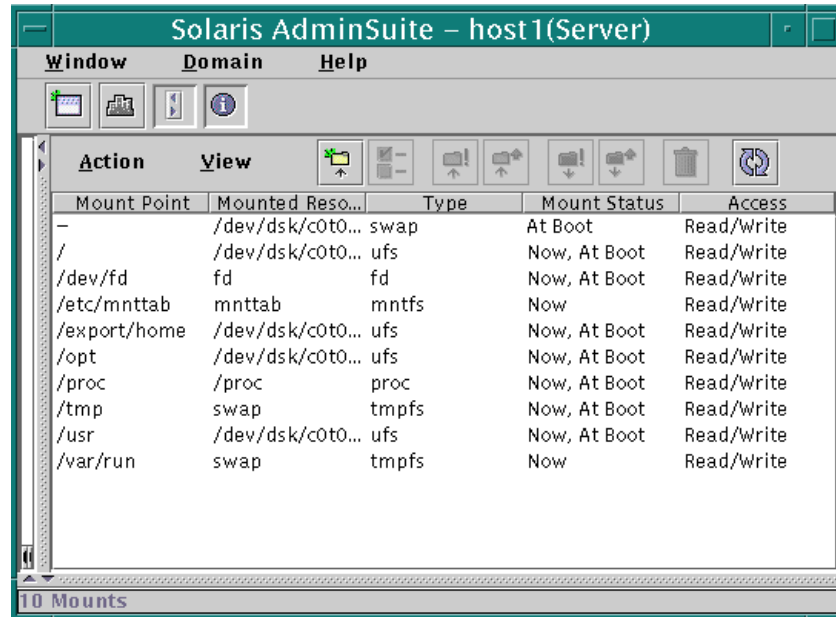


Figure 10-60 Solaris AdminSuite – Mounted Resource Window

You can change the display order using the View menu.

1. Click View►Sort by►*sort_category*.

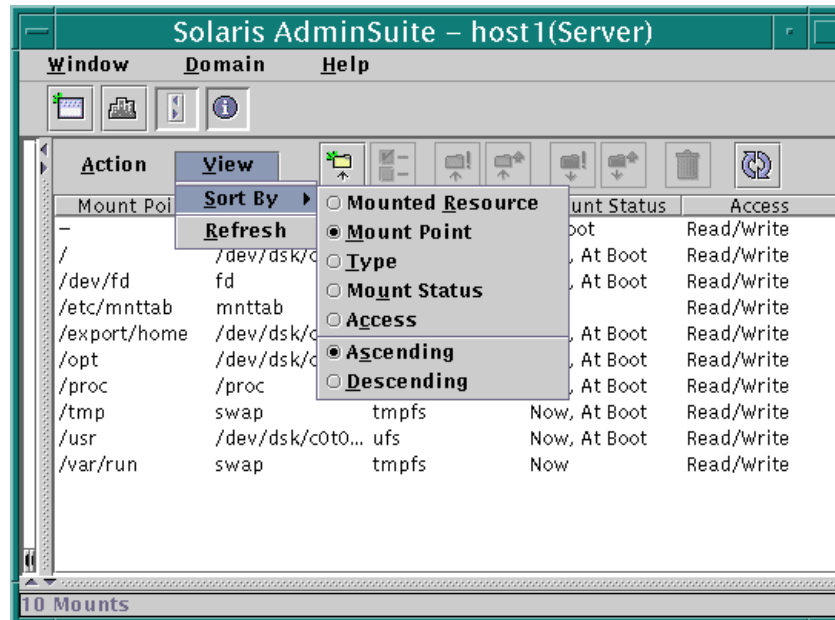


Figure 10-61 Solaris AdminSuite – Mounted Resource View Sort By Menu

To acquire statistics on mounted file systems, complete the following steps:

2. Click the file system (/mnt in this example).

3. Click Action►Properties.

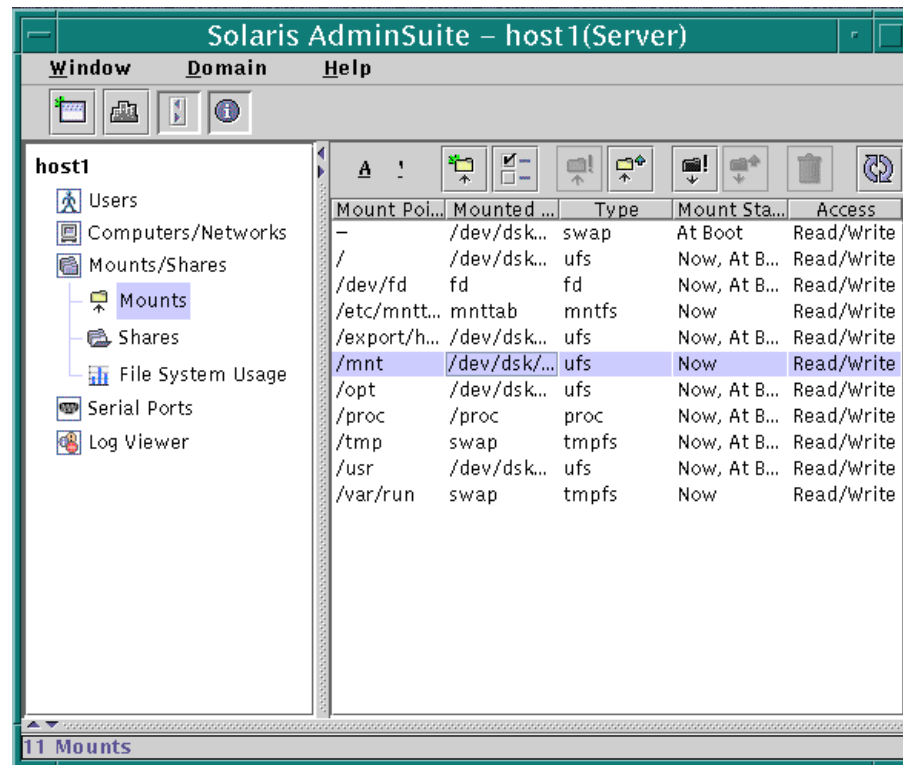


Figure 10-62 Solaris AdminSuite – /mnt File System Selected

The mount properties are reported in two pages. The first page contains the mount point name and its corresponding slice. Additionally, the mount status (for example, Now or Mount at Boot), and the access mode (for example, Read/Write or Read Only) are reported.

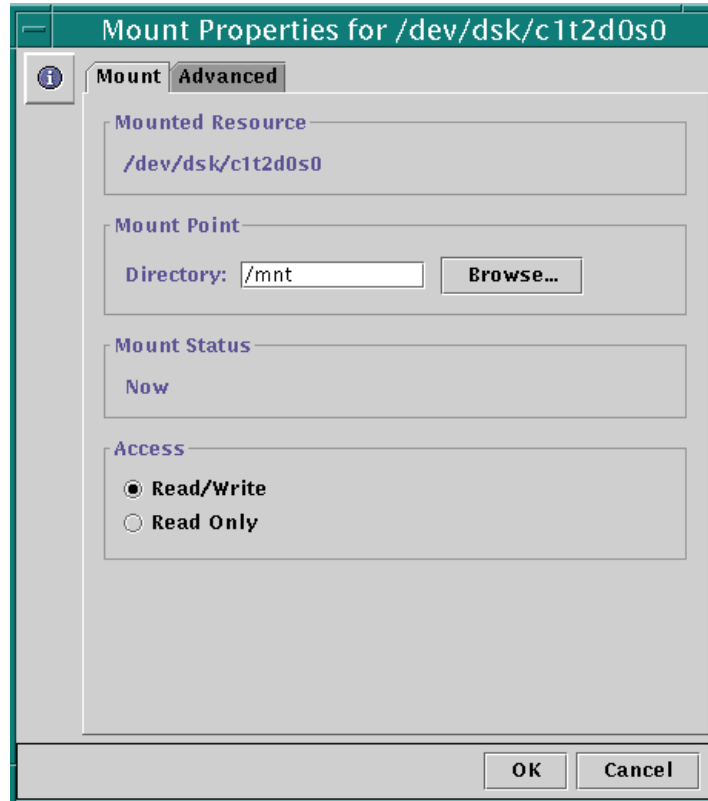


Figure 10-63 Mount Properties Window

The second page displays the mount options, and it reports how damaged file systems are handled. For damaged file systems, you can also select how many times damage to a file system can be detected in a given time frame, before the file system is shut down.

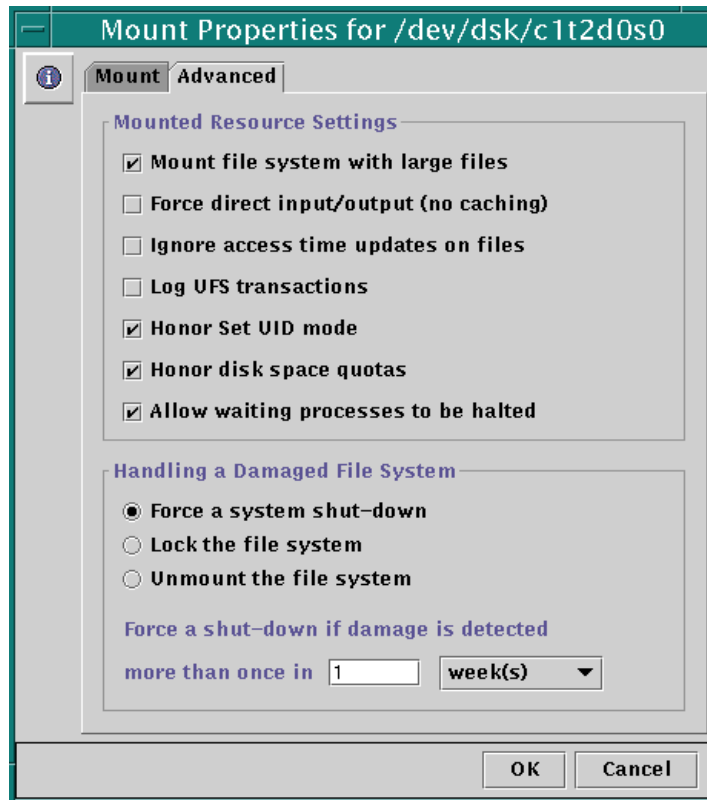
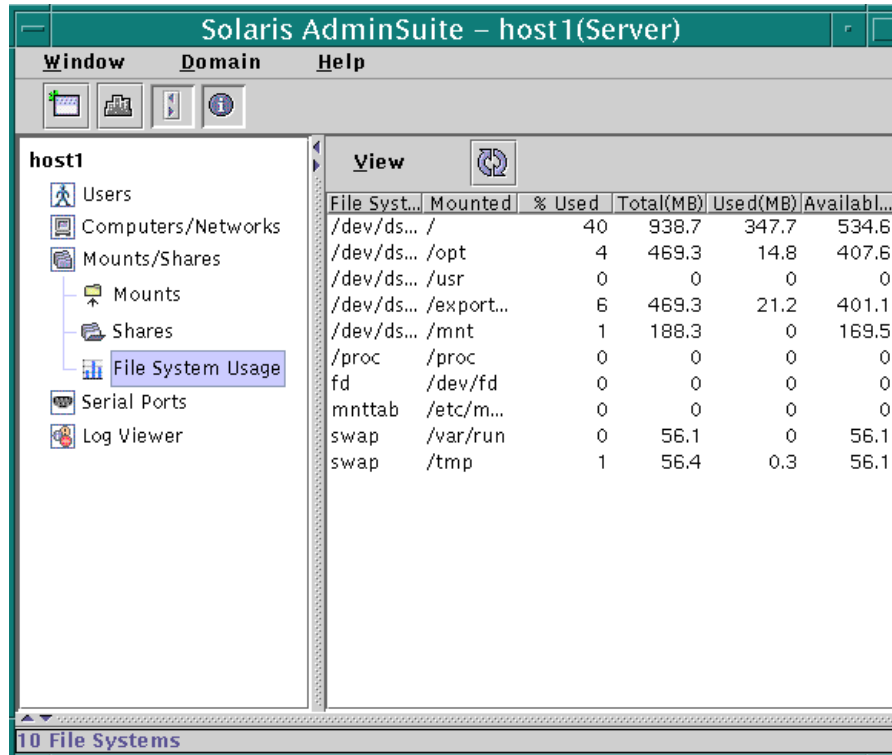


Figure 10-64 Mount Properties Window, Advanced

File System Usage

The output of the File System Usage menu item is similar to the output of the `df -k` command. This window displays the mounted file systems' mount points, the total amount of allocated space, the total space currently in use, and the amount of remaining space.



File Syst...	Mounted	% Used	Total(MB)	Used(MB)	Availabl...
/dev/ds...	/	40	938.7	347.7	534.6
/dev/ds...	/opt	4	469.3	14.8	407.6
/dev/ds...	/usr	0	0	0	0
/dev/ds...	/export...	6	469.3	21.2	401.1
/dev/ds...	/mnt	1	188.3	0	169.5
/proc	/proc	0	0	0	0
fd	/dev/fd	0	0	0	0
mnttab	/etc/m...	0	0	0	0
swap	/var/run	0	56.1	0	56.1
swap	/tmp	1	56.4	0.3	56.1

Figure 10-65 Solaris AdminSuite – File System Usage Window

The File System Usage window contains a sort mechanism similar to those in the other Solaris AdminSuite features; however, the sort fields correspond to the data fields displayed within this window.

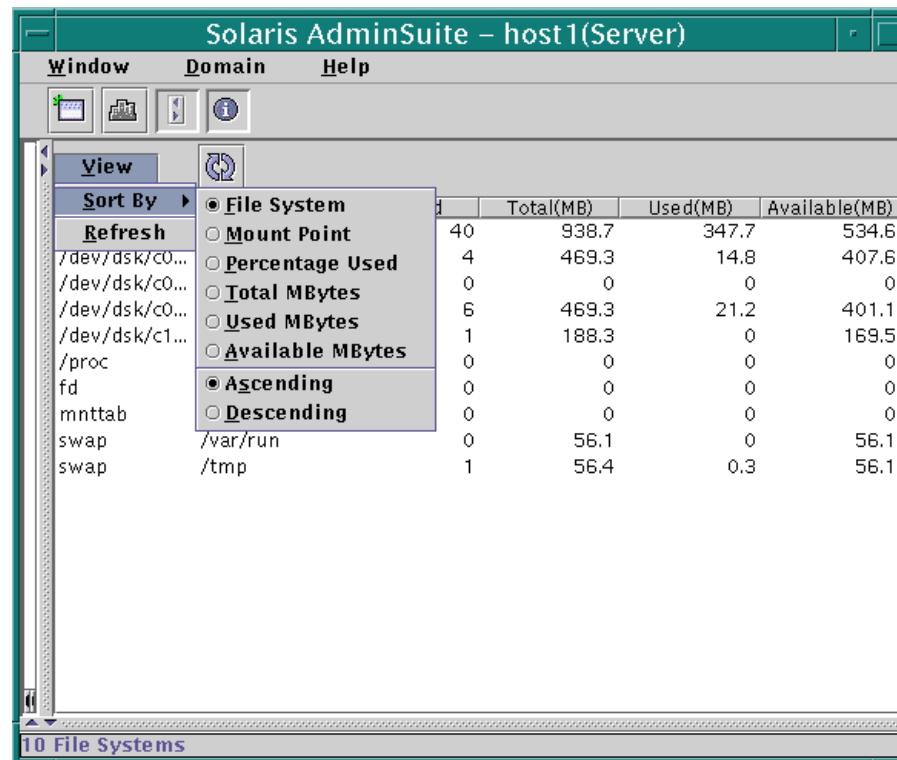


Figure 10-66 Solaris AdminSuite – File System Sort Criteria

Serial Ports Feature

Selecting the Serial Ports menu item allows you to configure the serial ports on the system.

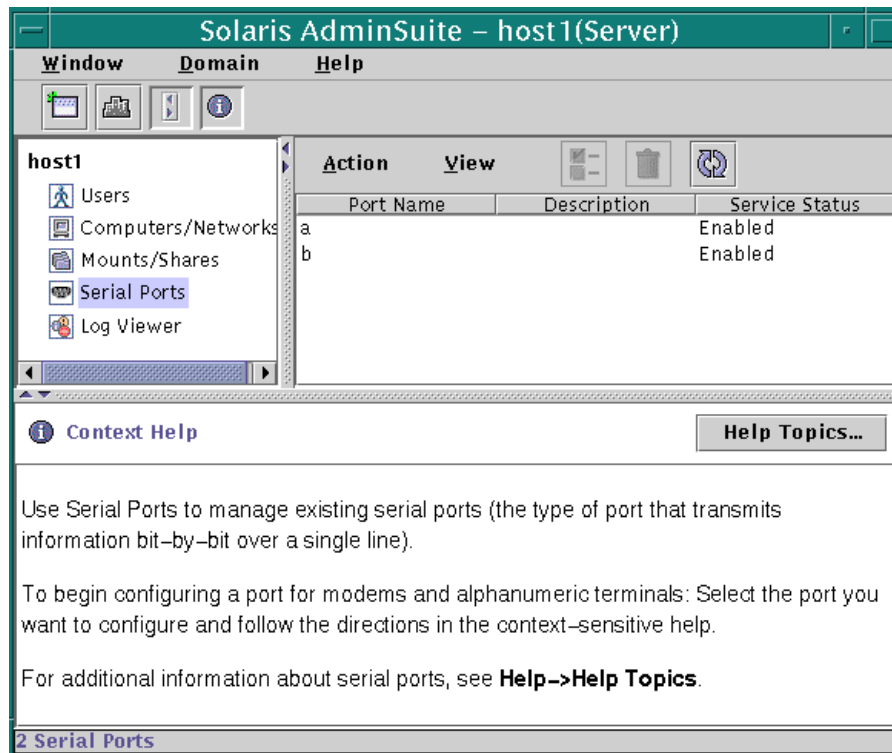


Figure 10-67 Solaris AdminSuite – Serial Ports Window

The ports are initially displayed alphabetically, but you can alter the display by performing the following steps:

1. Click View►Sort by►*sort_characteristic*.

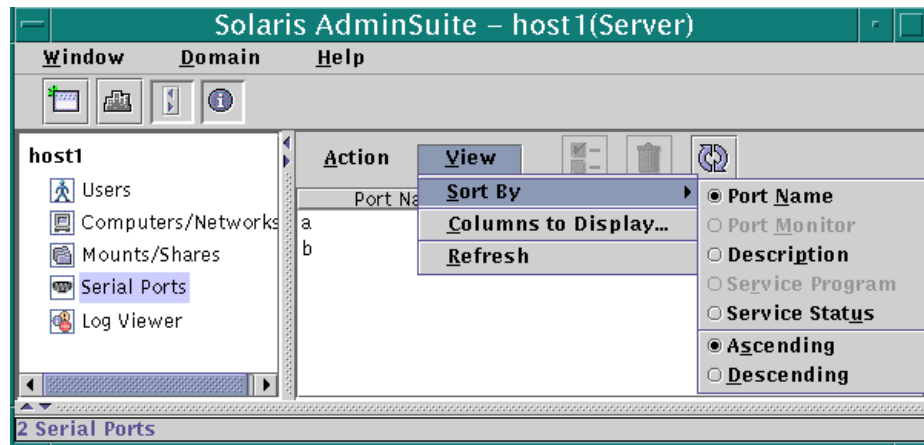


Figure 10-68 Solaris AdminSuite – Serial Ports View Sort By Menu

To alter the types of information displayed, perform the following steps:

1. In the Serial Ports display, click View►Columns to Display.

The Columns to Display window is displayed, as shown in Figure 10-69.

2. Click any item in the Available Columns list, and click Add.
3. Click any item in the Columns to Display, and click Remove.

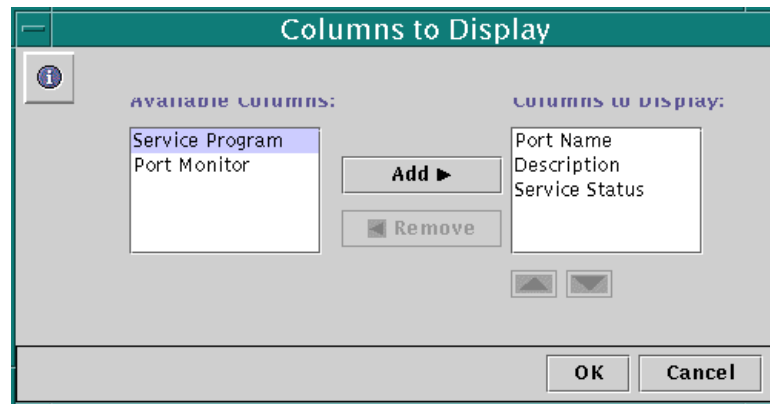


Figure 10-69 Columns to Display Window

4. Click OK.

Figure 10-70 illustrates a serial ports view with all columns displayed.

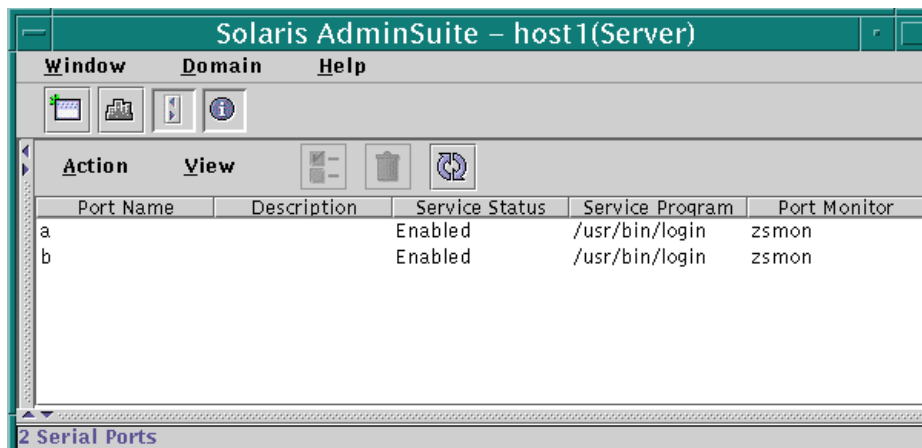


Figure 10-70 Solaris AdminSuite – Serial Ports Window

Configuring Serial Ports

Configuring a serial port differs only slightly from configuring other Solaris AdminSuite features.

To configure any serial port, perform the following steps:

1. Click the serial port that you want to configure.
2. Click Action►Configure %*configuration_mode*.

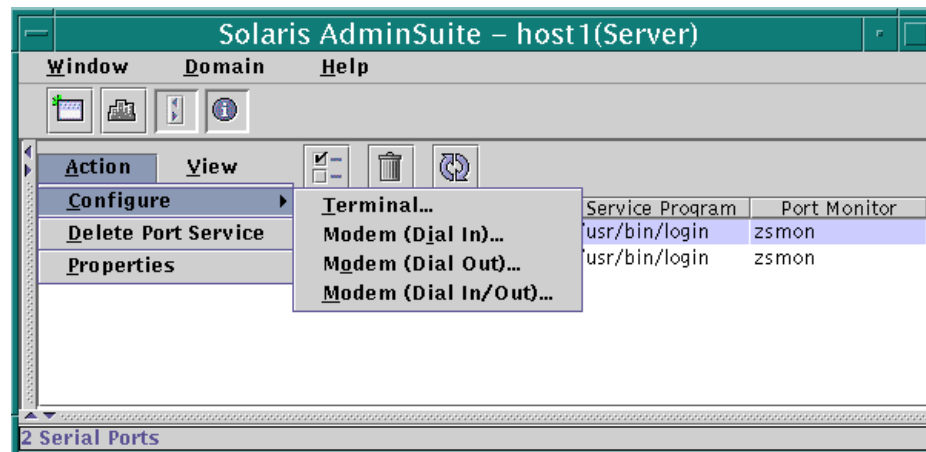


Figure 10-71 Solaris AdminSuite – Serial Ports Action Menu

This sequence displays a confirmation window that has your selection already displayed within the appropriate fields.

3. Confirm the options are correct.

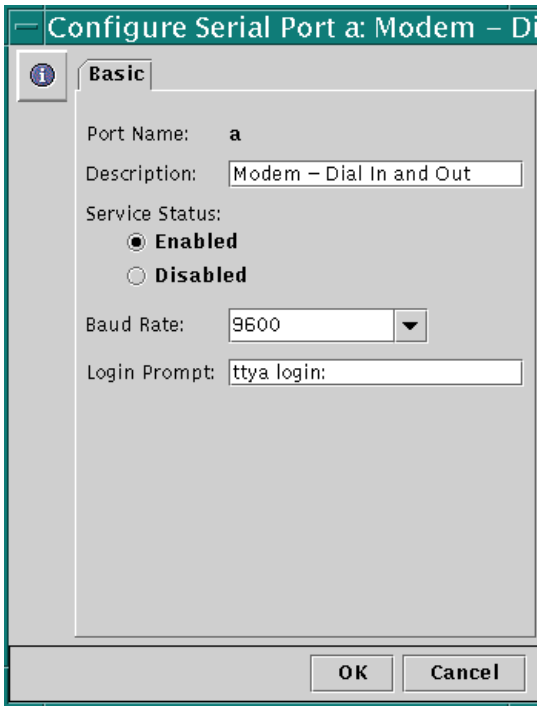


Figure 10-72 Configure Serial Port Window

4. Click OK.

You can use an alternative method for modifying serial port configuration.

To modify the serial port configuration, perform the following steps:

1. Click the serial port that you want to configure.
2. Click Action►Properties.
3. The basic properties page is identical to the Configure Serial Port pop-up. You can manually modify any field within the page.
4. Advanced properties, such as carrier detection, timeout, and port monitor are configured on the Serial Port advanced properties page.

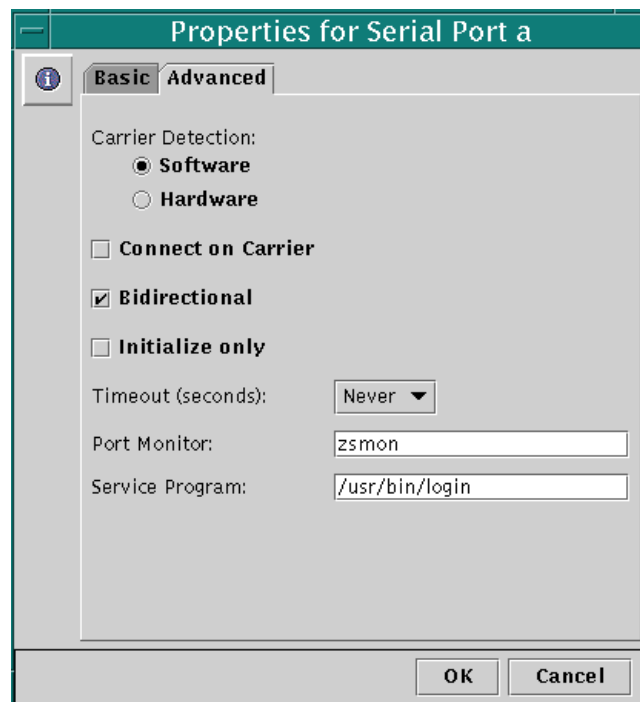


Figure 10-73 Properties for Serial Port Window

5. When you are satisfied with your modifications, click OK.
6. Use the Storage Manager to add a remote mount point and mount the NFS file system, `/usr/share/man`, from a network server.

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ List the benefits of the Solaris Management Console
- ☐ Install the Solaris Management Console software
- ☐ Add an application to the Solaris Management Console
- ☐ List the features of Solaris AdminSuite
- ☐ Install the Solaris AdminSuite software
- ☐ Create and modify user accounts using the Users feature of Solaris AdminSuite
- ☐ Add hosts to the server using the Computers/Networks feature of Solaris AdminSuite
- ☐ Manipulate mount states on existing file systems using the Mounts/Shares feature of Solaris AdminSuite
- ☐ Configure serial ports using the Serial Ports feature of Solaris AdminSuite

Objectives

Upon completion of this module, you should be able to:

- Describe the concept of a naming service
- List the available naming services
- Compare the functionality of naming services
- Describe the name service switch process and determine which configuration is appropriate for your network

Additional Resources



Additional resources – The following references provide additional details on the topics discussed in this module:

- *Solaris Naming Administration Guide*, Sun Part Number 806-1387-10

Name Services Overview

The name service concept centralizes the shared information in your network. A single machine, the name server, maintains the information previously maintained on each individual host. The name servers provide information, such as host names and IP addresses, user names, passwords, and automount maps.

Other hosts in your network, *clients*, request the information from the name server. This name server system responds to clients, and translates or resolving their requests from its memory-based (cached) or disk database(s).

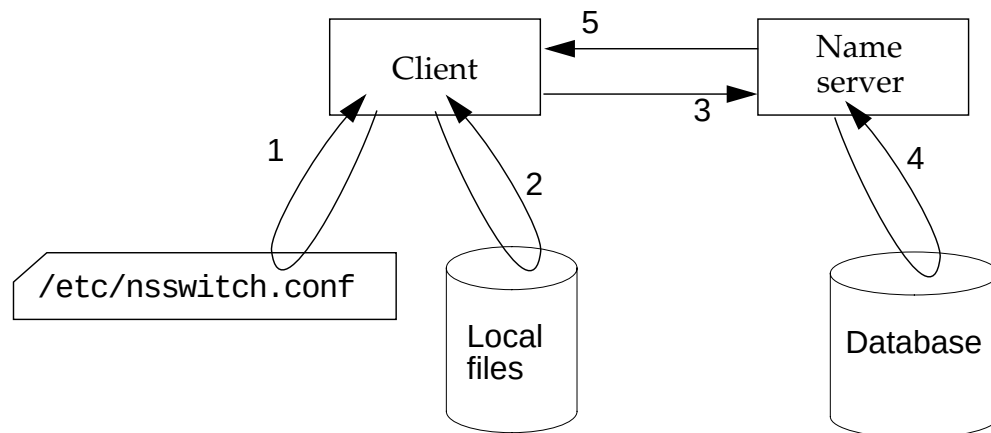


Figure 11-1 Overview of Name Service Functionality

The name service concept provides:

- Ease of management:
 - ▼ Single point of administration
 - ▼ Consistent information
 - ▼ Uniform view of network
- Immediate reflection of changes to all clients
- Assurance that clients do not miss updates

In a file-based scheme, updates received using ftp or copied in some way to client machines could be missed if a host were down or off the network when the changes were propagated.

- Secondary servers prevent a single point of failure

While a single master server is all that is required, the name service scheme allows for the creation of secondary servers (sometimes referred to as *slaves* or *replicas*). These secondary servers maintain a copy of the master's database, receive changes and updates to the database from the master, and participate in client resolution. As such, they not only overcome a single point of failure, but also play a role in increased network performance.

Available Name Services

Some common name service solutions address specific needs or architectures, as follows:

- Domain Name Service (DNS) – Used within a TCP/IP network to translate host names to their associated IP addresses.
- Network Information Service (NIS) – Provides a centralized lookup for LAN resources, such as user accounts, host names and addresses, services, automount maps, and other key files that would otherwise be needed on each host of the LAN.
- Network Information Service Plus (NIS+) – Provides a centralized lookup location for LAN resources. However, NIS+ is greatly expanded to support today's intranet with the features of a hierarchical naming structure, distributed administration, built-in security authentication, and cross-domain lookups.

- Lightweight Directory Access Protocol (LDAP) – Extends the naming services with a directory service. While a naming service allows you to look up an object given its name, a directory service also allows these objects to have attributes. Therefore, in addition to lookup, you can also get the attributes for these objects or search for objects given their attributes. LDAP is one such service.

DNS Overview

The DNS is application software that primarily provides for the distributed administration of IP addresses throughout the Internet. It also does the following:

- Enables local administrators to maintain information about their own local hosts and enables them to share this information with others throughout the Internet.
- Is commonly implemented by the Berkeley Internet Name Domain (BIND) software developed at the University of California at Berkeley. Sun uses a port of the BIND software.

You can use DNS to resolve host name and IP address requests on the intranet. If you connect your network to the Internet, you must use DNS because it is the name service used to resolve host name and IP address requests, and organize the millions of hosts and domains, on the Internet.

A *domain* is a collection of network hosts that share some common information. DNS domain names are dot-notated, and the names of hosts within the domain include the host name plus the domain name. For example, the host `merton` in the domain `sun.com` would be known to other systems, outside of the domain, as `merton.sun.com`.

All hosts known to DNS are included within the DNS namespace. The DNS namespace is divided into hierarchical domains. The namespace begins with the `root (.)` domain and includes all subdomains. Figure 11-2 on page 11-6 shows several top-level domains.

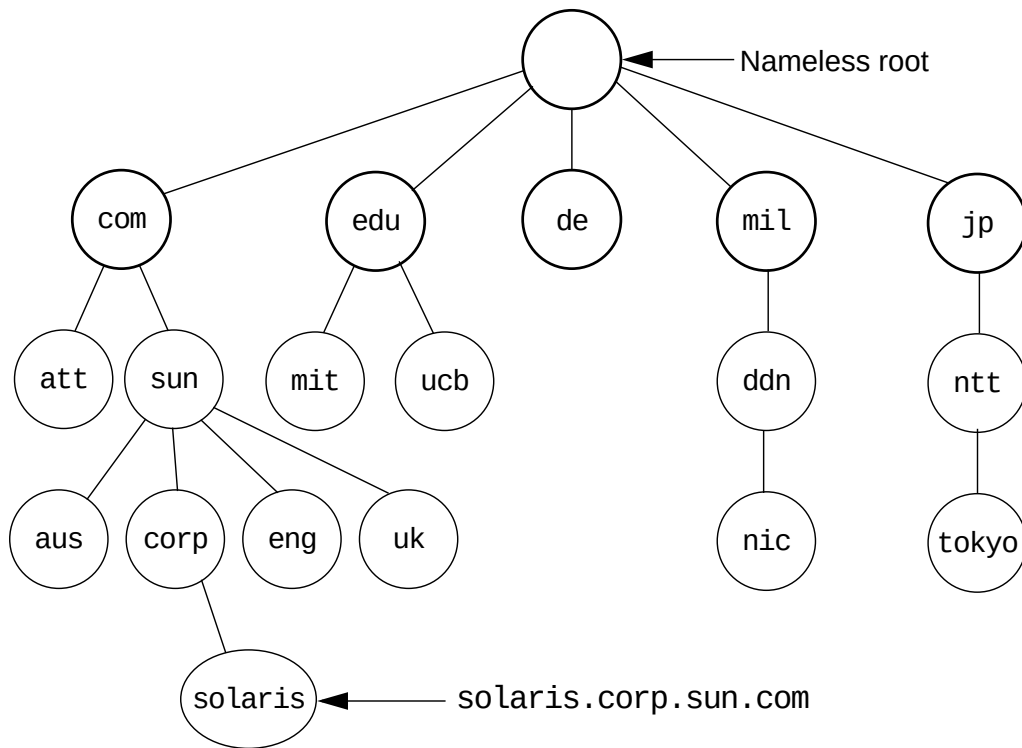


Figure 11-2 Top-Level Domains

The DNS nsswitch Template

The nsswitch template file for DNS is in `/etc/nsswitch.dns`, and the keyword is `dns`.

Top-Level Domains

The top-level domains are administered by Network Solutions, the NSI registry. Administration of the lower-level domains is delegated to the various organizations that are a part of the Internet.

The top-level domain you choose can depend on which one best suits the needs of your organization. Large organizations tend to use the organizational domains while small organizations or individuals often choose to use a country code.

Network Information Service Overview

NIS focuses on making network administration more manageable by providing centralized control over a variety of network information. NIS stores information about workstation names and addresses, users, the network itself, and network services. This collection of network information is referred to as the NIS *namespace*.

NIS Domains

NIS uses domains to arrange the machines, users, and networks in its namespace. However, it does not use a domain hierarchy, therefore, an NIS namespace is flat.

You cannot directly connect a NIS domain to the Internet using just NIS. However, organizations that want to use NIS and also be connected to the Internet can combine NIS with DNS. You can use NIS to manage all local information and use DNS for Internet host lookup. NIS provides a forwarding service that forwards host lookups to DNS if the information cannot be found in a NIS map. The Solaris Operating Environment also allows you to set up the `nsswitch.conf` file so that lookup requests from hosts go only to DNS, or to DNS and then NIS if the requests are not found by DNS, or to NIS and then DNS if the requests are not found by NIS.

Client-Server Arrangement

By running the NIS service, you can distribute administrative database maps among a variety of servers (master and slaves), and update those databases from a centralized location in an automatic and reliable fashion to ensure that all clients share the same name service information in a consistent manner throughout the network.

NIS Maps

NIS namespace information is stored in NIS *maps*. NIS maps were designed to replace UNIX `/etc` files, as well as other configuration files, so they store more than names and addresses. As a result, the NIS namespace has a large set of maps.

NIS maps are database files created from source files in the `/etc` directory (or a special directory you specify). The NIS domain maps typically include the following files:

- `auto_home`
- `auto_master`
- `bootparams`
- `ethers`
- `group`
- `hosts`
- `netgroup`
- `netmasks`
- `networks`
- `protocols`
- `passwd`
- `rpc`
- `services`
- `aliases`
- `timezone`
- `IP nodes`

Note – NIS administrators can also create custom maps for their specific network environment needs.

The NIS nsswitch Template

The `nsswitch` template for NIS is `/etc/nsswitch.nis` and the keyword is `nis`.

The NIS+ Environment

NIS+ enables you to store information about workstation addresses, security information, mail information, Ethernet interfaces, printers, and network services in locations where all workstations on a network can have access to it.

NIS+ Namespace

The NIS+ namespace is dynamic because updates can occur and be put into effect at any time by any authorized user.

The NIS+ namespace is hierarchical, and similar in structure to the DNS name service. The hierarchical structure allows you to configure a NIS+ namespace to conform to the logical hierarchy of an organization. The namespace's layout of information is unrelated to its physical arrangement. Thus, you can divide a NIS+ namespace into multiple domains that can be administered autonomously. Clients can have access to information in other domains, in addition to their own if they have the appropriate permissions.

An Example of the NIS+ Hierarchical Namespace

A software company named Solar, Inc. (`solar.com.`), with three divisions: Engineering (`eng`), Sales (`sales`) and Finance (`fin`). Set up the NIS+ namespace hierarchy illustrated in Figure 11-3. Each branch represents a domain. The Engineering branch has subdomains for development and software

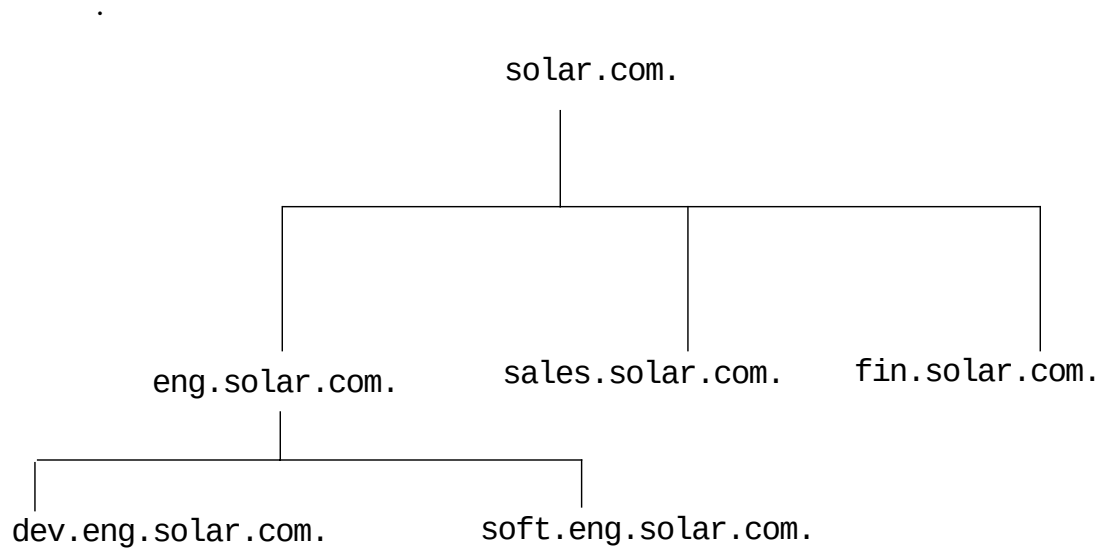


Figure 11-3 NIS+ Hierarchical Namespace

NIS+ Tables

The objects controlled in NIS are maps, and the objects controlled in NIS+ are tables.

Each NIS+ domain can contain the following table objects:

- auto_home
- auto_master
- bootparams
- client_info
- cred
- ethers
- group
- hosts
- mail_aliases
- netgroup
- netmasks
- networks
- passwd
- protocols
- rpc
- sendmailvars
- services
- timezone

The NIS+ nsswitch Template

The nsswitch template for NIS+ is `/etc/nsswitch.nisplus` and the keyword is `nisplus`.

Lightweight Directory Access Protocol (LDAP) Overview

The implementation of NIS within the Solaris Operating Environment provided a mechanism for advertising (identifying and locating) network objects and resources. Two major drawbacks are its flat structure (a single domain) and the proprietary nature of the naming services.

NIS+ is an improvement to the NIS structure. NIS+ is built in a hierarchical structure so that it more closely resembles the internal structure of an organization and, therefore, it can access multiple domains (provided the authorization and authentication features are properly enabled). However, like NIS, NIS+ is somewhat proprietary (Every organization cannot access the information).

These proprietary naming services are often decipherable only from within an organization or group of organizations (and sometimes from within a particular application), and they create islands of information that must translate requests for information from the World Wide Web.

With the implementation of standardized directory services, an organization can configure one or more servers to direct requests for information through the organization to the appropriate servers.

A standardized directory service implies that all participating organizations adhere to a common rule set for hierarchical naming structures, authorization and authentication practices, and configuration of various other attributes.

The X.500 Directory Access Protocol (DAP) is one such standard; however, when organizations attempt to implement it, many find it difficult to administer, and this becomes a barrier to the success of the directory server concept as a whole. Additionally, implementation of the X.500 structure often requires the power and resources of a UNIX system, thus putting up a barrier to the world of personal computer users. It is readily apparent that an easier to administer, lighter weight protocol is needed, so more than 40 companies have joined with Netscape and the University of Michigan to support a Lightweight Directory Access Protocol (LDAP) as a proposed standard for Internet directories.

Common Uses of LDAP

LDAP is useful when you need a resource locator; however, it is practical only in read-intensive environments where you do not need frequent updates. You can use LDAP as a repository for the same information stored in NIS and NIS+. The following lists some common uses:

- A resource locator for an online phone directory. Authorized users can update it as necessary to maintain its accuracy. This eliminates the need for a printed phone directory (which is outdated as soon as a change is made following the printing).

Note – LDAP is useful for phone directories that are updated relatively infrequently, but would be ineffective for sales transaction databases that center around constantly updating data.

- The address book in most e-mail clients
- A repository of information for Web-based applications that support sales and inventory-control processes. However, you must keep in mind that heavy-transaction oriented sites are better suited to other relational databases, which are suited to those applications.
- For automatically locating network resources. It provides a mechanism to locate printers, file servers, and network services.
- To centralize network management. Instead of maintaining duplicate information across many servers, the LDAP server is configured so that a single directory can be accessed by all applications.
- To tighten security. Authorization and authentication attributes can be configured to control access to applications, resources, and modifications.

These are just a few of the possibilities. The list is limited only by the creativity of the administrators that implement LDAP.

LDAP applications are grouped three ways: LDAP used to locate network users and resources, those used to manage these resources, and those that provide authentication and authorization security features.

The LDAP nsswitch Template

The nsswitch template for LDAP is `/etc/nsswitch.ldap` and the keyword is `ldap`.

The Name Service Switch

All Solaris Operating Environment workstations use `/etc/nsswitch.conf` as the name service switch. The `/etc/nsswitch.conf` file is used by the operating system for any network information lookups. It is commonly referred to as the *name service switch file*. The `/etc/nsswitch.conf` file determines which sources of information your system uses and the order in which those sources are used.

The `nsswitch.conf` Configuration Files

The Solaris 8 Operating Environment includes the following five templates for the name service switch configuration file:

- `/etc/nsswitch.files` – The template name service switch file that, when copied to the `/etc/nsswitch.conf` file, permits only searches of the local `/etc` files.
- `/etc/nsswitch.dns` – The template name service switch file that, when copied to the `/etc/nsswitch.conf` file, searches only the local `/etc` files for all entries with the exception of the `hosts` entry. The `hosts` entry can be directed to use DNS for lookup.
- `/etc/nsswitch.nis` – The template file that uses the NIS database as the primary source of all information except the `passwd`, `group`, `automount`, and `aliases` maps, which are directed to use the local `/etc` files first and then the NIS databases. With the name service search order for the `passwd` and `group` files established as the local files first followed by the NIS database, there is no need for a plus (+) in the `passwd` file.
- `/etc/nsswitch.nisplus` – The template file that uses NIS+ as the primary source for all information except the `passwd`, `group`, `automount`, and `aliases` tables, which use the local `/etc` files first and then the NIS+ databases.
- `/etc/nsswitch.ldap` – The template file that uses LDAP as the primary source for all information except the `passwd`, `group`, `automount`, and `aliases` tables, which use the local `/etc` files first and then the LDAP databases.

After determining which naming service to use, you select the appropriate template file; /etc/nsswitch.files, /etc/nsswitch.dns, /etc/nsswitch.nis, /etc/nsswitch.nisplus, or /etc/nsswitch.ldap and copies it to /etc/nsswitch.conf.

The /etc/nsswitch.nis Template

The following example is the default /etc/nsswitch.nis file (which references name service resolution provided by NIS) that is provided with the installation of the Solaris 8 Operating Environment software:

```
#
# /etc/nsswitch.nis:
#
# An example file that could be copied over to /etc/nsswitch.conf; it
# uses NIS (YP) in conjunction with files.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file has a "-" for nametoaddr_libs of "inet" transports.
jade!7

# the following two lines obviate the "+" entry in /etc/passwd and
# /etc/group.
passwd:                files nis
group:                 files nis

# consult /etc "files" only if nis is down.
hosts:                 nis [NOTFOUND=return] files
ip nodes:              files
#Uncomment the following line and comment out the above to resolve
#both IPv4 and IPv6 addresses from the ipnodes databases. Note that
#IPv4 addresses are searched in all of the ipnodes databases before
#searching the hosts databases. Before turning this option on, consult
#the Network Administration Guide for more details on using IPv6.
networks:              nis [NOTFOUND=return] files
protocols:             nis [NOTFOUND=return] files
rpc:                   nis [NOTFOUND=return] files
ethers:                nis [NOTFOUND=return] files
netmasks:              nis [NOTFOUND=return] files
bootparams:            nis [NOTFOUND=return] files
publickey:             nis [NOTFOUND=return] files

netgroup:              nis

automount:             files nis
```

```
aliases:                files nis

# for efficient getservbyname() avoid nis
services:               files nis
sendmailvars:           files
printers:               user files nis

auth_attr:              files nis
prof_attr:              files nis
```

This example `/etc/nsswitch.nis` file identifies NIS as the *only* source of data for `hosts`, `netgroup`, and `netmasks` entries. Alternatively, the lines for `passwd`, `group`, and `automount` have `files` followed by `nis`. This indicates that the system uses the local files first and the NIS database second for lookups.

If you are using NIS as the naming service, copy the `/etc/nsswitch.nis` file to the `/etc/nsswitch.conf` file.

Modification of the `/etc/nsswitch.conf` File

After copying the appropriate template file to `/etc/nsswitch.conf`, you might have to modify the file to enable the operating system to access network information. For example, to enable `hosts` to be resolved using local files first, DNS second, and NIS third, the line for `hosts` would appear as follows:

```
hosts:    files  dns    nis
```

Conversely, to restrict login access to only those users with local accounts, you can remove `nis` from the line for `passwd` as in this example:

```
passwd:   files
```

The name service switch file contains a list of over 19 databases, their name service sources for resolution, and the order in which these sources are searched. As shown in Table 11-1, one or more sources from this list can be specified for each database.

Table 11-1 Database Sources

Source	Description
<code>files</code>	Refers to the client's local <code>/etc</code> files
<code>nisplus</code>	Refers to an NIS+ table
<code>nis</code>	Refers to an NIS map
<code>user</code>	Applies to the <code>printers</code> entry.
<code>dns</code>	Applies <i>only</i> to the <code>hosts</code> entry
<code>ldap</code>	Refers to a Directory Information Tree (DIT)
<code>compat</code>	Supports an old-style "+" syntax for <code>passwd</code> and <code>group</code> information

Name Service Switch Status and Action Values

Suppose the default `/etc/nsswitch.nis` file shown on page 11-16 was copied to the `/etc/nsswitch.conf` file. The name service switch now presents some action values for several of the entries. The naming service search for resolution from the source specified returns a status code that presents an appropriate value in response to the user requesting NIS information. Table 11-2 describes these status codes.

Table 11-2 Name Service Search Return Status Codes

Status Code	Description
SUCCESS	Requested entry was found
UNAVAIL	Source was unavailable
NOTFOUND	Source contains no such entry
TRYAGAIN	Source returned "I am busy, try later" message

Actions

For each status code, two actions are possible. Table 11-3 describes these actions.

Table 11-3 Status Code Actions

Action	Description
continue	Try the next source
return	Stop looking for the entry

The default actions are

- SUCCESS = return
- UNAVAIL = continue
- NOTFOUND = continue
- TRYAGAIN = continue

The following entry assumes that the NIS name service is running, the syntax for this entry means that only the NIS hosts table is searched. If a NIS server has no map entry for a host lookup, the system would *not* reference the local files. Remove the [NOTFOUND=return] entry if you want to search the NIS hosts table and the local hosts file.

```
hosts: nis [NOTFOUND=return] files
```

Table 11-4 on page 11-20 shows the naming service features.

Table 11-4 Naming Service Features

	Files	NIS	NIS+	DNS	LDAP
Intended Scope (Best fit)	Small local network LAN	LAN	Multiple LANs	LAN WAN	LAN WAN
Name space	Flat	Flat	Hierarchical	Hierarchical	Hierarchical
Template for nsswitch.conf	nsswitch.files	nsswitch.nis	nsswitch.nisplus	nsswitch.dns	nsswitch.ldap
Object that stores information	File	Map	Table	Zone files	Directory information tree (DIT)

Exercise: Reviewing Naming Services



Exercise objective – In this lab, you evaluate your understanding of the naming services concepts presented in this module.

Preparation

If necessary, refer to your lecture notes to answer these exercise questions.

Tasks

Answer the following questions:

1. What are the four name service solutions mentioned in this module?

2. What are two main services provided by DNS?

3. What types of information does NIS+ enable you to store?

4. What are the map files used in NIS?

5. What file is referred to as the name service switch file and why?

6. If you decide to use the NIS for name service resolution, what template file would you use to create the name service switch file?

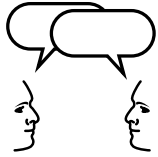
7. What does it mean if the following entry exists in the name service switch file?

hosts: nis [NOTFOUND=return] files

8. Is the following an appropriate entry to the `/etc/nsswitch.conf` file? Why or why not?

groups: dns files nis

Exercise Summary



Discussion – Take a few minutes to discuss the experiences, issues, or discoveries that you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Task Solutions

1. What are the four name service solutions mentioned in this module?

DNS, NIS, NIS+, and FNS.

2. What are two main services provided by DNS?

DNS provides host name to IP address translation and the concept of each system belonging to a domain.

3. What types of information does NIS+ enable you to store?

NIS+ enables you to store information about workstation addresses, security information, mail information, Ethernet interfaces, printers, and network services in locations where all workstations on a network can have access to it.

4. What are the map files used in NIS?

• auto_home	• networks
• auto_master	• protocols
• bootparams	• passwd
• ethers	• rpc
• group	• services
• hosts	• aliases
• netgroup	• timezone
• netmasks	• client_info
• mailaliases	• sendmailvars

5. What file is referred to as the name service switch file and why?

The /etc/nsswitch.conf file is referred to as the name service switch file because it is used by the operating system for any network information lookups. This file indicates whether DNS, NIS, NIS+, or local files are to be used for name service resolution, and if more than one naming service is to be used, this file indicates the order in which these services should be accessed.

6. If you decide to use the NIS for name service resolution, what template file would you use to create the name service switch file?

`/etc/nsswitch.nis`

7. What does it mean if the following entry exists in the name service switch file?

```
hosts: nis [NOTFOUND=return] files
```

Assuming that the NIS name service is running, the syntax for this entry means that only the NIS hosts table is searched. If a NIS server has no map entry for a host lookup, the system would not reference the local files. Remove the [NOTFOUND=return] entry if you want to search the NIS hosts table and the local hosts file.

8. Is the following an appropriate entry to the `/etc/nsswitch.conf` file? Why or why not?

```
groups: dns files nis
```

This entry does not make sense because `dns` only applies to the `hosts` entry in the name service switch file.

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ Describe the concept of a naming service
- ☐ List the available naming services
- ☐ Compare the functionality of naming services
- ☐ Describe the name service switch process and determine which configuration is appropriate for your network

Objectives

Upon completion of this module, you should be able to:

- Describe the NIS components, master server, slave server, and client, and the NIS processes
- Configure an NIS master, slave, and client
- List the steps to add a new NIS map
- Use commands to update and propagate an NIS map

Additional Resources



Additional resources – The following references provide additional details on the topics discussed in this module:

- *Solaris Naming Administration Guide*,
Sun Part Number 806-1387-10
- *Solaris Naming Setup and Configuration Guide*,
Sun Part Number 806-1386-10

Introduction to NIS Concepts

NIS enables the creation of server systems that act as central repositories for several of the administrative files found on UNIX systems. The benefits of NIS include:

- Centralized administration of files
- Better scaling of file administration as networks grow

As Figure 12-1 illustrates, NIS is organized into named administrative domains. Within each domain exists one NIS master server, zero or more slave servers, and one or more clients.

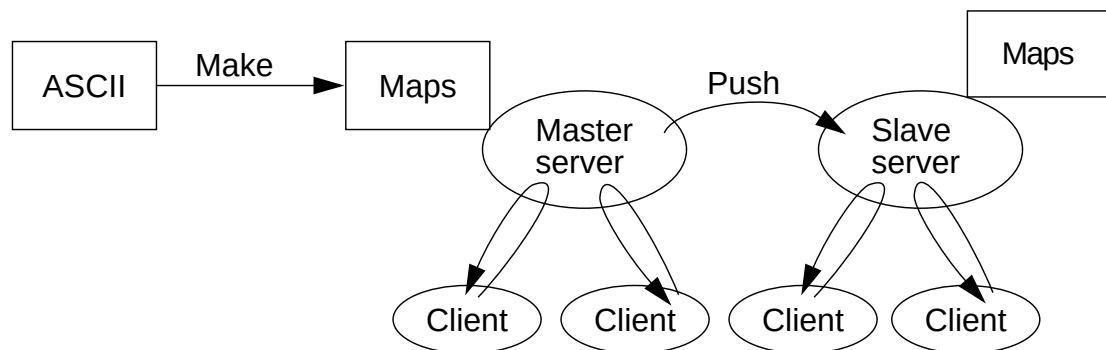


Figure 12-1 NIS Domains

NIS Master Server

Within each domain, the NIS master server:

- Contains the original /etc ASCII files used to build the NIS maps
- Contains the NIS maps generated from the ASCII files
- Provides a single point of control for the entire NIS domain
- Is easy to set up

NIS Slave Servers

Within each domain, the NIS slave servers:

- Do not contain the original /etc ASCII files (which are used to build the NIS maps)
- Contain copies of the NIS maps copied from the NIS master server
- Provide a backup repository for NIS map information
- Provide redundancy in case of server failures
- Provide load sharing on large networks

NIS Clients

Within each domain, the NIS clients:

- Do not contain the original /etc ASCII files (which are used to build the NIS maps)
- Do not contain any NIS maps
- Bind to the master server or a slave server to obtain access to the administrative file information contained in that server's NIS maps
- Dynamically rebind to another server in case of server failure
- Make all appropriate system calls aware of NIS

Note – All hosts in the NIS environment are clients, including the NIS master and slaves.

NIS Processes

The two main processes involved in the running of an NIS domain are:

- `ypserv` – Runs on master and slave servers
- `ypbind` – Runs on master and slave servers, as well as client systems

There are three daemons that used in an NIS domain on the master server:

- `rpc.yppasswdd`
- `ypxfrd`
- `rpc.ypupdated`

Figure 12-2 illustrates a domain with these NIS processes and daemons.

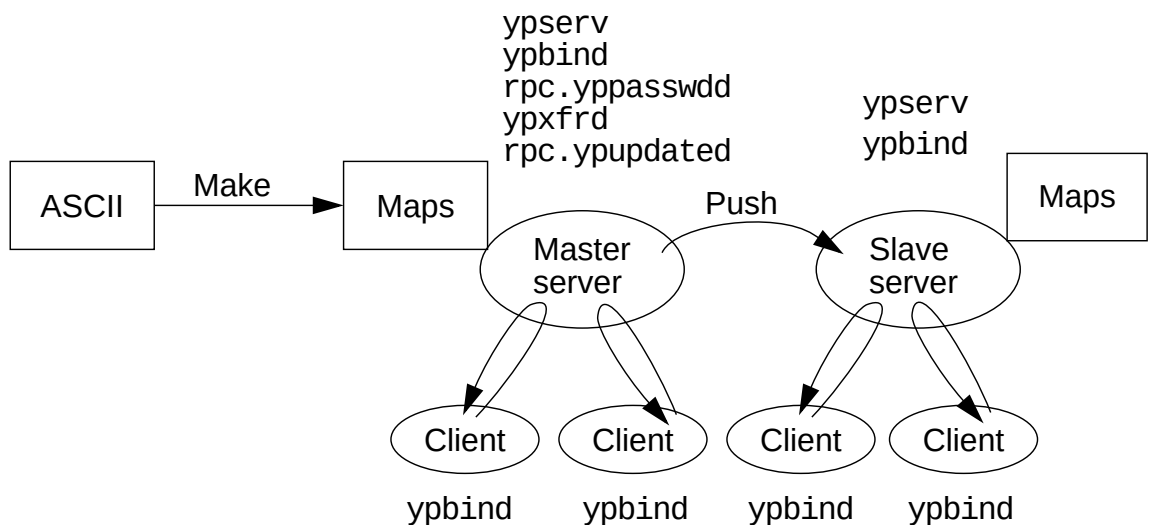


Figure 12-2 NIS Processes and Daemons

The ypserv Daemon

The ypserv daemon is a utility that:

- Runs on master and slave servers
- Answers ypbind requests from clients
- Responds to client information requests

The ypbind Daemon

The ypbind daemon is a process that:

- Runs on all NIS systems, servers as well as clients
- Makes initial client-to-server binding requests
- Stores binding information in the `/var/yp/binding/domainname` directory
- Rebinds to another server if the connection is lost with the initial server
- Requests NIS map information at the library-call level

The rpc.yppasswdd Daemon

The rpc.yppasswdd daemon is a process that:

- Allows users to change their passwords
- Updates the `/etc/passwd` and `/etc/shadow` files on the master server
- Updates the NIS password map
- Provides or “pushes” the NIS password map to all slave servers

The ypxfrd Daemon

The ypxfrd daemon is a process that:

- Runs on the NIS master server only
- Responds to slave requests (using ypxfr) to pull the maps from the master
- Transfers NIS maps at high speed

The rpc.yupdated Daemon

The rpc.yupdated daemon is a process that:

- Runs on the NIS master server only
- Updates the `publickey` map if secure RPC is enabled

The Structure of NIS Maps

NIS maps are located in the `/var/yp/domainname` directory (where *domainname* is the name of the NIS domain). There are two files (`.pag` and `.dir` file) for each map in this directory.

NIS Maps Filenames

The syntax for the NIS maps is:

map.key.pag or *map.key.dir*

where:

- *map* – The base name of the map (hosts, passwd, and so on)
- *key* – The map's sort key (byname, byaddr, and so on)
- *pag* – The map's data
- *dir* – An index to the `.pag` file if the `.pag` file is large

The `.dir` file can be empty if the `.pag` file is small.

Map Contents and Sort Keys

The *contents* of each map is a key and value pair. The *key* represents the data used to perform the lookup in the map while the *value* represents the data returned upon a successful lookup. Maps can be duplicated in the `/var/yp/domainname` directory; they represent the results of the sorting of the map's data based on different keys.

For example, the map `/var/yp/domainname/hosts.byaddr.pag` contains the data for the hosts map indexed by host IP addresses. Similarly, the `/var/yp/domainname/hosts.byname.pag` map contains the same host data using the host name as the lookup key. For the domain name `training`, the following would be a list of the NIS map files for the hosts map:

- `/var/yp/training/hosts.byname.pag`
- `/var/yp/training/hosts.byname.dir`
- `/var/yp/training/hosts.byaddr.pag`
- `/var/yp/training/hosts.byaddr.dir`

Commands to Read Maps

You can use two commands to read maps:

- `ypcat [ -k ] map` – This command is similar to the `cat file` command
- `ypmatch [ -k ] value map` – This command is similar to the `grep value file` command

Generating NIS Maps

To generate NIS maps, you need the source files, which are located in either the `/etc` directory on the master server or copied to an alternative directory. You should not keep the source files in `/etc` because the contents of the maps are then the same as the contents of the local files on the master server. This is a special problem for `passwd` and `shadow` files, because all users would have access to the master server maps and the root password would be passed to all YP clients through the `passwd` map.

If you choose to locate the source files in another directory, you must modify the `/var/yp/Makefile` by changing the `DIR=/etc` line and the `PWDIR=/etc` line to `DIR=your-choice` and `PWDIR=your-choice`, where *your-choice* is the name of the directory you are using to store the source files. This enables you to treat the local files on the server as if they were those of a client. (You should first save a copy of the original `Makefile`.)

The following is an excerpt from the default Makefile showing the variable DIR and PWDIR set to their default values:

```
#
# Copyright (c) 1998, by Sun Microsystems, Inc.
# All rights reserved.
#
#ident "@(#)Makefile 1.23 98/05/01 SMI"
#
#----
# It is somewhat confusing to note that Solaris 2.x uses /etc/auto_master
# instead of the 4.x /etc/auto.master file name because of NIS+ treating
# a
# "." in a special way.
#
# Set the following variable to "-b" to have NIS servers use the domain
# name
# resolver for hosts not in the current domain.
#B=-b
B=
DIR =/etc
#
# If the passwd, shadow and/or adjunct files used by rpc.yppasswdd
# live in directory other than /etc then you'll need to change the
# following line.
# DO NOT indent the line, however, since /etc/init.d/yp attempts
# to find it with grep "^PWDIR" ...
#
PWDIR =/etc
DOM = `domainname`
NOPUSH = ""
ALIASES = /etc/mail/aliases
YPDIR=/usr/lib/netsvc/yp
```

The ypinit Command and the NIS Makefile

The NIS maps are generated by the NIS configuration binary, `/usr/sbin/ypinit`, and the `make` command. The `ypinit` command reads the `/var/yp/Makefile` for source file locations and converts ASCII source files into NIS maps.

Password File

For security reasons, and to prevent unauthorized root access, the files used to build the NIS password maps should not contain an entry for root.

To do this, copy the files to an alternative directory and modifying the `PWDIR` entry in the `Makefile`.

Figure 12-3 on page 12-12 shows the important files on the NIS mast.

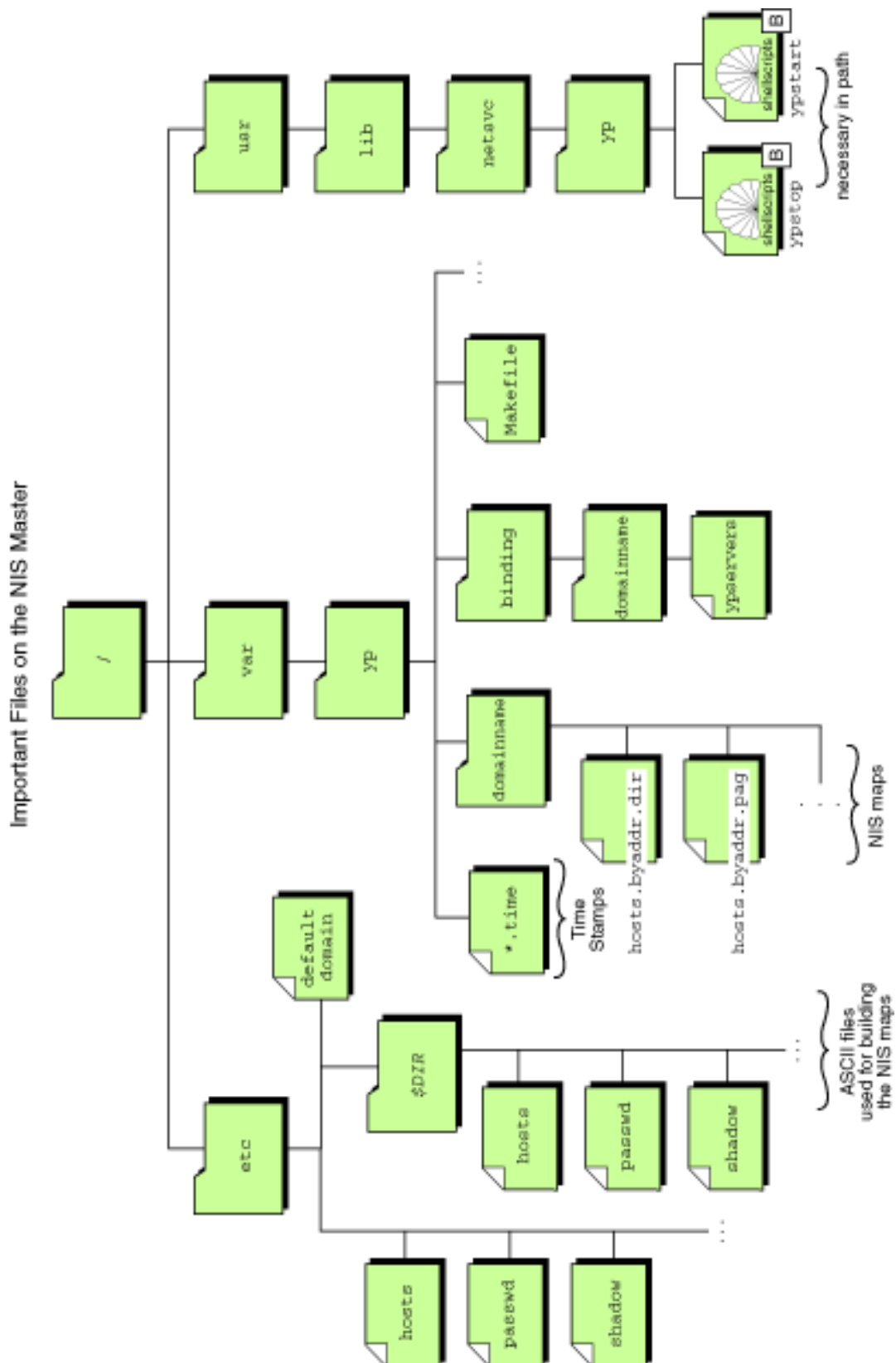


Figure 12-3 Important Files on the NIS Master

Configuring the NIS Master Server

To set up the NIS name service master server, perform the following steps:

1. Determine which machines within your network domain will be NIS servers; there will be one NIS master and as many NIS slaves as needed. Typically, *all* systems within the domain will be NIS clients.

Note – The NIS kit that was supplied with releases before the Solaris 2.6 Operating Environment is no longer provided. NIS is now part of the release rather than a separate file.

2. Copy the `/etc/nsswitch.nis` file to `/etc/nsswitch.conf` and modify it, if necessary.
3. Choose an NIS domain name. This is usually less than 32 characters in length. (The maximum length is 256 characters.)
4. Execute the `domainname` command to set the local NIS domain.
5. Create an `/etc/defaultdomain` file with the domain name.
6. Make sure to maintain the format established by the original files, and update the text files in the `/etc` directory (all of the files that are used for NIS maps) on the master server with information about the domain.

Note – You can also copy the network information files to some other location on the system and modify them there rather than modifying them in the `/etc` directory.

7. Use the `touch` command to create zero-length files with the following names: `/etc/ethers`, `/etc/bootparams`, `/etc/locale`, `/etc/timezone`, `/etc/netgroup`, and `/etc/netmasks`. These files are necessary for the creation of the complete list of NIS maps as directed in the `Makefile`. When you initialize NIS, you will receive error messages for each of these files if they do not exist.



8. Install an updated Makefile in /var/yp if you intend to use NIS on the system that functions as your JumpStart™ server. This provides entries that create a map for the /etc/locale file.

To create a Makefile that supports unassisted JumpStart installation capability, make the following changes:

- a. Add the following text after the existing *.time entries; all beginning white space must be tabs:

```
locale.time: $(DIR)/locale
    -@if [ -f $(DIR)/locale ]; then \
        sed -e "/^#/d" -e s/#.*$$// $(DIR)/locale \
        | awk '{for (i = 2; i<=NF; i++) print $$i, $$0}' \
        | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/locale.byname; \
        touch locale.time; \
        echo "updated locale"; \
        if [ ! $(NOPUSH) ]; then \
            $(YPPUSH) locale.byname; \
            echo "pushed locale"; \
        else \
            : ; \
        fi \
    else \
        echo "couldn't find $(DIR)/locale"; \
    fi
```

- b. Append the word locale to the line beginning with the word all.
- c. Add the following line after the auto.home: auto.home.time entry:

 locale: locale.time

9. Create or populate the file /etc/locale and make an entry for each domain on your network using the following format:

domainname locale

For example:

classroom.Central.Sun.COM en_US

10. Edit the Makefile, and change every reference to the *.attr to add the security subdirectory to the pathname, as follows:

```
$(DIR)/auth_attr
$(DIR)/exec_attr
```

```
$(DIR)/prof_attr  
$(DIR)/audit_user
```

becomes:

```
$(DIR)/security/auth_attr  
$(DIR)/security/exec_attr  
$(DIR)/security/prof_attr  
$(DIR)/security/audit_user
```

11. Initialize the master server using the local `/etc` files by executing the `ypinit -m` command.

ypinit -m

- a. The program prompts you for a list of slave servers. When you complete your list, press **Control-d**. You can make entries for all slaves now or rerun the command after you determine that you need more or fewer slave servers.
- b. The program asks if you want to terminate on the first fatal error. If you answer **n**, the procedure completes the creation of the NIS database files. If you answer **y**, the process aborts with the first error. You can fix it and restart the `ypinit` program.

The following dialog provides the text feedback displayed as the program begins:

```
# ypinit -m
```

```
In order for NIS to operate successfully, we have to construct a list of
the NIS servers. Please continue to add the names for YP servers in order
of preference, one per line. When you are done with the list, type a
<control D> or a return on a line by itself.
```

```
    next host to add: server1
```

```
    next host to add: ^D
```

```
The current list of yp servers looks like this:
```

```
server1
```

```
Is this correct? [y/n: y] y
```

```
Installing the YP database will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.
```

```
Do you want this procedure to quit on non-fatal errors? [y/n: n] n
```

```
OK, please remember to go back and redo manually whatever fails. If you
don't, some part of the system (perhaps the yp itself) won't work.
```

Note – If you have to restart the `ypinit` program, you are prompted to destroy the `/var/yp/domainname` directory.
Answer y.

12. Start the NIS daemons on the master server with the following command:

```
# /usr/lib/netsvc/yp/ypstart
```

13. Once `ypbind` is running you need to complete the following steps to build the `mail.aliases` map.

```
# cd /var/yp
```

```
# /usr/ccs/bin/make
```

If you want to stop the NIS service running on the NIS master, issue the following command:

```
# /usr/lib/netsvc/yp/ypstop
```



Caution – Installations that select Core, End User, or Developer software configuration clusters do not have all of the necessary files in the `/usr/lib/netsvc/yp` to allow a host to function as an NIS server.

Accessing and Testing the NIS Service

The initial way that users access NIS information is during login. When the user types in a user name and password, the NIS database verifies this information before it enables a login shell. If the user's home directory is on a remote system in the NIS domain, the NIS `auto_home` map will reference the server information and automatically mount the appropriate directory.

There are some informative commands that display information in the NIS database. You can use these commands to test NIS service.

The most commonly used NIS commands are:

Note – You do not have to be the superuser to use these commands.

- `ypcat` – Prints values from the NIS database.

Example: Print the information from the `hosts` database

```
$ ypcat hosts
129.0.0.1      localhost
192.9.200.1    host1 loghost
192.9.200.2    host2
192.9.200.6    host6
192.9.200.8    host8
192.9.200.101  server1
192.9.200.102  server2
```

- `ypmatch` – Prints the value of one or more keys from the NIS database.

Example: Match individual host entries

```
$ ypmatch host1 server1 hosts
192.9.200.1    host1
192.9.200.101  server1
```

Example: Match a specific user in the password database

```
$ ypmatch user1 passwd
user1:Q7icI6NRPEmak:11001:10:User1:export/home/user1:/bin/ksh
```

- **ypwhich** – Returns the name of the NIS server that supplies the NIS map services to an NIS client.

Example: Return the name of the NIS master server

```
$ ypwhich  
server1
```

When used with the **-m** option, the **ypwhich** command provides a list of all databases and the name of the master server.

Example: List all databases on masterserver

```
$ ypwhich -m  
auto.home      server1  
auto.master    server1  
timezone.byname server1  
netmasks.byaddr server1  
publickey.byname server1  
<remaining output omitted>
```

Configuring the NIS Client

Typically, you configure all systems within a NIS domain as clients:

1. Copy the `/etc/nsswitch.nis` file to `/etc/nsswitch.conf` and modify it if necessary.
2. Edit the `/etc/hosts` file to ensure that the NIS master server and all slave servers have been defined.
3. Execute the '`domainname domainname`' command to set the local NIS domain. For example,

```
# domainname classroom.Central.Sun.COM
```

Note – You can use this command to set the name of a domain within a classroom in the central region training center.

4. Create or populate the `/etc/defaultdomain` file with the domain name.
5. Initialize the system as an NIS client with the following command:

```
# ypinit -c
```

6. When prompted for a list of NIS servers, enter the names of the NIS master and all slave servers.
7. Start the NIS software with the following command:

```
# /usr/lib/netsvc/yp/ypstart
```

8. On the newly configured NIS client, test the NIS functionality by entering the following command:

```
# ypwhich -m
```

The output should include the name of the NIS master server along with the database maps it is serving.

Configuring the NIS Slave Server

You must have at least one NIS slave server provide backup should the NIS master server become unavailable. You can do this by using the following steps on the system that is designated to become the slave server:

1. Copy the `/etc/nsswitch.nis` file to `/etc/nsswitch.conf` and modify it if necessary.
2. Edit the `/etc/hosts` file to ensure that the NIS master and all NIS slave servers have been defined.
3. Execute the `domainname` command to set the local NIS domain.

```
# domainname domainname
```

For example,

```
# domainname classroom.Central.Sun.COM
```

4. Create or populate the `/etc/defaultdomain` file with the domain name. Add a one-line entry to represent the selected domain name (for example, *domainname* in step 3).
5. Initialize the system as an NIS client with the following command:

```
# ypinit -c
```

6. When prompted for a list of NIS servers, enter the NIS master host followed by the name of the local host and all other NIS slave servers on the local network.
7. On the NIS master, ensure that the `ypserv` process is running by running this command:

```
# ps -ef | grep ypserv
```

If it is not running, refer to the previous section on how to start NIS daemons on the master.

8. Return to the proposed NIS slave system and run `ypstart`.

```
# /usr/lib/netsvc/yp/ypstart
```


9. Initialize the system as an NIS slave with the following command:

```
# ypinit -s master
```

where *master* is the name of the NIS master.

Note – If you did not add the name of the NIS slave server when you initially configured the NIS master server using the `ypinit` command, run the `ypinit -m` command once more on the NIS master server. In the process of updating the NIS master, the script prompts you for confirmation when it is about to destroy the existing domain database. Confirm by entering **y**.

10. Stop the NIS daemons on the slave server with the following command:

```
# /usr/lib/netsvc/yp/ypstop
```

11. Restart the NIS daemons on the slave server with the following command:

```
# /usr/lib/netsvc/yp/ypstart
```

12. On the newly configured NIS slave server, test the NIS functionality with the following command:

```
# ypwhich -m
```

The output should include the name of the NIS master server along with a list of database maps it is serving to the NIS domain.

Updating the NIS Map

Database files change as time goes on and your NIS maps must be updated. To update the NIS maps (on the master server), perform the following steps:

1. Update the text files in your source directory (typically `/etc` unless it was changed in the `Makefile`) with the new or modified information.
2. Change to the `/var/yp` directory.

```
# cd /var/yp
```

3. Refresh the NIS database maps by executing the `make` command.

```
# /usr/ccs/bin/make
```

Updating the Hosts Map and Propagating to Slave Servers

The following steps manually update the NIS hosts map on the master server and propagate all maps to the slave servers:

1. Edit a map source file on the NIS master.

```
# vi /etc/hosts
```

2. Remake and push the NIS maps to the slave servers.

```
# cd /var/yp; make
```

The following commands manually “pull” only the host maps from the master server.

```
# /usr/lib/netsvc/yp/ypxfr hosts.byaddr
```

```
# /usr/lib/netsvc/yp/ypxfr hosts.byname
```

You can also pull all of the maps from the master server at once using the following command:

```
# ypinitt -s nis_master
```

Updating the NIS Password Map

If the NIS master is running the `rpc.yppasswdd` daemon, you can update any client system to the NIS password map by using the `yppasswd` or `passwd` commands (Figure 12-4).

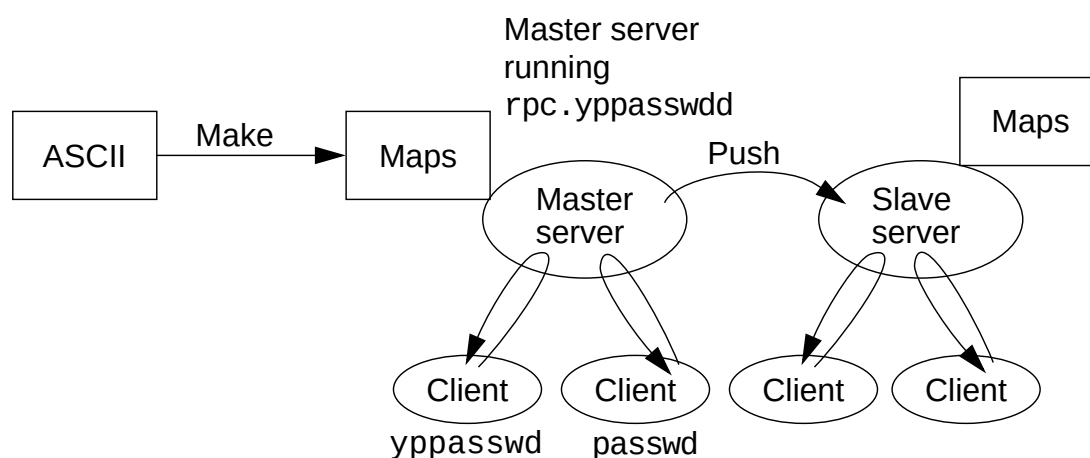


Figure 12-4 Updating the NIS Password Map

The following describes what you need to do to be successful at updating of the password map:

- Running the `rpc.yppasswdd` daemon on the NIS master server

```
# /usr/lib/netsvc/yp/rpc.yppasswdd /etc/passwd -m passwd
```

The `rpc.yppasswdd` daemon updates the NIS master's `/etc/passwd` file and `passwd` map whenever users change their NIS password (with the `passwd` or `yppasswd` commands). The `passwd` map is then pushed to all slave servers.

- Run the `passwd` command on any NIS client

% **passwd**

Changing NIS password for user1 on server1.

Old password:

New password:

Retype new password:

NIS entry changed on server1

Updating the NIS Slave Server Map

Sometimes maps fail to propagate and you must use `ypxfr` manually to retrieve new map information. To automate the updating and propagating of NIS maps on slave servers, you can install shell scripts to run as `crontab` jobs. Because maps have different rates of change, scheduling a map transfer using the `crontab` command enables you to set specific propagation times for individual maps.

Sun provides several template scripts in the `/usr/lib/netsvc/yp` directory that you can use and modify to meet local site requirements. These scripts are useful when slave servers are down during NIS map propagations. In such cases, the slave server might not receive the update unless you run a “safety valve” script (Figure 12-5).

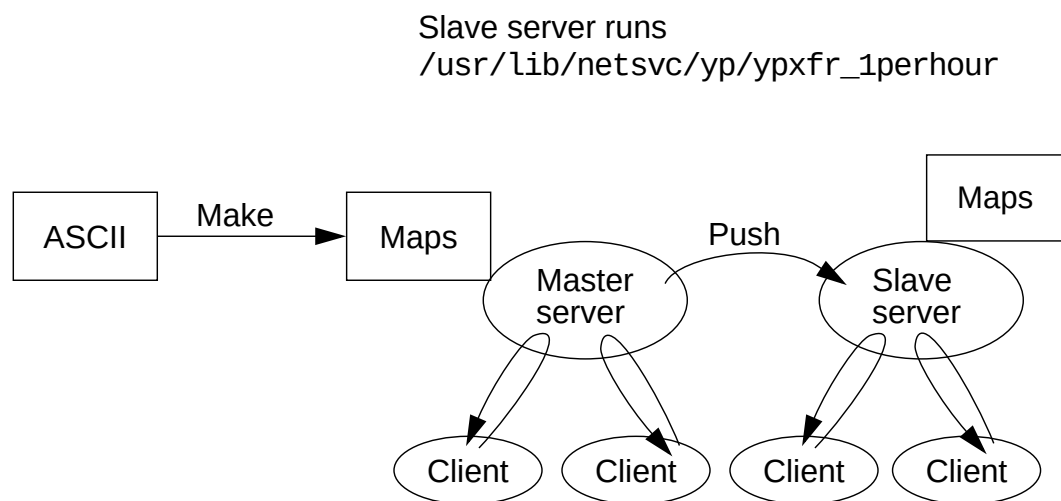


Figure 12-5 Updating passwd Maps on Slave Servers with Scripts

The following text is the contents of the `ypxfr_1perhour` script that, if run hourly using `cron`, ensures that the NIS slave server's `passwd` map is never more than one hour out of date.

```
#!/bin/sh
#
# @(#)ypxfr_1perhour.sh 1.9 92/12/18 Copyright 1999 Sun Microsystems,
# Inc.
#
# ypxfr_1perhour.sh - Do hourly NIS map check/updates
#

PATH=/bin:/usr/bin:/usr/lib/netsvc/yp:$PATH
export PATH

# set -xv
ypxfr passwd.byname
ypxfr passwd.byuid
```

Updating Other Scripts

There are scripts called `ypxfr_1perday` and `ypxfr_2perday`. The `ypxfr_1perday` script checks or updates the following maps daily:

- `group.byname`
- `group.bygid`
- `protocols.byname`
- `protocols.bynumber`
- `networks.byname`
- `networks.byaddr`
- `services.byname`
- `ypservers`

The `ypxfr_2perday` script checks and updates the following NIS maps twice per day:

- `hosts.byname`
- `hosts.byaddr`

- `ethers.byaddr`
- `ethers.byname`
- `netgroup`
- `netgroup.byuser`
- `netgroup.byhost`
- `mail.aliases`

Makefile *Syntax and New Maps*

You can build custom NIS maps for use with local utilities or with Sun utilities, such as the automounter. By generating an NIS automounter map and setting up all NFS clients appropriately, NFS-mountable resources become available over the network with minimum administration. Changes need to be made only to the NIS master server, and the NFS server grants NFS access to clients within the entire NIS domain.

The make Utility

Building customized NIS maps is essentially a lesson in the `make` utility. The `make` utility:

- Is used by programmers to build programs
- Is used by administrators to build NIS maps
- Can be generalized to build customized NIS maps

The `make` utility receives its instructions from the `Makefile` file. The `Makefile` uses variable definitions (called *macros*), targets, and dependencies.

You can use macros as variables, similar to those that are used in a shell script. A macro is defined at the beginning of the `Makefile` and is used throughout the `Makefile` by prefixing the macro name with a dollar sign (\$).

The `make` utility builds *targets*. Targets need dependencies. *Dependencies* can represent other targets that must be built before the original target is considered “made.” This structure enables you to nest the target and dependency pairs to an arbitrary depth, allowing for hierarchical building of complex code structures.

When making NIS maps, you should keep the target and dependency relationship is fairly simple.

First Section of Makefile

The NIS Makefile is located in the `/var/yp` directory and is composed of four main sections. The first section contains the following macro definitions:

```
#B=-b
B=
DIR =/etc
PWDIR =/etc
DOM = `domainname`
NOPUSH = ""
ALIASES = /etc/mail/aliases
YPDIR=/usr/lib/netsvc/yp
SBINDIR=/usr/sbin
YPDBDIR=/var/yp
YPPUSH=$(YPDIR)/yppush
MAKEDBM=$(YPDIR)/makedbm
MULTI=$(YPDIR)/multi
REVNETGROUP=$(SBINDIR)/revnetgroup
STDETHERS=$(YPDIR)/stdethers
STDHOSTS=$(YPDIR)/stdhosts
MKNETID=$(SBINDIR)/mknetid
MKALIAS=$(YPDIR)/mkalias
```

Second Section of Makefile

The second section contains the first target, `all`.

```
all: passwd group hosts ethers networks rpc services protocols \
    netgroup bootparams aliases publickey netid netmasks c2secure \
    timezone auto.master auto.home auth.attr exec.attr prof.attr \
    user.attr audit.user
```

The `all` target has several dependencies, each of which represents one of the NIS maps to be built. This enables the entire set of NIS maps to be built by typing:

```
# cd /var/yp; make
```

The `all` target is not considered to be built until each of its targets is first built in turn. Each of the targets for `all` depends on another target.

When adding custom maps to NIS, the name of the new map to be built should be added to the end of the `all` target list (`auto.direct` in the following example).

```
all: passwd group hosts ethers networks rpc services protocols \  
    netgroup bootparams aliases publickey netid netmasks c2secure \  
    timezone auto.master auto.home auth.attr exec.attr prof.attr \  
    user.attr audit.user auto.direct
```

Fourth Section of Makefile

The entry in the fourth section of the `Makefile` for each of the dependencies specified in the `all` target is:

```
passwd: passwd.time  
group: group.time  
hosts: hosts.time  
ethers: ethers.time  
networks: networks.time  
rpc: rpc.time  
services: services.time  
protocols: protocols.time  
netgroup: netgroup.time  
bootparams: bootparams.time  
aliases: aliases.time  
publickey: publickey.time  
netid: netid.time  
passwd.adjunct: passwd.adjunct.time  
group.adjunct: group.adjunct.time  
netmasks: netmasks.time  
timezone: timezone.time  
auto.master: auto.master.time  
auto.home: auto.home.time  
$(DIR)/netid:  
$(DIR)/timezone:  
$(DIR)/auto_master:  
$(DIR)/auto_home:  
$(PWDIR)/shadow:
```

Using the previous example of an `auto.direct` map, add a new map to the NIS domain by appending the appropriate entries to the end of this “second level” target/dependency pair.

```
...
auto.direct: auto.direct.time
...
$(DIR)/auto_direct:
```

Therefore, the final lines from the fourth section would look like this after the `auto.direct` map was modified.

```
auto.master: auto.master.time
auto.home: auto.home.time
auto.direct: auto.direct.time
$(DIR)/netid:
$(DIR)/timezone:
$(DIR)/auto_master:
$(DIR)/auto_home:
$(DIR)/auto_direct:
$(PWDIR)/shadow:
```

The target in this case, `auto.direct`, depends on another target, `auto.direct.time`.

Third Section of Makefile

In the third section of the Makefile, the final target and dependencies are defined, along with instructions on how to build each map in the domain.

You must add the following lines to the Makefile to build a new `auto_direct` map:

```
auto.direct.time: $(DIR)/auto_direct
    -@if [ -f $(DIR)/auto_direct ]; then \
        sed -e "/^#/d" -e s/#.*$$// $(DIR)/auto_direct \
        | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/auto.direct; \
        touch auto.direct.time; \
        echo "updated auto.direct"; \
        if [ ! $(NOPUSH) ]; then \
            $(YPPUSH) auto.direct; \
            echo "pushed auto.direct"; \
        else \
            : ; \
```

```

        fi \
else \
    echo "couldn't find $(DIR)/auto_direct"; \
fi

```

You should be aware of the following:

- `auto.home.time` depends on `$(DIR)/auto_home`.

In this case, the dependency is a file. The `make` utility checks the timestamp of the target (assumed to be a file, and in the current directory) against the timestamp of the dependency (usually in the `/etc` directory). If the target has a newer modification time than the dependency, the target is not built, and this section is skipped. If, on the other hand, the dependency has a more recent modification timestamp, the target is built according to the instructions in the section that immediately follows.

- Subsequent lines of `make` instructions are indented by tabs. (This is required.)
- You can use `make` macros in the instructions.
- Instructions that begin with the at (`@`) sign are *not* echoed to the screen. Removing the `@` sign is useful for debugging new instructions.
- Instructions that begin with a leading dash (`-`), occurring before the leading `@` sign, do *not* have error messages echoed to the terminal.

Building NIS Maps

Most map builds consist of the following sequence of actions, which are specified in the third section of the `Makefile`:

1. Extract the source file key and value pairs of information using the `awk` or `sed` commands.
2. Send these key and value pairs to the `makedbm` program to generate the NIS map.

3. Use `touch` on the timestamp file so that this map is not remade unless and until the source file is updated.
4. Echo the message stating the map has been updated.
5. Push the map to the slave servers.
6. Echo a message stating the push is done.



Caution – The first time you build a new map, the slave servers do not know of its existence and so the push process attempt fails. Send an interrupt (Control-c) to the build process when the push process hangs, and execute the `ypxfr` command on the map from the slave server(s) to complete the build process. (This is necessary only during the first build of a new NIS map.)

Note – Details of the `awk`, `sed`, and `makedbm` programs are beyond the scope of this class.

Exercise: Configuring NIS



Exercise objective – In this lab, you configure a NIS master server and one NIS client.

Preparation

Choose two partners for this lab and determine which systems will be configured as the NIS master server and which will serve as the NIS slave and NIS client. You use the NIS master as the Jumpstart server later in the course. Verify that entries for all systems exist in the `/etc/hosts` file. Refer to your lecture notes as necessary to perform the steps listed.

Task Overview

In this exercise, you accomplish the following:

- On the system that will become the NIS master server, replace the `/var/yp/Makefile` file with an updated copy that references `/etc/locale`. Your instructor will provide the `Makefile`.
- Use the following commands and files to create and configure an NIS master server. Configure only a master server. Verify that the configuration works using the `ypwhich -m` command.

```
/etc/nsswitch.nis  
/etc/nsswitch.conf  
/etc/defaultdomain  
/etc/ethers  
/etc/bootparams  
/etc/locale  
/etc/netmasks  
/etc/timezone  
/etc/security/audit_user  
/etc/security/auth_attr  
/etc/security/exec_attr
```

```
/etc/security/prof_attr  
domainname  
ypinit  
ypstart
```

- Use the following commands and files to create and configure an NIS client. Verify that the configuration works using the `ypwhich -m` command.

```
/etc/nsswitch.nis  
/etc/nsswitch.conf  
domainname  
/etc/defaultdomain  
ypinit  
ypstart/etc/nsswitch.nis  
/etc/nsswitch.conf  
/etc/defaultdomain  
/etc/ethers  
/etc/bootparams  
/etc/locale  
/etc/netmasks  
/etc/timezone  
/etc/security/audit_user  
/etc/security/auth_attr  
/etc/security/exec_attr  
/etc/security/prof_attr  
domainname  
ypinit  
ypstart
```

- Use the following commands and files to create and configure an NIS slave server. Verify that the configuration works using the `ypwhich -m` command.

```
/etc/nsswitch.nis  
/etc/nsswitch.conf  
domainname  
/etc/defaultdomain  
ypinit  
ypstart  
ypcat
```

- Use the following commands and files to create and update a new NIS map, `auto.home`. Verify that the new map works using the `ypcat` command.

```
/etc/nsswitch.nis
/etc/nsswitch.conf
domainname
/etc/defaultdomain
ypinit
ypstart
```

Tasks

Perform the following steps to create the NIS master and configure the NIS maps.

Setting up the NIS Master

1. Install an updated `Makefile` in the `/var/yp` directory. If you intend to use NIS on the system that will act as your JumpStart server, you must update the `/var/yp/Makefile` file so that it creates a map from `/etc/locale` file.
2. If available, get the modified `Makefile` from the classroom server (includes the entries for `locale` discussed in step 8 of the “Configuring the NIS Master Server” section on page 12-13).

Note – The exact directory location on the server might be different from the one in this example.

```
# cd /net/classroom_server/export/software
# cp Makefile /var/yp/Makefile
```

Refer to “Configuring the NIS Master Server” section on page 12-13 for more information about the changes required to include `/etc/locale` in the `Makefile`.

3. Create the file `/etc/locale` and make an entry for your NIS domain.

```
your.domain      en_US
```


4. Add the `timehost` alias to the `/etc/hosts` entry for the NIS master, for example:

```
172.16.64.103      nis_master  loghost    timehost
```

5. Because this system will be the source for NIS database information supplied to the network, you can comment out the `+auto_master` entry in the `/etc/auto_master` file.

```
# Master map for automounter#
# +auto_master
/net      -hosts      -nosuid,nobrowse
/home     auto_home    -nobrowse
/xfn      -xfn
```

6. For the same reason given in the previous step, you can comment out the `+auto_home` entry in the `/etc/auto_home` file. Modify two lines in the `/etc/auto_home` file by commenting out the single entry for `+auto_home` and adding a global entry for all user accounts located in `/export/home` on the NIS master as shown:

```
# Home directory map for automounter
#
# +auto_home
*      nis_master:/export/home/&
```

7. Use the `touch` command to create the following files.

```
# cd /etc
# touch ethers bootparams netgroup
```

8. Create the `/etc/timezone` file with the appropriate entry for your timezone. The example that follows is for the US/Mountain timezone in the `Central.Sun.COM` domain.

```
US/Mountain      your.domain
```

9. Copy the `/etc/nsswitch.nis` file to the `/etc/nsswitch.conf` file.

```
# cd /etc
# cp nsswitch.nis nsswitch.conf
```

10. Select a name to use as your NIS domain name. Set it using the `domainname` command. Replace `nisdomain` with your own domain name.

```
# domainname your.domain
```

11. Populate the `/etc/defaultdomain` file with your domain name.

```
# domainname > /etc/defaultdomain
```

12. Create the following entries in the `/etc/dfs/dfstab` file for the user home directories and manual pages:

```
share -F nfs -d "home dirs" /export/home
```

13. Share the new file system entries to the network:

- a. If no directories are currently shared by the NIS master, run the following command to start sharing the file systems configured in the `/etc/dfs/dfstab` file:

```
# /etc/init.d/nfs.server start
```

- b. If the NIS master is already sharing other directories, run the `shareall` command to share the new entries created in the `/etc/dfs/dfstab` file.

```
# shareall
```

14. Create user accounts for all users in your group. Create their respective home directories in `/export/home`. For example, `/export/home/user1` for user1, `/export/home/user2` for user2, and so on.

15. To enable the automatic mounting of a user's directory on any host in the NIS domain, modify the user entries in the `/etc/passwd` file from `/export/home/username` to `/home/username`.

16. Use the `ypinit` command as follows to set up this system as a NIS master:

- a. Run `ypinit -m` to start the setup process.

```
# ypinit -m
```

- b. The `ypinit` command lists the current system as an NIS server, and then asks for the next host to add as a NIS slave server. In this lab, you set up only a master and a client. Press Control-d to terminate the list.

```
next host to add: master_server
next host to add: slave_server
next host to add: Control-d
(list of servers)
is this list correct? [y/n: y] y
```

- c. Indicate that you do not want `ypinit` to quit on nonfatal errors. The `ypinit` command then proceeds to build the required maps.

```
...quit on nonfatal errors? [y/n: n] n
```

If the initialization process is successful, `ypinit` displays a message indicating that the current system was set up as a Master server without any errors. This message is displayed even if nonfatal errors were encountered in the procedure.

If the initialization process fails, correct the problems indicated by the error messages and repeat steps a, b, and c.

17. Start the NIS daemons.

```
# /usr/lib/netsvc/yp/ypstart
```

18. Once the `ypserv` daemon is running you should run the `make` command to build the mail aliases map.

```
# cd /var/yp
```

```
# /usr/ccs/bin/make
```

19. Use the `ypwhich` command to verify that this system is the NIS master.

```
# ypwhich -m
```

Setting up the NIS Slave Server

20. Copy the `/etc/nsswitch.nis` file to the `/etc/nsswitch.conf` file.

```
# cd /etc
# cp nsswitch.nis nsswitch.conf
```

21. Edit the `/etc/hosts` file to ensure that the NIS master servers has been defined.

22. Use the `domainname` command to set the NIS domain for this client. Replace *nisdomain* with your own domain name.

```
# domainname your.domain
```

23. Populate the `/etc/defaultdomain` file with your domain name.

```
# domainname > /etc/defaultdomain
```

24. Use the `ypinit` command as follows to set up this system as an NIS client:

a. Run `ypinit -c` to start the setup process.

```
# ypinit -c
```

b. When prompted for a list of NIS servers, enter the NIS master host followed by the name of the local host (which subsequently becomes a slave server) and all other NIS slave servers on the local network, then enter Control-d to terminate the list.

```
next host to add: master_server
next host to add: slave_server
next host to add: Control-d
(list of servers)
is this list correct? [y/n: y] y
#
```

25. Start the NIS daemons.

```
# /usr/lib/netsvc/yp/ypstart
```

26. Use the `ypwhich` command to verify that this system is using NIS and is bound to the NIS master.

```
# ypwhich -m
```

27. Initialize the system as an NIS slave with the following command:

```
# ypinit -s master_server
```

where *master_server* is the name of the NIS master and *slave_server* is the name of the local host.

Note – If you did not add the name of the NIS slave server when you initially configured the NIS master using the `ypinit` command, run the **`ypinit -m`** command once more on the NIS master and add the slave server's host name. In the process of updating the NIS master, the script prompts you for confirmation when it is about to destroy the existing domain database. Confirm with **y**.

28. Stop the NIS daemons on the slave server with the following command:

```
# /usr/Lib/netsvc/yp/ypstop
```

29. Restart the NIS daemons on the slave server with the following command:

```
# /usr/Lib/netsvc/yp/ypstart
```

30. On the newly configured NIS slave server, test the NIS functionality by entering the following command:

```
# ypwhich -m
```

The output should include the name of the NIS master server along with a list of database maps it is serving to the NIS domain.

31. Use `ypcat` to list the `ypservers` known to the local domain. The output should include the name of the local host you just configured as an NIS slave server:

```
# ypcat -k ypservers
master_server
slave_server
```

Setting up the NIS Client

32. Copy the `/etc/nsswitch.nis` file to the `/etc/nsswitch.conf` file.

```
# cd /etc
# cp nsswitch.nis nsswitch.conf
```

33. Use the `domainname` command to set the NIS domain for this client. Replace *nisdomain* with your own domain name.

```
# domainname your.domain
```

34. Populate the `/etc/defaultdomain` file with your domain name.

```
# domainname > /etc/defaultdomain
```

35. Use the `ypinit` command as follows to set up this system as an NIS client:

a. Run `ypinit -c` to start the setup process.

```
# ypinit -c
```

b. The `ypinit` command asks for a list of NIS servers, and for you to add them in order of preference. Enter the name of the NIS master server, then the NIS slave server to be configured in the next section of the lab, and press Control-d to terminate the list.

```
next host to add: master_server
next host to add: slave_server
next host to add: Control-d
(list of servers)
is this list correct? [y/n: y] y
```

36. Start the NIS daemons.

```
# /usr/lib/netsvc/yp/ypstart
```

37. Use the `ypwhich` command to verify that this system is using NIS.

```
# ypwhich -m
```

Testing Dynamic Rebind

38. On the previously-configured NIS client, confirm that it is still bound to the NIS master by entering the following command:

```
# ypwhich
master_server
```

The output should contain *only* the name of the NIS master server.

39. Test the ability of the client to bind with the NIS slave server when the master is unavailable:

Note – This does *not* work if you entered *both* the NIS master and the NIS slave host names when you ran the `ypinit -c` command. It searches only for `ypservers` listed in the `/var/yp/binding/domainname/ypservers` file, which is created as a result of running the `ypinit -c` command.

- a. Abort the NIS master to run level 0 (the monitor level) using the Stop-A key sequence. You should see the ok prompt:

ok

- b. Return to the NIS client and determine to which NIS server it is bound. This can take a minute or two for the client to rebroadcast for a server and bind with the NIS slave.

Allowing a few moments for the NIS client to bind to the slave server, the output should contain *only* the name of the NIS slave server.

```
# ypwhich
slave_server
```

During the period while the NIS client is attempting to bind with the NIS slave server, the response from the `ypwhich` command is shown as follows:

```
# ypwhich
Domain domainname not bound on hostname.
```

For example:

```
# ypwhich
Domain classroom.Central.Sun.COM not bound on host1.
```

40. On the NIS master, type `go` at the ok prompt.

ok go

Adding a Custom Map to the NIS Master Database

If the custom `auto_direct` map entry does not already exist in the `Makefile` you acquired from the classroom server, use the following steps to add it.

41. Add `auto.direct` at the end of the list of maps that exist in the *second* section of the `/var/yp/Makefile`:

```
all: passwd group hosts ethers networks rpc services protocols \
    netgroup bootparams aliases publickey netid netmasks c2secure \
    timezone auto.master auto.home auth.attr exec.attr prof.attr \
    user.attr audit.user auto.direct
```

42. Add `auto.direct` at the end of the list of maps that exist in the *forth* section of the `/var/yp/Makefile`:

The final lines from the *fourth* section should look like the following after you modified the after the `auto.direct` map.

```
auto.master: auto.master.time
auto.home: auto.home.time
auto.direct: auto.direct.time
$(DIR)/netid:
$(DIR)/timezone:
$(DIR)/auto_master:
$(DIR)/auto_home:
$(PWDIR)/shadow:
$(DIR)/auto_direct:
```

43. If not already there, add the following lines to the `Makefile` in the *third* section to build a new `auto_direct` map:

```
auto.direct.time: $(DIR)/auto_direct
    -@if [ -f $(DIR)/auto_direct ]; then \
        sed -e "/^#/d" -e s/#.*$$// $(DIR)/auto_direct \
        | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/auto.direct; \
        touch auto.direct.time; \
        echo "updated auto.direct"; \
        if [ ! $(NOPUSH) ]; then \
            $(YPPUSH) auto.direct; \
            echo "pushed auto.direct"; \
        else \
            : ; \
        fi \
    else \
        echo "couldn't find $(DIR)/auto_direct"; \
    fi
```


44. Create an entry in the `/etc/auto_master` file to reference a new map, `auto_direct` (to be configured in subsequent steps):

```
# Master map for automounter
#
#+auto_master
/net          -hosts          -nosuid,nobrowse
/home         auto_home       -nobrowse
/xfn          -xfn
/-            auto_direct      -nosuid
```

45. On all hosts, rename the existing `/usr/share/man` directory to `/usr/share/man2`:

```
# mv /usr/share/man /usr/share/man2
```

46. Create the `/etc/auto_direct` file with the following:

```
/usr/share/man -ro nis_master:/usr/share/man2
```

47. Add an entry to the `/etc/dfs/dfstab` file on the NIS master to share `/usr/share/man2` and share the directory.

```
# vi /etc/dfs/dfstab
share -o ro /usr/share/man2
# shareall
```

48. Change to the `/var/yp` directory.

```
# cd /var/yp
```

49. Refresh the NIS database maps by executing the `make` command. This command hangs because a new `auto.direct` map has been added to the NIS master and `make` attempts to push it to the NIS slave server who refuses to accept it. Press `Control-C` to interrupt the `make` command.

```
# /usr/ccs/bin/make
```

50. To force the NIS slave server to manually pull this map, issue the following command:

```
# /usr/lib/netsvc/yp/ypxfr auto.direct
```

51. Reboot all NIS slaves and NIS clients using the `init 6` command:

```
# init 6
```

52. Log in to NIS slave and NIS client to test previously created (non-root) NIS user accounts.
53. Verify that your home directory automounted from the NIS master server.

\$ pwd

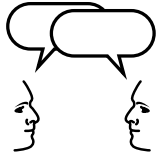
54. Attempt to access the /usr/share/man directory using the man command.

\$ man ls

\$ mount

If the content of the man page for `ls` is displayed, you were successful in creating the new `/etc/auto_direct` map and having the mount take place automatically as a result of the command requiring access to the man pages in `/usr/share/man`.

Exercise Summary



Discussion – Take a few minutes to discuss the experiences, issues, or discoveries that you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ Describe the NIS components, master server, slave server, and client, and the NIS processes
- ☐ Configure an NIS master, slave, and client
- ☐ List the steps to add a new NIS map
- ☐ Use commands to update and propagate an NIS map

Objectives

Upon completion of this module, you should be able to:

- Define the JumpStart™ program
- List the main components for setting up a JumpStart network installation
- Briefly describe the `add_install_client` script syntax
- Set up the boot services function on a subnet using the `setup_install_server` script
- Describe the JumpStart client boot sequence
- List the files necessary to support the JumpStart boot operation
- Describe the syntax of the `add_install_client` script as it relates to the boot server
- Describe the use of the `sysidcfg` file with and without name service support
- Set up a JumpStart installation server system to provide the Solaris 8 Operating Environment with the software necessary to install clients.
- Describe the use of the `add_to_install_server` and `modify_install_server` scripts
- Add install clients to the install servers and boot servers
- Create a configuration server with customized rules and class files

- Use the `pfinstall` command to test configuration and installation files
- Boot install clients
- Configure NIS name service support for the JumpStart program

Additional Resources

Additional resources – The following references provide additional details on the topics discussed in this module:

- *Solaris 8 Advanced Installation Guide*, Part Number 806-0957-10
- *Automating Solaris Installations - A Custom JumpStart™ Guide*, by Paul Anthony Kasper and Alan L. McClellan, ISBN 0-13-312505-X

Introduction to JumpStart

The JumpStart program is an automatic installation (auto-install) process available in the Solaris 8 Operating Environment. It enables you to categorize machines on your network and automatically install systems based on the category (*class*) to which a machine belongs.

The auto-install feature is complementary to the interactive install program, SunInstall™; you can use either program to install a system.

Who Should Use JumpStart and Why?

For those network and system administrators tasked with installing the Solaris 8 Operating Environment on several networked hosts, the JumpStart program can offer the capability of a custom installation without operator intervention. One advantage of doing automatic installations in this manner is that you can install additional, unbundled software to the operating system.

The advantages to using the JumpStart program include the following:

- Frees system administrators from the lengthy question and answer session that is part of the installation process
- Enables system administrators to install several standalone, dataless, or server systems or both in a day
- Installs the operating environment and other additional unbundled software applications in a short period of time on multiple networks
- Simplifies administration tasks when widely-used applications must be updated frequently

The JumpStart program provides networked computing environments with considerable time savings when multiple or ongoing installations are necessary.

JumpStart Components

There are three main components to the JumpStart program:

- Boot and system identification services – Provided by a networked boot server that offers boot parameters and identification information to JumpStart clients.

The boot parameters include the names for the boot server, install server, and configuration server with their respective directory paths for NFS mount by the JumpStart clients. If the install server resides on a different subnet, the boot server enables the initial boot on the local subnet, then directs the client to continue the installation from the subnet on which the install server resides. All of these functions can be provided by a single JumpStart server on the local subnet.

- Installation services – Provided by an install server that shares the Solaris Operating Environment installation software to the network for NFS mount by the JumpStart clients.

The install server shares the Solaris Operating Environment software image from a directory location on the local disk or CD-ROM. During the installation process, the Solaris software image is mounted from the install server to the JumpStart client file system using NFS.

- Configuration services – Provided by a *configuration* server that shares a directory of configuration files to the JumpStart clients.

Typically, the directory of files located on the configuration server include system specifications and rules for installation that apply to each JumpStart client.

If any one of these three main components is improperly configured, the JumpStart clients can:

- Fail to boot
- Fail to find a Solaris Operating Environment image to load
- Ask questions interactively for configuration
- Fail to start partitioning the disks, creating file systems, and loading the operating environment.

Adding Install Clients

Because each of the three main components of JumpStart is installed on three different types of servers, the `add_install_client` command must reference each component using the appropriate options and arguments.

Command Syntax

```
# ./add_install_client -e ethernet_addr -i ip_addr \  
-s install_svr:/distr -c config_svr:/config_dir \  
-p sysid_config_svr:/config_dir client_name client_arch
```

If you use a single server named `server1` to provide all JumpStart components (taking on the role of the boot, install, and configuration servers), use the following `add_install_client` command to create a client named `client1` with a `sun4u` architecture as an example of the necessary command options and arguments:

```
# ./add_install_client -e 8:0:20:2f:90:3d -i 192.9.200.2\  
-s server1:/export/install -c server1:/export/config \  
-p server1:/export/config client1 sun4u
```

Setting Up Boot Services

Boot services are provided by a networked boot server. During the initial boot process, the JumpStart client obtains boot, install, and configuration server designations from this server.

This section describes the configuration of the JumpStart boot server including:

- The JumpStart client booting sequence
- The boot operation support files
- Adding a bootable Solaris Operating Environment image
- Using the `add_install_client` script to specify a boot server that is separate from the install server

JumpStart Client Boot Sequence

Figure 13-1 illustrates the JumpStart client boot process.

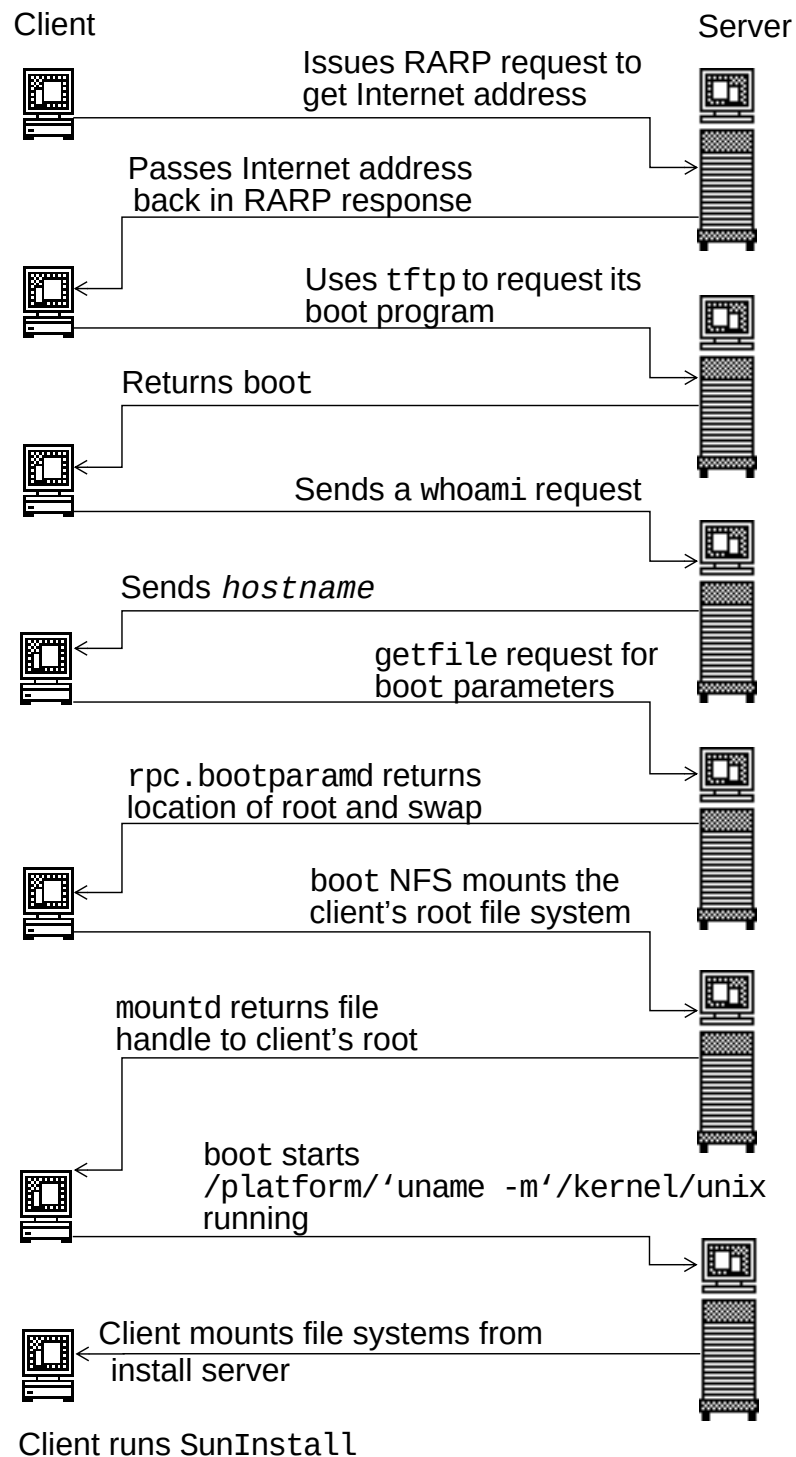


Figure 13-1 The JumpStart Client Boot Process

The following steps describe how the JumpStart process works:

1. When a network workstation is booted, the boot programmable read-only memory (PROM) issues a Reverse Address Resolution Protocol (RARP) broadcast to the network. The RARP request received by the JumpStart server translates the Ethernet address to an Internet address.

The boot server running the RARP daemon, `/usr/sbin/in.rarpd`, looks up the Ethernet number in the `/etc/ethers` file, checks for a corresponding name in the `/etc/hosts` file, and passes the Internet address back to the client.

2. The client's boot PROM sends a Trivial File Transfer Protocol (TFTP) request for its boot program. It broadcasts its Internet address in hexadecimal format. To observe this network exchange taking place, you can `boot` the client from the `ok` prompt using the `-v` option. For example,

```
ok boot net -v
Boot device:/pci@1F,0/pci@1,1/network@1,1 File and args: -v
2b000 Using RARP/BOOTPARAMS...
Internet address is: 192.9.200.100
hostname: client1
...
...
```

3. The server searches for a symbolic link with the Internet Protocol (IP) address in hexadecimal format as the file name in its `/tftpboot` directory. For SPARC™ machines, the file name is *hex-IP-address.architecture*.

```
C009C864.SUN4U -> inetboot.sun4u.Solaris_8-1
```

4. The server uses the `in.tftpd` daemon to copy the boot program into the client's memory. The client then runs the `boot` program.

5. The `boot` program tries to mount the `root` file system. To do so, it issues a `whoami` request to discover the client's host name. A server running the boot parameter daemon, `rpc.bootparamd`, looks up the host name, and responds to the client. Then, the `boot` program issues a `getfile` request to obtain the name of the client's server and the location of the client's root and swap space.
6. The server responds with the information obtained from the `/etc/bootparams` file.
7. Once the client has its boot parameters, the boot program on the client mounts the `/` (root) file system from the boot server and `/platform/'uname -m'/kernel/unix`, which starts the `init` program. When the boot server is finished bootstrapping the client, it redirects the client to the configuration server.
8. The client searches for the configuration server using `bootparams` information. The client mounts the configuration directory and runs `sysidtool`. The client then uses the information in the `bootparams` table to locate and mount the installation directory that which contains the Solaris software image from the install server. The client then runs the `SunInstall` program and installs itself.

Boot Operation Support Files

The JumpStart boot operation support files that must be available to the client during installation include `/etc/ethers`, `/etc/hosts`, `/etc/bootparams`, `/etc/dfs/dfstab`, and the `/tftpboot` directory.

On a networks running the NIS or NIS+ name services, the identification information from the JumpStart server files must also be incorporated in the domain database maps.

The /etc/ethers File

When the JumpStart client boots, it has no IP address so it broadcasts to the network using RARP and its Ethernet address. The JumpStart server receives the request and attempts to match the client's Ethernet address with an entry in the local `/etc/ethers` file.

If a match for the Ethernet number is found, the client name is matched to an entry in the `/etc/hosts` file. In response to the RARP request from the client, the JumpStart server sends the IP address from the `/etc/hosts` file back to the client. The client then continues the boot process using the IP address.

If a match is not found, the client cannot acquire its IP address and cannot continue the boot process. The usual (repeating) message displayed on the screen of a JumpStart client when this occurs is the following:

Timeout waiting for ARP/RARP packet

An entry for the JumpStart client can be created by editing this file or as one of the arguments to the `add_install_client` script. The following example is an entry in the `/etc/ethers` file for a JumpStart client:

```
8:0:20:2f:90:3d client1
```

The /etc/hosts File

The `/etc/hosts` file is the local database that associates the names of hosts with their IP addresses. The JumpStart server references this file when trying to match an entry from the local `/etc/ethers` file in response to a RARP request from a client.

If a match is not found, the client cannot acquire its IP address and cannot continue the boot process. The usual (repeating) message displayed on the screen of a JumpStart client when this occurs is the following:

Timeout waiting for ARP/RARP packet

An entry for the JumpStart client can be created by editing this file or as one of the arguments to the `add_install_client` script. The following example is an entry in the `/etc/hosts` file for a JumpStart client:

```
192.9.200.100 client1
```

The /tftpboot Directory

The /tftpboot directory contains the `inetboot.SUN4x.Solaris_8-1` file that is created for each JumpStart client when the `add_install_client` script is run. When booting over the network, the client's boot PROM makes a RARP request and when it receives a reply, the PROM broadcasts a TFTP request to fetch the `inetboot` file from any server that responds and executes it. For example, the `inetboot` file created for a JumpStart client with a sun4u architecture is named `inetboot.SUN4U.Solaris_8-1`. Two additional symbolic links to this file are also created at the same time containing the IP address and the architecture of the client system. The following is the output of a long listing of a /tftpboot directory that supports one sun4u client with an IP address of 192.9.200.100 appears in this manner:

```
# ls -l /tftpboot
total 344
lrwxrwxrwx  1 root    other          26 Apr 15 21:20 C009C864 ->
inetboot.SUN4U.Solaris_8-1*
lrwxrwxrwx  1 root    other          26 Apr 15 21:20 C009C864.SUN4U ->
inetboot.SUN4U.Solaris_8-1*
-rwxr-xr-x  1 root    other        159768 Apr 15 21:20
inetboot.SUN4U.Solaris_8-1*
-rw-r--r--  1 root    other          315 Apr 15 21:20 rm.192.9.200.100
```

The `inetboot` program makes another RARP request, then uses the `bootparams` protocol to locate its root file system. It then mounts the root file system across the network using the NFS protocol and runs the kernel.

If the files in the /tftpboot directory are unavailable to the JumpStart client when the boot process is initiated, the client cannot retrieve `bootparams` information for the root file system and stops the boot process without displaying an error message.

The /etc/bootparams File

The `/etc/bootparams` file contains entries that network clients use for booting. JumpStart clients retrieve the information from this file by issuing requests to a server running the `rpc.bootparamd` program. The `/etc/bootparams` file can be used in conjunction with, or in place of, other sources for the bootparams information. When the JumpStart client makes the request, the server references the `/etc/bootparams` file and responds with the file system information required for NFS mount to enable network installation.

If the required entries are not in the `/etc/bootparams` file, the JumpStart client cannot determine the appropriate server and file system to mount, and stops at the beginning of the boot process without displaying an error message.

Entries in this file are created by the options and arguments entered as part of the `add_install_client` script. The following example is an entry in the `/etc/bootparams` file for a JumpStart client named `client1`:

```
client1
root=server1:/export/install/Solaris_8/Tools/Boot
install=server1:/export/install
boottype=:in
sysid_config=server1:/export/config
install_config=server1:/export/config
rootopts=:rsize=32768
```

The following lists describes the entries:

- `client1` – The JumpStart client name
- `root=server1:/export/install/Solaris_8/Tools/Boot` – The boot server name and directory for the root file system
- `install=server1:/export/install` – The server name and directory for the Solaris software image
- `boottype=:in` – Indicates a network boot and installation
- `sysid_config=server1:/export/config` – The server name and directory for the JumpStart configuration file system
- `install_config=server1:/export/config` – The server name and directory for the operating environment installation files

- `rootopts=:rsize=32768` – Mount options for the root file system and NFS read size

The /etc/dfs/dfstab File

The `/etc/dfs/dfstab` file lists local file systems to be shared to the network. Typically, when you initially set up the JumpStart server, you must manually update this file with an entry for the configuration directory you want to share to the network to support remote installation. This file is again populated with the installation directory location as a result of the `add_install_server` script.

If the required entries are not in the `/etc/dfs/dfstab` file, the JumpStart client cannot mount the file systems specified in the `/etc/bootparams` and displays the following error message:

```
panic - boot: Could not mount filesystem
```

```
Program terminated  
ok
```

Adding a Bootable Image

To enable the client for JumpStart network installation, you must set up an install server, boot server, and a configuration server (see the "JumpStart Components" on page 13-4).

You can set up a *boot* server that uses the Solaris 8 software image located on the CD-ROM by using the following steps:

1. Ensure that the system has an empty directory (/export/install, for example) with approximately 156 Mbytes of available disk space
2. Insert the Solaris 8 Software CD-ROM 1 of 2 in the CD-ROM drive, allowing vold to automatically mount the CD-ROM
3. Change the directory to the location of the setup_install_server script

```
# cd /cdrom/cdrom0/s0/Solaris_8/Tools
```

4. Run the setup_install_server script:

```
# ./setup_install_server -b /export/install
```

Note – The add_install_client to create JumpStart clients and the rm_install_client to remove an existing JumpStart client are also in this directory.

Using add_install_client

You can create a JumpStart client using a server named *server1* to provide *only the boot function* JumpStart component. You use another system as the install server (named *server2*) for the Solaris 8 software image installation and configuration components by running the following *add_install_client* command to create a client named *client1* with a sun4u architecture:

```
# ./add_install_client  
-s server2:/export/install -c server2:/export/config \  
-p server2:/export/config client1 sun4u
```

You run the previous command from the `/export/install/Solaris_8/Tools` directory on the boot server (server1). The arguments to options `-s`, `-c`, and `-p`, redirect the JumpStart clients to server2 for the configuration information and the Solaris 8 software image.

Setting Up Client Identification

When a JumpStart client boots for the first time, the booting software tries to obtain system identification information about the system (such as the system's host name, IP address, locale, timezone, and root password) from a `sysidcfg` file *first* and then the name service database. This means that you can use a `sysidcfg` file to answer system identification questions during the initial part of the installation regardless of whether a name service (NIS or NIS+) is used. If the JumpStart server provides this information, the client bypasses the initial system identification portion of the Solaris 8 Operating Environment installation process without administrator intervention.

Without the `sysidcfg` file or a name service database, the client displays the appropriate interactive dialog boxes to request identification information that is not provided.

Using the sysidcfg File to Identify a Client

In the absence of a name service on the network, the sysidcfg file must be present to automate system identification.

Table 13-1 lists the keywords and arguments used in the construction of the sysidcfg file.

Table 13-1 Keywords and Arguments of the sysidcfg File

Keyword	Argument
name_service {domain_name}	name_service=NIS, NIS+, OTHER, NONE Options for NIS and NIS+: {domainname=<i>domain_name</i> name_server=hostname(<i>ip_address</i>)} Options for DNS: {domainname=<i>domain_name</i> name_server=<i>ip_address</i>, <i>ip_address</i>, <i>ip_address</i> (three maximum) search=<i>domain_name</i>, <i>domain_name</i>, <i>domain_name</i>, <i>domain_name</i>, <i>domain_name</i>, <i>domain_name</i> (six maximum, the total length is less than or equal to 250 characters)}
network_interface, hostname, Internet Protocol (IP) address, netmask, DHCP, IPv6	network_interface=NONE, PRIMARY, or <i>value</i> {hostname=<i>hostname</i> ip_address=<i>ip</i> <i>address</i> netmask=<i>netmask</i> protocol_ipv6=yes/no} If DHCP <i>is</i> used, specify: {dhcp protocol_ipv6=yes_or_no} If DHCP is <i>not</i> used, specify: {hostname=<i>host_name</i> ip_address=<i>ip_address</i> netmask=<i>netmask</i> protocol_ipv6=yes_or_no}
root_password	root_password=<i>root_password</i> (Encrypted password from /etc/shadow)
security_policy	security_policy=kerberos, NONE

Table 13-1 Keywords and Arguments of the sysidcfg File (Continued)

Keyword	Argument
	Options for kerberos: {default_realm= <i>FQDN</i> admin_server= <i>FQDN</i> kdc= <i>FQDN1,FQDN2,FQDN3</i> } where FQDN is a fully qualified domain name.
	Note: You can list a maximum of three key distribution centers (KDCs), but at least one is required.
system_locale	system_locale= <i>locale</i> (Entry from /usr/lib/locale)
terminal	terminal= <i>terminal_type</i> (Entry from /usr/share/lib/terminfo database)
timezone	timezone= <i>timezone</i> (Entry from /usr/share/lib/zoneinfo/)
timeserver	timeserver=localhost, <i>hostname</i> , or <i>ip_addr</i>

Sample sysidcfg File

The following rules apply to the sysidcfg file:

- Keywords can be in any order.
- Keywords are not case sensitive.
- Keyword values can be optionally enclosed in single (') or double (") quotes.
- Only the first instance of a keyword is valid; if a keyword is specified more than once, the first keyword specified is used.

The following is an example of a `sysidcfg` file:

```
# Sample sysidcfg file for SPARC systems
system_locale=en_US
timezone=US/Mountain
timeserver=localhost
terminal=vt100
name_service=NONE
security_policy=NONE
root_password=Hx23475vABDDM
network_interface=PRIMARY {protocol_ipv6=yes netmask=255.255.255.0}
```

Locating the sysidcfg File

The location of the `sysidcfg` file (host and absolute directory path) is specified by the `-p` argument to the `add_install_client` shell script used to create JumpStart client information files. This is discussed in "Adding Install Clients" on page 13-5 and "Using `add_install_client`" on page 13-14.

As previously mentioned, you can use the `sysidcfg` file to answer system identification questions during the initial part of installation regardless of whether a name service (NIS or NIS+) is used. When this file is used with the NIS naming service, identification parameters, such as `locale` and `timezone` can be provided from the name service. The `sysidcfg` file necessary for installing a JumpStart client on a network running the NIS name service is typically shorter and a separate `sysidcfg` file for each client is unnecessary.

You can use the `/etc/locale`, `/etc/timezone`, `/etc/hosts`, `/etc/ethers` and `/etc/netmasks` files as the source for creating NIS databases to support client JumpStart installations. The following paragraphs provide a brief explanation of how each file, when created or modified and then converted to its respective database map, determines a specific identification parameter for the client installation process.

The /etc/locale File

To enable NIS support for a network installation of a JumpStart client, you must create the `/etc/locale` file if it does not exist (this assumes that the `system_locale` keyword is not provided in a `sysidcfg` file). When converted to its respective NIS map, `locale.byname`, it provides the installation program running on the JumpStart client with the default language information. If this information is not available, the client installation displays a dialog box and prompt for it.

The following is an example of the content found in the `/etc/locale` file on an NIS master for the `coastal` domain that sets the default language to English.

```
Central.Sun.COM          en_US
```

Note – You can also specify separate entries based on a host name rather than a domain. For a list of possible `locale` entries for this file, run the `'locale -c'` command.

Setting Up Locale

If the installation media contains multiple languages, you are prompted for the language to use during installation unless the installation process can determine the default localization.

On the NIS server, complete the following steps:

1. Make the following changes to the `/var/yp/Makefile` file:
 - a. Add the following text after the existing `audit.user.time` entry (approximately line 424):

```
locale.time: $(DIR)/locale
-@if [ -f $(DIR)/locale ]; then \
    sed -e "/^#/d" -e s/#.*$$// $(DIR)/locale \
    | awk '{for (i = 2; i<=NF; i++) print $$i, $$0}' \
    | $(MAKEDBM) - $(YPDBDIR)/$(DOM)/locale.byname; \
    touch locale.time; \
    echo "updated locale"; \
    if [ ! $(NOPUSH) ]; then \
        $(YPPUSH) locale.byname; \
        echo "pushed locale"; \
    else \
        : ; \
    fi \
else \
    echo "couldn't find $(DIR)/locale"; \
fi
```

The /etc/timezone File

To enable NIS support for a network installation of a JumpStart client, you must create the `/etc/timezone` file. If it does not exist (this assumes that the `timezone` keyword is not provided in a `sysidcfg` file). When converted to its respective NIS map, `timezone.byname`, it provides the installation program running on the JumpStart client with the default time zone information. If this information is not available, the client installation displays a dialog box and prompts for it.

The following is an example of the content found in the `/etc/timezone` file on an NIS master for the `Central.Sun.COM` domain that sets the default timezone to U.S. Mountain Standard Time:

```
US/Mountain      Central.Sun.COM
```

Note – You can also specify separate entries based on a host name rather than a domain. A list of possible locale entries for this file exists in the `/usr/share/lib/zoneinfo` directory.

The /etc/hosts File

To enable NIS support for a network installation of a JumpStart client, you must update the `/etc/hosts` file to include the client IP address and host name. When converted to its respective NIS map, `hosts`, it provides the installation program running on the JumpStart client with its IP address. Additionally, this file must have a `timehost` alias specified so the client can obtain the time of day information required for installation. Typically, this alias is assigned to the JumpStart server or the NIS master. If the client IP address information is not available, the client installation displays a dialog box and prompts for it.

The following is an example of the content found in the `/etc/hosts` file on a JumpStart server named `server1` for a client named `client1` with an IP address of `192.9.200.100` (includes the `timehost` alias assigned to the server):

```
192.9.200.1      server      timehost
192.9.200.100   client1
```

The /etc/netmasks File

To enable NIS support for a network installation of a JumpStart client, you must update the `/etc/netmasks` file to include the local network netmask value. When converted to its respective NIS map, `netmasks.byaddr`, it provides the installation program running on the JumpStart client with the local netmask value. If the client netmask information is not available, the client installation displays a dialog box and prompts for it.

The `/etc/netmasks` file contains network masks used to implement IP subnetting. It supports both standard subnetting as specified in Request for Change (RFC)-950 and variable length subnetting as specified in RFC-1519. When using standard subnetting, there should be a single line for each network that is submitted in this file with the network number, any number of SPACE or TAB characters, and the network mask to use on that network. You can specify network numbers and masks in the conventional IP '.' (dot) notation (such as IP host addresses, but with zeroes for the host part). For example, you can use:

```
192.9.200.0      255.255.255.0
```

to specify that the Class C network, `192.9.200.0`, should have eight bits of host field and twenty-four bits in the network field.

Note – See the man page for `netmasks` for more examples of subnetting.

Setting Up an Install Server

To enable a networked client to install the Solaris Operating Environment, the JumpStart install server must have the Solaris Operating Environment release software image available either on the local disk or from a CD-ROM shared to the network. The most common configuration for the JumpStart install server is to have this software available from the local disk. You use the `setup_install_server` script to accomplish this task. This script was previously described in "Adding a Bootable Image" on page 13-14.

The Solaris Operating Environment releases before release 8 (Solaris 7, 2.6, 2.5, and so on), had only one CD-ROM that contained the entire operating environment. The Solaris 8 Operating Environment has three; an installation CD-ROM, 1 of 2, and 2 of 2. To establish an installation server that contains the capability provided by the three CD-ROM set, you must make use of three different installation scripts:

To set up a install server that uses the Solaris 8 software image located on the local disk, perform following steps:

1. Ensure that the system has an empty directory (`/export/install`, for example) with approximately 700 Mbytes of available disk space
2. Insert the Solaris 8 Software CD-ROM 1 of 2 in the CD-ROM drive, allowing `vold` to automatically mount the CD-ROM
3. Change the directory to the location of the `setup_install_server` script

```
# cd /cdrom/cdrom0/s0/Solaris_8/Tools
```

4. Run the `setup_install_server` script to copy the release software from the CD-ROM to the local disk (this process takes about one hour):

```
# ./setup_install_server /export/install
```

There are two additional scripts that add functionality to the JumpStart boot or installation server; `add_to_install_server` and `modify_install_server`. For more information, see "Adding a Bootable Image" on page 13-14.

The add_to_install_server Script

The add_to_install_server script located on the Solaris 8 Software CD-ROM 2 of 2 enables the installation of supplemental CD-ROM products directories to an existing install server. If you do not use this script to install the additional Solaris Operating Environment release software located on CD-ROM 2 of 2, you will be limited to Core and EndUser software clusters.

To add the Solaris 8 Operating Environment supplemental software products to an existing install server, perform the following steps (this process takes about 15 minutes):

1. Insert the Solaris 8 Software CD-ROM 2 of 2 in the drive.

The vold daemon automatically mounts the CD-ROM.

2. Change the directory to the location of the add_to_install_server script.

```
# cd /cdrom/cdrom0/Solaris_8/Tools
```

3. Run the add_to_install_server script to install the additional software into the installation directory on the JumpStart server (assuming the location to be /export/install).

```
# ./add_to_install_server /export/install
```

The modify_install_server Script

The modify_install_server script located on the Solaris 8 Software Installation CD-ROM enables an interactive WebStart style of installation on the client.

Warning – Running the modify_install_server script actually defeats the purpose of the JumpStart program. It disables the non-interactive benefit of the JumpStart program. The resulting installation process will be *interactive*.

Setting up the Configuration Server

This section elaborates on the JumpStart configuration server setup. This system provides the configuration files for the JumpStart clients as previously discussed (see “Jumpstart Components: section).

The configuration directory minimally contains the following files:

- The rules file

The rules file classifies the machines on your network using a set of predefined keywords (included in Appendix C, “JumpStart Rules and Class File Keywords, Time Zones, and NIS+”). It also specifies the *class* file to be used by each class of machines.

- A *class* file for each category of machines you have determined on your network

The class files specify how the installation is to be done and what software is to be installed. The name of a class file is chosen by the system administrator and should follow UNIX file name conventions.

- The check script

You must run the check script after the rules and class files are created. It checks the syntax in the rules and class files. If there are no syntax errors, the check script creates the *rules.ok* file.

- The *rules.ok* file is created from the rules file by the check script. It is read during the automatic installation process (the rules file is not looked at).

- Optional begin and finish scripts.

- The begin and finish scripts are used to perform pre-installation and post-installation tasks. These scripts are available to perform more advanced customization of the installation process, such as answering the power management question that is asked when the newly-installed system first boots.

Setting Up a Configuration Server Directory

To set up a configuration directory, perform the following steps:

1. Select the system that will be the JumpStart configuration server and create the directory where you want to store the configuration information files. For the purpose of this discussion, use the `/export/config` directory as the name.
2. Mount the CD-ROM and copy the contents of the `/cdrom/sol_8_sparc/s0/Solaris_8/Misc/jumpstart_sample` directory located on the Solaris 8 Software CD 1 of 2 to your local `/export/config` directory. The `jumpstart_sample` directory from the CD-ROM contains template configuration files that you can customize; the `rules` file, several class files, a finish script, and the check script.
3. Share the configuration directory.
 - ▼ Add an entry to share the configuration directory to the network in the `/etc/dfs/dfstab` file. For example:

```
share -d "configuration directory" /export/config
```

- ▼ Execute the `/etc/nfs.server start` command.

Note – If the system is already an NFS server, you need to run only the `shareall` command.

4. Determine the different classes of machines that are or will be on your network and create the `/export/config/rules` file.

During the auto-install process, the install client is matched to a class in the `rules` file. Each class defined in the `rules` file has a specified file, called a *class* file, associated with it that is used to install the software.

5. Determine what installation parameters to use for each class (category) of machines you listed in step 4 and create an `/export/config/class` file for each (see “Creating Class Files” section).

The class file specifies how to partition the disk, what software clusters and packages to install, and what file systems to mount. (See the `host_class` template file in the configuration directory.)

6. Create `begin` and `finish` scripts. (This is optional.)

A `begin` script is run before the class file; that is, before the actual installation of software specified in the class file.

A `finish` script is run after the class file but before the system is rebooted. You can use it to modify the files or file systems of the newly installed system.

7. After configuration of the `rules` file, the class files, and the `begin` and `finish` scripts, run the `check` script. This script checks the `rules` and class files for correctness and basic syntax.

If no fatal errors are found, the `rules.ok` file is created from the `rules` file. It is the `rules.ok` file that is used by the client during the installation process.

- a. If the configuration server is running the Solaris 8 Operating Environment, run the following commands:

```
# cd /configuration_directory
# ./check
```

If the configuration server is not running the Solaris 8 Operating Environment, use the `-p` option to specify the path to the Solaris 8 distribution.

- b. Mount the Solaris 8 distribution CD-ROM on the configuration server (unless the configuration server is also the install server and you copied the distribution to the install server).
- c. Run the following commands:

```
# cd /configuration_directory
# ./check -p /path_to_Solaris_distribution
```


Creating the rules File

The rules file classifies the machines on your network. You should have a template of a rules file (an actual file called `rules`) in your configuration directory after you copy the `jumpstart_sample` directory to your configuration directory.

The rules file is read sequentially. As soon as the system finds a match in the rules file, it stops reading the file and continues with the JumpStart process. The fields are defined in Table 13-2 on page 13-30.

Syntax

```
[!] match_key match_value [&& [!] match_key match_value]* \  
begin class finish
```

The following table describes the fields in the `rules` file.

Table 13-2 Fields in the `rules` File

Field	Definition
<i>match_key</i>	A predefined keyword that describes an attribute of the system being installed. Examples of system attributes include physical memory size, disk sizes, kernel architecture, and so on. Keywords are used to help match a machine to a particular class for installation and are interpreted with respect to the install client.
<i>match_value</i>	The value (or range of values) selected by the system administrator for the <i>match_key</i> .
<i>begin</i>	The name of the <code>begin</code> script. A <code>-</code> is used in the <code>begin</code> field if no <code>begin</code> script is to be run during the automatic installation process.
<i>class</i>	The name of the <code>class</code> file. The names for the <code>class</code> files are chosen by the system administrator and must follow UNIX file name conventions.
<i>finish</i>	The name of the <code>finish</code> script (or a dash, <code>-</code>).

Using the && Symbols

You can use several keywords in a rule. They are joined together by the logical AND symbol, &&.

Using the ! Symbol

The logical NOT ! symbol, is used in front of a keyword to express negation. That is, to express that the install client's value for *match_key* does not equal the *match_value* specified in the rule.

Comments

You can use comments in the rules file.

A comment begins after a hash(#) sign. If a line starts with a #, then the entire line is a comment line. If a # is found in the middle of a line, everything after the # is considered a comment.

Note – Blank lines are also allowed in the rules file.

Available Keywords

Use the following keywords to classify the machines on your network. See Appendix A, "JumpStart Rules and Class File Keywords, Time Zones, and NIS+" for a detailed description of each keyword.

Table 13-3 Keywords

Keywords		
any	hostname	model
arch	installed	network
domainname	karch	totaldisk
disksize	memsize	

Examples of rules File Entries

The following is an example of the rules file entries.

```
# The following rule matches only one system, client1:
#
hostname    client1      -      host_class      set_root_pw

hostname    client2      -      class_basic_user      -

network     192.43.34.0  &&    ! model 'SUNW,Ultra-5_10' -      class_net3      -

model       'SUNW,Ultra-5_10' -      class_ultra      complete_ultra

memsize     64-96      &&    arch    sparc      -      class_prog_user      -

# The following rule matches any system.
any         -          -      class_generic      -
```

In this rules file example:

- The first rule matches a machine on a network called `client1`. The class file is `host_class` and the finish script is `set_root_pw`.
- The second rule matches a machine with host name `client2`. The class file is `class_basic_user`.
- The third rule matches a machine on network 192.43.34 that is not an Ultra 5 or 10. The class file is `class_net3`; there is no `begin` or `finish` script.
- The fourth rule matches a machine that is an Ultra 5 or 10. The class file is `class_ultra`, and there is a `finish` script called `complete_ultra`.
- The fifth rule matches a machine with memory between 64 and 96 Mbytes and a SPARC architecture. The class file is `class_prog_user`.
- The sixth rule matches any machine. The class file is `class_generic` and there is no `begin` or `finish` script.

Creating the Class Files

A class file, which is specified in a rule, determines how the installation is performed on the client and what software is installed. Unlike the `rules` file, class files do not have required names. However, just as for the `rules` file, there are predefined keywords that require certain parameters.

Keywords and Arguments

The following keywords and argument parameters are used in a class file to specify how the installation is to be done and what software to install. Refer to Appendix A, “JumpStart Rules and Class File Keywords, Time Zones, and NIS+” for a detailed description of each of the keywords and parameters listed in Table 13-4.

Table 13-4 Keywords and Argument Parameters for class Files

Keywords	Parameters
<code>install_type</code>	<code>initial_install</code> <code>upgrade</code>
<code>system_type</code>	<code>standalone</code> <code>dataless</code> <code>server</code>
<code>partitioning</code>	<code>default</code> <code>existing</code> <code>explicit</code>
<code>cluster</code> <code>cluster_name</code>	<code>add</code> <code>delete</code>
<code>package</code> <code>package_name</code>	<code>add</code> <code>delete</code>
<code>usedisk</code>	<code>disk_name</code>
<code>dontuse</code>	<code>disk_name</code>
<code>locale</code>	<code>locale_name</code>
<code>num_clients</code>	<code>number</code>
<code>client_swap</code>	<code>size</code>
<code>client_arch</code>	<code>kernel_architecture</code>
<code>filesystem</code>	<code>device</code> <code>size</code> <code>file_system</code> <code>optional_parameters</code>

Examples of Class Files

This section contains examples of class files.

Example 1

```
# Select software for programmers
install_type      initial_install
system_type       standalone
partitioning       default
filesys           any    100 swap # specify size of swap
filesys           server1:/usr/share/man - /usr/share/man ro,soft
cluster           SUNWCprog
package           SUNWman delete
package           SUNWypu add
package           SUNWypu add
```

This class file installs a system for programmers. The partitioning is determined by the software to be installed and the swap size is set to 100 Mbytes. The configuration cluster `SUNWCprog` contains packages for developing software in the Solaris 8 Operating Environment. The man pages from this cluster are deleted because they are mounted from `server1`, a server on the network. The NIS server packages, `SUNWypu` and `SUNWypu` are added.

The list of possible entries for the `cluster` keyword as it relates to the interactive installation names are shown in Table 13-5.

Table 13-5 Possible Entries for the `cluster` Keyword

Interactive Installation Name	Class File Name
Core	SUNWCreg
User	SUNWCuser
Developer	SUNWCprog
Entire Distribution	SUNWCall
Entire Distribution plus OEM	SUNWCXall

Example 2

install_type	initial_install		
system_type	standalone		
partitioning	explicit		
filesystem	c0t3d0s0	150	/
filesystem	c0t3d0s1	128	swap
filesystem	c0t3d0s6	800	/usr
filesystem	c0t3d0s7	free	/var
filesystem	c0t1d0s7	all	/opt
cluster	SUNWCall		
package	SUNWman	delete	

Note – This class file is intended for an end-user with a small disk who does not need the manual pages package, `SUNWman`.

Appendix A of the *Solaris 8 System Installation and Configuration Guide* contains a description of the clusters and packages available on the Solaris 8 software distribution CD-ROM.

Testing the Configuration with the `pfinstall` Command

The `pfinstall` command checks the *semantics* of your class files. It tests what happens during the automatic installation process, without actually performing an installation.

This command is successful only if the configuration and install server are the same system or the two systems are both running the same version of the Solaris 8 Operating Environment.

Running the `pfinstall` Command

To run the `pfinstall` command, perform the following steps:

1. If you have copied the entire CD-ROM Solaris 8 Operating Environment distribution to the local disk, run the `pfinstall` command (optional).

Syntax

```
# /usr/sbin/install.d/pfinstall -D | -d disk_file \
[-c path_to_distr] class_file_name
```

Options

- D Performs a dry run installation on the system disks using the class file *class_file_name*. It displays the resulting disk configuration and software selected, but no information is written to the disks.
- d Tests the *class_file_name* against the disk configuration described in the file *disk_file*. The *disk_file* file contains output from the running of the `prtvtoc(1M)` command on various disks. This gives you the ability to test your class file on various disk configurations.
- c Specifies the path to the Solaris 8 Operating Environment distribution

✓

This section presents three examples of the `pfinstall` command run to test the default class file `host_class` and includes some of the system output.

Example 1

Testing the class file, `host_class`, against the Solaris 8 Operating Environment installation image located on the CD-ROM:

```
# cd /export/config
# /usr/sbin/install.d/pfinstall -D -c /cdrom/cdrom0/s0 prog_class
Parsing profile
  0: install_type    initial_install
  1: locale          en_US
  2: system_type     standalone
  3: partitioning    default
  4: cluster         SUNWCuser
  5: cluster         SUNWCown    delete
  6: cluster         SUNWctltk   delete
  7: cluster         SUNWCxgl    delete
  8: cluster         SUNWCxil    delete
  9: filesystems     srvr:/usr/openwin - /usr/openwin ro,intr
```

Processing default locales

Processing profile

- Selecting cluster (SUNWCuser)
- Deselecting cluster (SUNWCown)
- Deselecting cluster (SUNWctltk)

<output truncated>

Example 2

Testing a disk file, 4BGdisk file and the host_class file, against the Solaris 8 installation image located on the CD-ROM:

Note – The 4BGdisk file is created from the output of the prtvtoc command run on a 4-Gbyte disk. This disk file can be used to create standard disk partitioning on JumpStart clients with 4-Gbyte disks.

```
# cd /export/config
# /usr/sbin/install.d/pfinstall -d 4BGdisk -c /cdrom/cdrom0/s0 prog_class
```

....<some output deleted> ...

Verifying disk configuration

Verifying space allocation

- Total software size: 399.62 Mbytes

Preparing system for Solaris install

Configuring disk (c0t0d0)

- Creating Solaris disk label (VTOC)

slice: 0 (	/)	tag: 0x2	flag: 0x0
slice: 1 (	swap)	tag: 0x3	flag: 0x1
slice: 2 (	overlap)	tag: 0x5	flag: 0x0
slice: 3 (	)	tag: 0x0	flag: 0x0
slice: 4 (	)	tag: 0x0	flag: 0x0
slice: 5 (	)	tag: 0x0	flag: 0x0
slice: 6 (	)	tag: 0x0	flag: 0x0
slice: 7 (	/export/home)	tag: 0x8	flag: 0x0

Creating and checking UFS file systems

- Creating / (c0t0d0s0)
- Creating /export/home (c0t0d0s7)

<output truncated>

Example 3

Testing an install image that has been copied from the CD-ROM
Solaris 8 software distribution to the local /export/install directory
against the host_class file:

```
# cd /export/config
# /usr/sbin/install.d/pfinstall -D -c /export/install host_class
```

....<some output deleted> ...

```
SUNWtleu....done.    1.81 Mbytes remaining.
SUNWnamdt...done.    1.80 Mbytes remaining.
      SUNWnamos...done.    1.60 Mbytes remaining.
SUNWnamow...done.    1.52 Mbytes remaining.
```

Completed software installation

Solaris 8 software installation succeeded
Solaris 8 packages fully installed

```
      SUNWxwrtx
      SUNWxwrtl
      SUNWwsr
      SUNWwbapi
```

<some output omitted>

Customizing system devices

- Physical devices (/devices)
- Logical devices (/dev)

Installing boot information

- Installing boot blocks (c0t0d0s0)

Installation log location

- /a/var/sadm/system/logs/install_log (before reboot)
- /var/sadm/system/logs/install_log (after reboot)

Mounting remaining file systems

- Mounting /a/export/home (/dev/dsk/c0t0d0s7)

Installation complete

Test run complete. Exit status 0.

Using add_install_client

Use `add_install_client` and `rm_install_client` to add or remove clients to the install server or boot server or both you have set up to support the JumpStart installation (these commands update the `/etc/bootparams` file).

Running the add_install_client Script

The `add_install_client` command must be run from the install server's Solaris installation image (a mounted Solaris Operating Environment CD-ROM or a Solaris Operating Environment CD-ROM copied to disk) or the boot server's boot directory (if a boot server is configured). The Solaris installation image or the boot directory must be the same Solaris Operating Environment release that you want installed on the client.

Syntax

```
# ./add_install_client -e ethernet_addr -i ip_addr \  
-s install_svr:/distr -c config_svr:/config_dir \  
-p sysid_config_svr:/sysid_config_dir client_name client_arch
```

- e Specifies the Ethernet address of the install client and is necessary if the client is not defined in the name service.
- i Specifies the IP address of the install client and is necessary if the client is not defined in the name service.
- s Specifies the name of the install server and the path to the Solaris 8 Operating Environment distribution. This option is necessary if the client is being added to a boot server.
- c Specifies the configuration server and the path to the configuration directory.
- p Specifies the configuration server and the path to the `sysidcfg` file. This option is available on Solaris 8 Operating Environment and later distributions.

You can apply the following associations to the examples of the `add_install_client` command arguments:

Install server	<i>install_svr</i>
Distribution	copied to /export/install
Configuration server	<i>config_svr</i>
Configuration directory	/export/config
Boot server	<i>boot_svr</i>
Install client	<i>client_name</i>
Client architecture	<i>client_arch</i>

Adding a Client Using a Solaris CD-ROM Image on the Local Disk

To create a JumpStart client from a server that has the Solaris 8 software copied to the local disk (see "Setting Up an Install Server" on page 13-24), perform the following steps:

1. Change the directory to the location of the installed Solaris 8 Operating Environment image:

```
# cd /export/install/Solaris_8/Tools
```

2. Create the JumpStart client using the `add_install_client` script found in the local directory. The following command creates a sun4u architecture client named `client1` using `server1` as its install and configuration server:

```
# ./add_install_client -s server1:/export/install \  
-c server1:/export/config \  
-p server1:/export/config client1 sun4u
```

Note – The location of the Solaris 8 software installation files (`-s` option) in the command indicate the `/export/install` directory. The location of the JumpStart configuration files (`-c` option) on `server1` in the previous command indicate the `/export/config` directory. Discussion of the JumpStart configuration files is subsequent to this section.

Adding a Client Using a Solaris CD-ROM Image From the CD-ROM

To create a JumpStart client from a server that does *not* have the Solaris 8 software CD-ROM Image copied to the disk, perform the following steps:

1. Insert the Solaris 8 Software Installation CD-ROM in the drive.

The `vold` daemon automatically mounts the CD-ROM.

2. Change the directory to the location of the `add_install_client` script on the Solaris 8 CD-ROM 1 of 2:

```
# cd /cdrom/cdrom0/s0/Solaris_8/Tools
```

3. Create the JumpStart client using the `add_install_client` script found in the that directory. The following command created a `sun4u` architecture client named `client1` using the Solaris 8 software CD-ROM for its installation and a local directory (`/export/config`) on `server1` for the `sysidcfg` and other configuration files:

```
# ./add_install_client -c server1:/export/config \  
-p server1:/export/config client1 sun4u
```

Note – Additionally, running the previous `add_install_client` command creates an entry in the `/etc/dfs/dfstab` file to share the `/cdrom` directory to the network for mount by the JumpStart client. Installation software is obtained from the media.

The /etc/bootparams File Content

The `/etc/bootparams` file is updated each time the `add_install_client` script is run. The resulting content provides the server name(s) and the directory locations for the installation and configuration files.

Content With Locally Available Installation Files

A server named `server1` with Solaris 8 software files copied to the local disk (see "Adding a Client Using a Solaris CD-ROM Image on the Local Disk" on page 13-41) and shared to the network has a `client1 /etc/bootparams` entry as follows:

```
client1 root=server1:/export/install/Solaris_8/Tools/Boot
install=server1:/export/install boottype=:in
sysid_config=server1:/export/config install_config=server1:/export/config
rootopts=:rsize=32768
```

Content from the CD-ROM Installation Files

A server named `server1` with Solaris 8 software files shared to the network from the `/cdrom` directory (see "Adding a Client Using a Solaris CD-ROM Image From the CD-ROM" on page 13-42) has a `client1 /etc/bootparams` entry as follows:

```
client1 root=server1:/cdrom/sol_8_sparc/s0/Solaris_8/Tools/Boot
install=server1:/cdrom/sol_8_sparc/s0 boottype=:in
sysid_config=server1:/export/config install_config=server1:/export/config
rootopts=:rsize=32768
```

The /etc/dfs/dfstab File Content

The `/etc/dfs/dfstab` file is populated with the appropriate entry for either the local file system or the CD-ROM, depending on the directory location from which the `add_install_client` command is run. If the location of the shell was `/cdrom/cdrom0/s0/Solaris_8/Tools`, the entry in the `/etc/dfs/dfstab` would be updated with a `share` command for the CD-ROM. Conversely, if the location of the shell was `/export/install/Solaris_8/Tools` (or some other arbitrary installation directory location), the `/etc/dfs/dfstab` would be updated with a `share` command for that local directory.

Referencing the Solaris Software from Locally Available Installation Files

The following `/etc/dfs/dfstab` file entry reflects the shared directory (`/export/install`) of installation files from a local disk:

```
share -F nfs -o ro,anon=0 /export/install
```

Referencing the Solaris Software from CD_ROM Installation Files

The following `/etc/dfs/dfstab` file entry reflects the shared directory of installation files from the CD-ROM:

```
share -F nfs -o ro,anon=0 /cdrom/sol_8_sparc/s0
```

Note – All directory entries listed in the `/etc/bootparams` *must* be shared file systems.

Initiating a JumpStart Installation

Once you have created the boot, install, and configuration services for the local network, you must boot the JumpStart client.

Boot Install Clients

You can use one of the following methods to boot a JumpStart client.

- New machines

Turn on the machine.

- Existing machines

Issue the following command:

ok boot net - install

Note – If your local network is running the NIS name services, you *must* update the appropriate maps to support the JumpStart installation, otherwise, the client boot process will be unsuccessful.

JumpStart Capabilities and Limitations

Table 13-6 compares the capabilities and limitation of the JumpStart program.

Table 13-6 JumpStart Capabilities and Limitations

	Capabilities	Limitations
Solaris Operating Environment software	Installs the Solaris Operating Environment on new systems. Upgrades existing Solaris 2.1 Operating Environment and later systems to later releases.	Default partition sizes for previous Solaris Operating Environment releases can prevent the use of the JumpStart upgrade to later releases.
Site-specific customizations	Allows for site-specific profiles and scripts to be added for flexibility.	None.
Sun unbundled products	Have no specific support in the Solaris 8 Operating Environment.	Use with Sun unbundled products is not prohibited with the Solaris 8 Operating Environment JumpStart product, but you must write product-specific scripts to install the software packages in the product. You also have to consider disk partitioning and software licensing for the product.
Third-party products	Have no specific support in the Solaris 8 Operating Environment.	Use with third-party products is not prohibited with the Solaris 8 Operating Environment JumpStart product, but you must write product-specific scripts to install the software packages in the product. You also must consider disk partitioning and software licensing for the product.

Table 13-6 JumpStart Capabilities and Limitations (Continued)

	Capabilities	Limitations
Patches	Can be applied automatically.	None.
Configurations supported	Installs: Servers Standalone systems Dataless systems	Does not install diskless clients.
Operations handled by default	Partitions disks and sizes of file systems. Assigns host name and name service domain (NIS+, NIS, or local files).	Requires more administrative steps; for example, the script <code>set_root_pw</code> , which is located in the <code>jumpstart_sample</code> directory is the script needed to assign root password. Requires IP addresses to be manually allocated by the administrator.
Operation handled by additional scripts	Enables arbitrary site-specific customization, such as: setting up a second Ethernet port on a machine and making it a router; adding additional non-root users to a local system, setting up print servers; adding known print servers to a print client; and adding entries to the automount map.	None.
Network configuration/routers		Requires a boot server on the local network or subnet. Allows install servers to be placed on the opposite side of a router.

Table 13-6 JumpStart Capabilities and Limitations (Continued)

	Capabilities	Limitations
Concurrent use/batch mode	Has no software limitation on the number of clients that can be installed concurrently.	Has physical limitations that include: The install server is more responsive if its copy of the Solaris Operating Environment distribution is on disk, rather than on the CD-ROM device, which is slow for random accesses. The number of clients concurrently doing an installation also negatively affects performance.

Worksheet for Configuring for JumpStart Installation Exercise

Install server name: _____

Timehost server name: _____

Note – Without an assigned `timehost` entry for one of the Solaris Operating Environments, the JumpStart process becomes interactive, prompting you for the timehost information. The NIS master is a good candidate for this exercise.

Solaris Operating Environment distribution location at:

Configuration server name: _____

Configuration directory: _____

Boot server name: _____

Directory containing client support: _____

Install client's name: _____

Install client's IP address: _____

Install client's Ethernet address: _____

Install client's architecture: _____

Exercise: Configuring for a JumpStart Installation



Exercise objective – In this lab, you configure a JumpStart server to support one install client. The install client loads the Solaris Operating Environment from the installation CD-ROM.

Preparation

Locate the Solaris 8 Software CD-ROM 1 of 2. Use the NIS master as the JumpStart server, and the former NIS client as the install client. The JumpStart process completely replaces the content of the client's disk. Verify that an entry for the client system exists in the `/etc/hosts` file on the server. Determine the Ethernet (MAC) addresses of the server and the client systems.

Refer to your lecture notes as necessary to perform the steps listed.

Tasks

Complete the following steps:

1. Log in as the user root.
2. Make the installation and configuration directories that will hold the Solaris 8 software and JumpStart configuration files, respectively.

```
# mkdir /export/install /export/config
```

3. Insert the Solaris 8 software CD-ROM 1 of 2 in the drive and run the `setup_install_server` script to copy the release software from the CD-ROM to the local disk (this process takes about one hour):

```
# cd /cdrom/cdrom0/s0/Solaris_8/Tools
# ./setup_install_server /export/install
```

4. Insert the Solaris 8 Software CD-ROM 2 of 2 in the drive and change directory to the location of the `add_to_install_server` script.

```
# cd /cdrom/cdrom0/sol_8_sparc_2/Solaris_8/Tools
```

5. Run the `add_to_install_server` script to install the additional software into the installation directory on the JumpStart server (assuming the location to be `/export/install`).

```
# ./add_to_install_server /export/install
```

6. Copy the `jumpstart_sample` directory from the local `/export/install/Solaris_8/Misc` directory to the `/export/config` directory.

```
# cd /export/install/Solaris_8/Misc/jumpstart_sample
# cp -r * /export/config
```

The `/export/config` directory now contains the files that you edit to create your specific JumpStart configuration.

7. Edit the `/etc/dfs/dfstab` file to add an entry for the `/export/config` directory as follows:

```
share -F nfs -o ro,anon=0 /export/config
```

8. If the NFS server daemons are not running, start them.

```
# /etc/init.d/nfs.server start
```

If the NFS server daemons were already running, run the `shareall` command.

```
# shareall
```

9. Edit the `/export/config/rules` file and modify an existing entry for a JumpStart client. Change the line that reads

```
hostname sample_host - host_class set_root_pw
```

so that it reflects the name of your JumpStart client. Replace `sample_host` with the name of your JumpStart client, and replace the `set_root_pw` finish script name with `finish_script` (your instructor will provide this script):

```
hostname client1 - host_class finish_script
```

10. Edit the `/export/config/host_class` file so that it contains the following:

```
install_type    initial_install  
system_type    standalone  
partitioning   default  
cluster        SUNWCall  
filesys        any 128 swap
```

11. Edit the `/export/config/finish_script` file so that it contains the following:

```
${SI_CONFIG_DIR}/auto_shut_down  
${SI_CONFIG_DIR}/setup_root_pw
```

The `auto_shut_down` script will be used to automatically answer no to the power management questions. The `set_root_pw` script will be used to set the root user password.

12. Edit the `/export/config/auto_shut_down` file so that it contains the following:

```
touch /a/noautoshtutdown
```


13. Edit the `/export/config/set_root_pw` file. Copy and paste the root user encrypted password from the shadow file to the `"PASSWD="` variable.

PASSWD=encrypted_passwd

14. Run the `/export/config/check` program and correct any problems in the rules or `host_class` files that it reports. Verify that the `rules.ok` file exists once check completes successfully.

./check

15. Edit the `/etc/hosts` file and add an entry for the JumpStart client and the `timehost` alias in the JumpStart server's entry.

```
#
# Internet host table
#
127.0.0.1      localhost
192.9.200.1    server1  loghost    timehost
192.9.200.100 client1
```

Ensure that there is a `timehost` entry on the same line with the JumpStart server.

16. Edit the `/etc/ethers` file and add an entry for the JumpStart client; for example:

```
8:0:20:2f:90:3d    client1
```

17. Edit the `/etc/timezone` file and add an entry for the appropriate time zone for the JumpStart client. Entries in this file are case sensitive.

```
US/Mountain      domainname
```

18. Update the NIS maps by running the `make` command.

```
# cd /var/yp
# /usr/ccs/bin/make
```

19. Use the `add_install_client` program to add support for your JumpStart client. The following command example is appropriate for a server that will provide access to the operating environment using a mounted Solaris 8 CD-ROM. Replace *server* with the name of your JumpStart server, *client* with the name of your JumpStart client, and *sun4x* with either *sun4u*, *sun4m*, or *sun4c*, depending on the type of client system you are using.

```
# cd /export/install/Solaris_8/Tools
# ./add_install_client -c config_server:/export/config \
-s install_server:/export/install -p sysidcfg_server:/export/config \
client1 sun4x
```

The `add_install_client` program updates the `/etc/bootparams` file with information for the client you name, and updates the list of directories shared in the `/etc/dfs/dfstab` file to include the `/export/install` directory.

20. Update the NIS maps by running the `make` command.

```
# cd /var/yp
# /usr/ccs/bin/make
```

21. Create the `sysidcfg` file.

```
# cd /export/config
# vi sysidcfg
```

Add the following lines to the file:

```
security_policy=NONE
network_interface=PRIMARY {protocol_ipv6=yes netmask=255.255.255.0}
```

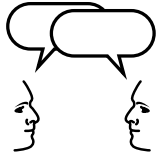
22. Test your setup:

```
# /usr/sbin/install.d/pfinstall -D \
-c /export/install /export/config/host_class
```

23. Boot the JumpStart client:

```
ok boot net - install
```

Exercise Summary



Discussion – Take a few minutes to discuss the experiences, issues, or discoveries that you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ Define the JumpStart program
- ☐ List the main components for setting up a JumpStart network installation
- ☐ Briefly describe the `add_install_client` script syntax
- ☐ Set up the boot services function on a subnet using the `setup_install_server` script
- ☐ Describe the JumpStart client boot sequence
- ☐ List the files necessary to support the JumpStart boot operation
- ☐ Describe the syntax of the `add_install_client` script as it relates to the boot server
- ☐ Describe the use of the `sysidcfg` file with and without name service support
- ☐ Set up a JumpStart installation server system to provide the Solaris 8 Operating Environment with the software necessary to install a new system
- ☐ Describe the use of the `add_to_install_server` and `modify_install_server` scripts
- ☐ Add install clients to install servers and boot servers
- ☐ Create a configuration server with a customized rules file and class files
- ☐ Use the `pfinstall` command to test configuration and installation files
- ☐ Boot install clients
- ☐ Configure NIS name service support for the JumpStart program

Objectives

Upon completion of this module, you should be able to:

- Install the Solaris 8 Operating Environment using both the WebStart and the interactive methods
- Partition a disk to a given format
- Build new file systems and mount the file systems
- Configure an automount
- Set up an NIS domain
- Create additional swap space
- Set up an NFS mount
- Configure RBAC to allow a `non-root` user to shut down a system
- Create a local and a network printer

Solaris Operating Environment Administration Workshop

This workshop requires systems configured with two disks, of at least 2 Gbytes each.

You also need an external tape drive. Turn off the power to the external tape drive before installing the Solaris 8 Operating Environment.

Identify groups of three systems. The configuration goals in the following section refer to systems by number (1, 2, and 3). Form groups of students to configure each group of three systems. Determine the host name and IP address for each system.

Detach any servers from the network that could supply identification information to systems that are loading the Solaris Operating Environment, or that could interfere with the JumpStart process later in the exercise.

Set all OBP parameters to their default values before starting to configure the systems.

Configuration Goals

Configure the systems according to the following configuration goals and verify that each group accomplishes each goal.

- System 1 provides the following services:
NIS slave, JumpStart server, man page server, work area server
- System 2 provides the following services:
NIS master, home directory server, application server
- System 3 provides the following services:
NIS client, JumpStart client

To configure the systems, complete the following, steps:

Use the WebStart installation method to install system 1 and 2. Use the Interactive installation method to install system 3.

1. Configure systems 1 and 2 as follows:

a. Configure the swap area.

Use the default placement of the swap area and set the size to the minimum size listed.

b. Complete the system information.

Use the host name, IP address, and netmask information provided by your instructor.

Do not select a naming service. Specify the time zone and time information appropriate for your location.

Use `cangetin` for the root password.

c. Complete the configuration information.

Disable power management. Select a locale appropriate for your location. Do not install additional products.

d. Complete the file system layout.

Configure only one disk, as follows:

Slice	Use	Size
0	/	200
1	swap	(fixed)
5	/opt	300
6	/usr	(remainder)
7	/export/home	100

2. Configure system 3 as follows:

a. Complete the identification information.

Use host name, IP address, and netmask information provided by your instructor.

Use `cangetin` for the `root` password.

b. Complete the configuration information.

Disable power management. Select a locale appropriate for your location.

Install the entire Solaris 8 Operating Environment distribution.

c. Complete the file system layout.

Configure only one disk, as follows:

Slice	Use	Size
0	/	300
1	swap	128
6	/usr	(remainder)

Note – Finish installing all three systems before proceeding.

3. On system 1, divide an unused disk into three partitions where:

- ▼ One has enough space to hold the entire Solaris 8 software release (from the 1 of 2 and 2 of 2 CD-ROMs)
- ▼ The other two use at least 500 Mbytes each.

4. Create file systems and make the changes required to cause:

- ▼ The larger file system to mount automatically as `/export/install`
- ▼ The 500-Mbyte file systems to mount automatically as `/data1` and `/data2`.

5. On system 1, load the Solaris 8 software into `/export/install` to support JumpStart operations. Load the software from both the 1 of 2 and 2 of 2 CD-ROMs.
6. On systems 2 and 3, identify and remove all of the man page packages.
7. On system 1, share the man pages that the other systems are now missing. Make this a read-only share.
8. On all systems, modify the template initialization file for the Bourne and Korn shells so they include the appropriate definitions of the following environment variables.

EDITOR
LPDEST
ENV

Note – Use a printer name that matches the printer you intend to set up. Your instructor can supply values as required.

9. On all systems, create an initialization file for the user `root` so it sets the same variables listed in the previous step. Which of these variables is not appropriate for the current `root` shell?

10. Add users as follows:
 - a. On system 2, create five users called `user1` through `user5`.
 - b. On system 1, create five users called `user6` through `user10`.
 - c. On system 3, create five users called `user11` through `user15`.
 - d. Assign a unique UID number to each user.
 - e. Use `/export/home` as the base path for home directories.
 - f. Use the Korn shell for all users.
 - g. Verify that the correct initialization files for these users are in place.
11. On all systems, make the changes required for each system to communicate with the others using the network.
12. Set up all systems to allow remote `root` logins.
13. On systems 2 and 3, add a direct map to the automount service so these systems automatically mount the man pages from system 1.
14. On system 2, make the changes required to share the home directories of `user1` through `user5`.
 - ▼ Modify the entries in `/etc/passwd` for `user1` through `user5` so they all use home directories below `/home` instead of `/export/home`.
 - ▼ Make changes to the existing indirect map in the automount service to support mounting home directories for `user1` through `user5`.
 - ▼ Attempt to use just one entry in the indirect map to accomplish this task.

15. Set up NIS as follows:
 - a. Establish system 2 as an NIS master server. Establish system 1 as an NIS slave server.
 - b. Establish system 3 as an NIS client.
 - c. Make certain the following files in `/etc` are brought under NIS control:
 - `bootparams`
 - `ethers`
 - `locale`
 - `netmasks`
 - `timezone`
 - `netgroup`.
16. Update system 2 to include `/etc/auto_direct` in NIS.
17. Set up system 2 to treat system 1 and 3 as trusted hosts for remote `root` operations.
 - a. Copy the `home` directories found on system 1 for `user6` through `user10` to the corresponding location on system 2.
 - b. Copy the `home` directories found on system 3 for `user11` through `user15` to the corresponding location on system 2.
 - c. Recursively remove these users' `home` directories from system 1 and system 3.

18. On system 2, duplicate the users you originally created on system 1 and 3, but change their home directories to fall below /home instead of /export/home.
 - a. Update NIS
 - b. Remove those user definitions from system 1 and 3
 - c. Verify that the user logins work on all systems
19. Change the host name of system 3 in all files where it is specified. Update the NIS master accordingly.
20. Create a directory called /data1/workgroup on system 1.
 - a. Create a new group called class1 and associate user1 through user5 with class1 as a secondary group.
 - b. Change permissions for /data1/workgroup so all files created within it are owned by the class1 group.
 - c. Make the changes required to have system 2 and 3 mount /data1/workgroup from system 1 when they boot.
21. Configure all systems with attached external tape drives to make use of those devices.
22. On all systems, create a new device alias at the OBP level and use it as the default boot device.
23. Establish system 2 as the log host.
 - a. Enable inetd tracing on all systems.
 - b. Have systems 1 and 3 pass inetd tracing messages to the log host.
 - c. On the log host, save inetd tracing messages to a file called /var/log/inetlog.

24. Use a script, provided by your instructor, to announce all run-level changes.
 - a. Edit this script to use the `logger` command to send run-level change messages to the `syslog` utility.
 - b. Modify `/etc/syslog.conf` to log these messages into a file called `/var/log/runlevel`.
25. On system 2 and 3, divide a spare disk into four equal partitions.

Note – These partitions must be identical in size.

- a. Create two file systems that use default characteristics from `newfs`.
 - b. Create one file system with an inode ratio of 1 inode per 16384 bytes of data space.
 - c. Create one file system that uses a 1 percent `minfree` value.
 - d. Compare and describe the differences in the values reported by `df -k` for these file systems.
 - e. Permanently mount one of these file systems as `/morespace`.
26. On systems 2 and 3, create a 30 Megabyte swap file in `/morespace`. Make the changes required to permanently add this file to the swap area used on these systems.

27. On all systems, use `coreadm` to enable saving global core files.
 - a. Set the global core file path to `/corefiles/core.nodename.filename.PID`.
 - b. Run a `csch` process on each system.
 - c. Send the `SIGABRT` signal to each `csch` processes to generate core files.
 - d. For each `csch` process, verify the presence of a core file in the current directory and in `/corefiles`.
28. On systems 2 and 3, change the NFS mounts for the man pages to `cachefs` mounts. Make these changes permanent.
29. On system 3, configure RBAC to allow a non-`root` user to shut down the system.
30. Create a local and network printer on systems 1 and 2.
31. Configure system 1 as a JumpStart server to load the Solaris Operating Environment on system 3.
 - a. Use the memory size parameter as the identifying class criteria.
 - b. Specify that the client should load the core Solaris Operating Environment installation level.
 - c. Specify a disk configuration for system 3 that places the operating environment in one partition.
 - d. Use the JumpStart program on system 3.

-
32. Back up the root file system on the JumpStart server.
 - a. Destroy the JumpStart program's root file system.
 - b. Restore the JumpStart program's root file system.
 - c. Verify that system 3 can still use the JumpStart services from this server.

Check Your Progress

Before continuing on to the next module, check that you are able to accomplish or answer the following:

- ☐ Install the Solaris 8 Operating Environment using both the WebStart and the interactive methods
- ☐ Partition a disk to a given format
- ☐ Build new file systems and mount the file systems
- ☐ Configure an automount
- ☐ Set up an NIS domain
- ☐ Create additional swap space
- ☐ Set up an NFS mount
- ☐ Configure RBAC to allow a non-root user to shut down a system
- ☐ Create a local and a network printer

The JumpStart *rules* and Class Files



Table A-1 describes the keywords for the `rules` file.

Table A-1 Keywords for the `rules` File

Keyword	Description
<code>any</code>	The match always succeeds. The <i>match_value</i> field is ignored. Use a dash (-) as a placeholder for the <i>match_value</i> .
<code>arch</code>	The machine's processor type as reported by the <code>/usr/ucb/mach</code> command or by <code>uname -p</code> in the Solaris 8 Operating Environment.
<code>domainname</code>	The domain name of the network in which the machine has been configured as reported by the <code>domainname</code> command. The domain name is used by a client to request and obtain name service information.
<code>disksize</code>	This keyword is used with two parameters for the <i>match_value</i> field:
<code>device</code>	Any disk device name, such as <code>c0t3d0</code> or the special word <code>rootdisk</code> .
<code>size_range</code>	The size of the disk specified in Mbytes. For example, <code>disksize c0t3d0 250-300 . . .</code> , which matches a target 3 disk that is between 250 and 300 Mbytes in size.
<code>hostname</code>	The host name assigned to the machine as reported by the <code>uname -n</code> command.

Table A-1 Keywords for the rules File (Continued)

Keyword	Description
installed	This keyword uses two parameters for the <i>match_value</i> field: • <i>disk</i> – The value of the <i>disk</i> parameter can be any, bootdisk, or a disk slice name (such as c0t3d0s5). If any is used, the rule is applied to any disk attached to the system. If rootdisk is used, only the disk where the root file system resides is considered. • <i>version</i> – The version parameter can be any, upgrade, or a version name (such as Solaris_2.2). If any is used, any Solaris Operating Environment or SunOS release is considered. If upgrade is used, any upgradable Solaris 2.1 Operating Environment or greater release is matched.
karch	The machine's kernel architecture, as reported in the output of /usr/ucb/arch -k or the uname -m command.
memsize	The amount of physical memory on the machine (in Mbytes). This is the third field of the line beginning with Memory in the output from the prtconf command. A range must be specified for the <i>match_value</i> field for this keyword. For example, a <i>match_value</i> field value of 16–32 specifies a physical memory size of 16–32 Mbytes of memory.
model	The model number of the machine. This is system dependent and varies by manufacturer. Samples can be seen on line 5 of the output from the prtconf command. Table A-2 shows the model numbers for various machines. If the model name includes spaces, enclose the entire model name in quotes. For example, specify a Sun 4/110 as 'Sun 4_100 Series'.
network	The network number of the IP address assigned to the machine which is determined after the subnet mask has been applied. (The network number always ends with a 0.)

Table A-1 Keywords for the rules File (Continued)

Keyword	Description
totaldisk	The total amount of disk space on the machine (in Mbytes). Specify a range for the <i>match_value</i> field for this keyword.

Table A-2 describes the model names.

Table A-2 Model Names

Machine	Model
4/110	Sun-4 TM _100 Series
4/200	Sun-4_200 Series
ss1	Sun-4_60
ss1+	Sun-4_65
slc TM	Sun-4_20
IPC TM	SUNW,Sun 4_40
ELC TM	SUNW,Sun 4_25
IPC	SUNW,Sun 4_50
ss2	SUNW,Sun 4_75
4/300	Sun SPARCsystem 300
4/400	Sun SPARCsystem 400
4/600	SUNW,SPARCsystem-600
ss10	SUNW,SPARCstation-10
classic	SUNW,SPARCclassic
lx	SUNW,SPARCstation-LX
SC1000	SUNW,SPARCserver-1000
SC2000	SUNW,SPARCcenter-2000

Initial Install Keywords and Arguments

Class files contain the following keywords and parameters:

`install_type initial_install`

This keyword specifies the type of installation the class file describes. The values `initial_install` and `upgrade` are the values supported in the Solaris 2.2 Operating Environment. Every class file must contain this keyword; otherwise, the check script exits with a fatal error and the `rules.ok` file is not created.

`system_type (standalone | dataless | server)`

This keyword defines the type of system being installed. If this keyword is not used in the class file, the system type defaults to `standalone`.

`partitioning (default | existing | explicit)`

The keyword `partitioning` defines how the disk is partitioned during installation. If this keyword is not specified in a class file, the partitioning type `default` is used.

`default`

This keyword indicates that the installation process partitions the disk based on the software selected. This means that the installation process selects the disks and slices on which to install the selected packages. The rootdisk is selected first. However, if the selected packages do not fit on this disk, an additional disk is selected. Pre-existing data on a disk can be overwritten unless it is preserved. Refer to Appendix C of the *Solaris 2.2 System Configuration and Installation Guide* for information on backing up data, or use the `filesys` keyword to preserve partitions.

existing

This keyword indicates that the existing partitions and file systems on the disk should be used; no repartitioning is done. It also implies that all file systems except `/`, `/usr`, and `/var` are preserved. The disk is not repartitioned.

When the `existing` keyword is specified, the last mount point field from the file system superblock is used to determine which file system mount point the partition represents.

If the system has duplicate file systems (for example, two `/usr`, two `/`, and so on), the installation aborts. Use the `filesys` keyword (see page C-10) with the argument `existing` to specify which file systems to preserve.

explicit

This keyword configures only the file systems specified with the `filesys` keyword.

cluster *cluster_name* (add | delete)

This keyword is used to specify the cluster (or configuration cluster) to be installed. There are four configuration clusters included in the Solaris 2.2 distribution: `SUNWCall`, `SUNWCprog`, `SUNWCuser`, and `SUNWCreq`. In addition to the configuration clusters, there are also various clusters, which are defined groups of packages. See Appendix A of the *Solaris 2.2 System Configuration and Installation Guide* for more details on clusters and packages.

You can specify only one configuration cluster. If the cluster specified is one of those listed previously, the third field (add or delete) is ignored. Otherwise, the third field indicates whether the cluster should be added or removed from the previously specified configuration cluster. If no configuration cluster is specified, the end-user configuration cluster (SUNWuser) is used.

`package` *package_name* (add | delete)

This keyword defines which package is to be installed. The third field specifies whether the package is added or deleted from the configuration cluster and clusters previously specified. If the third field is not specified, add is assumed.

Your class files can contain any number of `package` keywords.

`usedisk` *disk_name*

This keyword specifies which disk is to be used with default partitioning. The installation process then uses only the disks specified by `usedisk`. The *disk_name* is given in the form *cxydz*; for example, c0t2d0.

Your class files can contain any number of `usedisk` keywords.

`dontuse` *disk_name*

This keyword explicitly specifies that *disk_name* is not to be used by the installation process.

Your class files can contain any number of `dontuse` keywords.

`locale` *locale_name*

Install all localization packages associated with the selected software for the *locale_name* specified; for example, usage for the French locale is `locale fr`.

`num_clients` *number*

This keyword indicates the number of diskless clients that the server supports. (The default number of clients is five.) The installation process then allocates enough space for each diskless client's root and swap file systems.

This keyword can only be used when `system_type` is `server` (see page C-6).

`client_swap` *size*

This keyword indicates the amount of swap space allocated for each diskless client (the default size is 24 Mbytes). Use this keyword only when `system_type` is `server`.

`client_arch` *kernel_architecture*

This keyword specifies that the client has a different kernel architecture than the server. It tells the installation process to install packages on the server to support this client architecture. The default is to install client support based on the assumption that the client has the same architecture as the server.

Use this keyword only if `system_type` is `server`. You can specify multiple `client_arch` keywords in your class files.

`filesys` *device size file_system optional_parameters*

This keyword is used to specify a size (in Mbytes) and device for the file system indicated. You can use any number of `filesys` keywords in your class files.

device

This keyword is specified as one of the following:

any – Use any disk that meets the install software's criteria.

device_name (*ctxdxsx*) – Explicitly state where the file system is to be placed.

rootdisk.sn – Specify the logical name for the disk where the root file system is to reside. The *sn* suffix indicates a specific slice on the disk.

server:path – A remote file system. Remote file systems are not changed by the installation process, but are set up so that the newly installed machine automatically mounts them when it boots.

Using *server:path* with the *filesys* keyword adds NFS mount entries to the */etc/vfstab* file of the newly installed system.

size

This keyword is specified as one of the following:

num – Allocate *num* Mbytes for the file system during installation.

existing – Use the current size of the partition. If you use *existing* as the value of the *size* field, the *device* field must specify a real device (*device_name* or *rootdisk.sn*).

auto – Automatically size the partition based on the software selected.

all – Use the entire disk for the file system.

free – Use the remaining space on the disk for the file system. If *free* is used as an argument to *filesys*, make the *filesys* entry the last entry in your class file for the specified disk.

start:size – Explicitly partition the device. *start* is the cylinder where the slice begins; *size* is the number of cylinders for the slice. If you use *start:size* for the size argument to the *filesys* keyword, the *device* field must specify a real device (*device_name* or *rootdisk.sn*).

ip_address – Use only when installing a dataless client and use *device* as the *server:path*. This value indicates to the server the IP address of the dataless client providing the /usr and /usr/kvm file systems. "-" – Use a dash (-) as the value of the *size* field when the *device* field is *server:path*, but you are not installing a dataless client.

file_system

This argument to the *filesys* keyword is optional. If it is not used, the *device* is set up as specified by the other arguments, but no file system is created. If it is used, its value is either:

mount_point_name – A real file system name indicating the mount point for the *device*.

swap – The *device* to be used for swap.

optional_parameters

This argument to `filesys` is optional. If you use it, specify the following:

preserve – Preserves the data on an existing partition. Use only when the *size* field is existing.

mount_options

A general string of mount options used to add to the `/etc/vfstab` entry for the file system. Use *mount_options* only when the *file_system* field specifies a real file system (a local file system or a remotely mounted file system).

Upgrade Parameters

The following keywords and argument values are used in a class file in which the type of installation is an upgrade. The `install_type` keyword is the only one required; all others are optional.

`install_type upgrade`

This keyword defines the type of installation described by the remainder of the class file. The values `initial_install` and `upgrade` are supported in the Solaris 2.2 environment.

`locale locale_name`

The localizations for the specified `locale_name` added during the upgrade. Multiple `locale` keywords can be specified.

`cluster cluster_name (add | delete)`

Which cluster is to be installed. If the specified `cluster_name` already exists on the system, it is automatically upgraded. The `add` and `delete` field indicates whether the cluster should be added or deleted from the configuration cluster and clusters previously specified. If you use this keyword in an upgrade, `delete` does not remove an existing cluster from the system.

`package package_name (add | delete)`

The package to be installed. If the specified `package_name` already exists on the system, it is automatically upgraded. The `add` and `delete` field indicates whether the package should be added or deleted from the configuration cluster and clusters previously specified. If you use this keyword in an upgrade, `delete` does not remove an existing package from the system.

Time Zones

The following is a list of legitimate time zone names. This information was gathered from the `/usr/share/lib/zoneinfo` directory.

CET	Jamaica	Brazil/East
CST6CDT	Japan	Brazil/West
Cuba	Libya	Canada/Atlantic
EET	MET	Canada/Central
EST	MST	Canada/East-Saskatchewan
EST5EDT	MST7MDT	Canada/Eastern
Egypt	NZ	Canada/Mountain
GB-Eire	Navajo	Canada/Newfoundland
GMT+10	PRC	Canada/Pacific
GMT+11	PST8PDT	Canada/Yukon
GMT+12	Poland	Chile/Continental
GMT+13	ROC	Chile/EasterIsland
GMT+8	ROK	Mexico/BajaNorte
GMT+9	Singapore	Mexico/BajaSur
GMT-1	Turkey	Mexico/General
GMT-10	W-SU	Mideast/Riyadh87
GMT-11	WET	Mideast/Riyadh88
GMT-12	Australia/Broken-Hill	Mideast/Riyadh89
GMT-2	Australia/LHI	US/Alaska
GMT-3	Australia/NSW	US/Aleutian
GMT-4	Australia/North	US/Arizona
GMT-5	Australia/Queensland	US/Central
GMT-6	Australia/South	US/East-Indiana
GMT-7	Australia/Sturt	US/Eastern
GMT-8	Australia/Tasmania	US/Hawaii
GMT-9	Australia/Victoria	US/Michigan
HST	Australia/West	US/Mountain
HongKong	Australia/Yancowinna	US/Pacific
Iran	Brazil/Acre	US/Samoa
Israel	Brazil/DeNoronha	US/Yukon

System Configuration Using NIS+

If your network is running NIS+, perform the following steps on the name server:

1. Create the locale table.
 - a. If your system is not in NIS compatibility mode, issue the command:

```
# nistbladm -D access=og=rmcd,w=r,n= -c \  
locale_tbl name=SI,nogw= locale=,nogw= \  
comment=,nogw= locale.org_dir.'nisdefaults -d'
```

- b. If your system is in NIS compatibility mode, issue the command:

```
# nistbladm -D access=og=rmcd,nw=r -c locale_tbl \  
name=SI,nogw= locale=,nogw= comment=,nogw=\  
locale.org_dir.'nisdefaults -d'
```

2. Use the following format to add an entry, which specifies the domain's default locale, to the locale table for each domain:

domainname locale

3. Use the `nistbladm` command to add the entries to the locale table. For example, the following command adds the locale US/Eastern to the domain garden (the default domain returned by the `nisdefaults -d` command):

```
# nistbladm -a name=sun_com. locale=de\  
comment=german locale.org_dir,'nisdefaults -d'
```

Configuring System Information

Certain information, such as the Ethernet address, host name, and IP address of the machine to be installed must be configured on the network name server so that the system can access the information during the automatic installation process.

If the network name server is running the Solaris 2.x environment, use Host Manager to update the `ethers` and `hosts` databases; otherwise, perform the following steps:

1. Update the network `ethers` database (`/etc/ethers`) by entering the Ethernet address and host name for the system to be installed.
2. Update the network `hosts` database (`/etc/hosts`) by entering the IP address and host name for the system to be installed.
3. Add the hostname alias `timehost` to one of the existing hosts in the `/etc/hosts` file.
4. Rebuild the network NIS+ tables.

For NIS+, issue the following commands on the NIS+ master server:

```
#nisaddent -avf /etc/ethers ethers
```

```
#nisaddent -avf /etc/hosts hosts
```

Setting Up Time Zone

To set up a time zone, perform the following steps:

1. Create or update the `timezone` database (`/etc/timezone`) by entering the time zone and the client name.
2. Update the `netmasks` database (`/etc/netmasks`) even if there is no actual subnetting in use.
3. Rebuild the network NIS+ tables.

For NIS+, issue the following commands on the NIS+ master server:

```
#nisaddent -avf /etc/timezone timezone
#nisaddent -avf /etc/netmasks netmasks
```

Adding Time Zone Information to the Name Server Switch File

To add time zone information to the name server switch file, add the following line to the `/etc/nsswitch.conf` file of the NIS+ servers:

```
timezone:                nisplus
```


Index

Symbols

- ! symbol 13-31
- # symbol 13-31
- && symbol 13-31
- /etc/bootparams file 13-12, 13-42
- /etc/bootparamsfile 13-9
- /etc/dfs/dfstab
 - command 13-43
- /etc/dfs/dfstab file 6-8, 13-13, 13-43
- /etc/ethers file 13-9
- /etc/hosts file 13-10, 13-22
- /etc/locale file 13-20
- /etc/netmasks file 13-23
- /etc/syslog.conf file 3-5, 3-9
- /etc/timezone file 13-22
- /etc/user_attr database 9-3, 9-6
- /etc/vfstab file 6-17
- /tftpboot directory 13-8, 13-11
- /usr/bin/prodreg
 - command 10-11
- /usr/lib/nfs/lockd
 - daemon 6-6
- /usr/lib/nfs/mountd
 - daemon 6-5
- /usr/lib/nfs/nfsd
 - daemon 6-5
- /usr/lib/nfs/statd
 - daemon 6-6
- /usr/sbin/in.rarpd
 - daemon 13-8

A

- action field 3-5, 3-8
- add_install_client
 - command 13-5, 13-14, 13-40
- add_to_install_server
 - command 13-25
- Address Resolution Protocol (ARP) 2-8
- AdminSuite 10-15
- ARP 2-8
- Asynchronous Transfer Mode (ATM) 2-8
- ATM protocol 2-8
- auth_attr database 9-9
- auto_master file 7-6
- autofs file 7-3, 7-5, 7-12
- automount command 7-3, 7-11
- automountd daemon 7-4

B

- banner command 2-12
- block device path 4-3
- boot net - install
 - command 13-45

C

- cachefslog command 8-3
- cachefsstat command 8-3, 8-7
- cachefswssize command 8-3, 8-11
- careadm commands 5-17

cfsadmin command 8-3, 8-8
client 1-4
 NFS 6-2
 NIS 12-3
commands
 /etc/dfs/dfstab 13-43
 /usr/bin/prodreg 10-11
 add_install_client 13-5,
 13-14, 13-40
 add_to_install_server 13
 -25
 automount 7-3, 7-11
 banner 2-12
 boot net - install 13-45
 cachefslog 8-3
 cachefsstat 8-3, 8-7
 cachefswssize 8-3, 8-11
 careadm 5-17
 cfsadmin 8-3, 8-8
 dfmounts 6-15
 dfshares 6-13, 6-14
 dumpadm 5-13
 fsck 8-13
 ifconfig -a 2-12
 logger 3-17
 lpadmin 9-19
 make 12-11, 12-28
 modify_install_server 13
 -25
 mount 6-16
 mountall 6-21
 NFS 6-4
 pfinstall 13-36
 roleadd 9-20, 9-22
 rpcinfo 2-20
 share 6-7, 6-9
 shareall 6-11
 su 9-19
 swap 5-9
 umount 6-21
 umountall 6-22
 unshare 6-11
 unshareall 6-12
 ypcat 12-17
 ypinit 12-11
 ypmatch 12-17

ypwhich 12-18
Console 10-3

D

dad disk driver 4-3
daemons
 /usr/lib/nfs/lockd 6-6
 /usr/lib/nfs/mountd 6-5
 /usr/lib/nfs/nfsd 6-5
 /usr/lib/nfs/statd 6-6
 /usr/sbin/in.rarpd 13-8
 automountd 7-4
 devfsadmd 4-9
 NFS 6-4, 6-5
 NFS server 6-5
 NFS server, client 6-6
 nfslogd 6-24
 rpc.bootparamd 13-9
 rpc.yppasswdd 12-5
 rpc.yppupdated 12-6
 syslogd 3-11
 ypbind 12-5
 ypserv 12-5
 ypxfrd 12-6
de-encapsulation 2-6
devfsadmd daemon 4-9
dfmounts command 6-15
dfshares command 6-13, 6-14
DHCP 2-9
directories
 /tftpboot 13-8, 13-11
DNS 2-9
Domain Name System
 (DNS) 2-9
dumpadm commands 5-13
Dynamic Host Configuration
 Protocol (DHCP) 2-9

E

encapsulation 2-6
exec_attr database 9-15

F

fdfs file system 5-2

File Transfer Protocol (FTP) 2-10
files

- /etc/bootparams 13-9,
13-12, 13-42
- /etc/dfs/dfstab 6-8, 13-13,
13-43
- /etc/ethers 13-9
- /etc/hosts 13-10, 13-22
- /etc/locale 13-20
- /etc/netmasks 13-23
- /etc/syslog.conf 3-5, 3-9
- /etc/timezone 13-22
- /etc/vfstab 6-17
- auto_masters 7-6
- autofs 7-3, 7-5, 7-12
- NFS 6-4
- rules 13-29
- sysidcfg 13-16

fsck command 8-13

FTP 2-10

H

hash symbol 13-31

HTTP 2-9

Hypertext Transfer Protocol
(HTTP) 2-9

I

ICMP 2-9

ifconfig -a command 2-12

Internet Control Message
Protocol (ICMP) 2-9

Internet layer 2-5

ISO/OSI network model
layers 2-4

J

JumpStart

- boot services 13-6
- boot, system identification
services 13-4
- configuration services 13-4
- definition 13-3
- installation services 13-4

process 13-8

L

layers

- ISO/OSI network model 2-4

- TCP/IP model 2-5

logger command 3-17

lpadmin command 9-19

M

m4 macro processor 3-11

make command 12-11, 12-28

master servers, NIS 12-2

modify_install_server
command 13-25

mount command 6-16

mountall command 6-21

N

Network File System (NFS) 2-9

Network Information System
Plus (NIS+) 2-9

network interface 2-5

NFS 2-9

- client 6-2

- commands 6-4

- daemons 6-4

- files 6-4

- mount daemon 6-5

- server 6-2

- server daemons 6-5

- server, client daemons 6-6

nfslogd daemon 6-24

NIS 13-9

- clients 12-3

- configuring client 12-19

- configuring master
server 12-13

- configuring slave
server 12-20

- master server 12-2

- slave server 12-3

NIS processes 12-4

NIS+ 2-9, 13-9

NIS, maps 11-7
NIS, namespace 11-7

P

pfinstall command 13-36
physical disk drivers
 dad 4-3
 sd 4-3
Point-to-Point Protocol
 (PPP) 2-8
PPP 2-8
processes, NIS 12-4
procfs file system 5-2
prof_attr database 9-12
pseudo file systems
 fdfs 5-2
 procfs 5-2
 swapfs 5-2
 tmpfs 5-2

R

RARP 2-8, 13-8
raw device path 4-4
RBAC 9-2
 databases 9-3
 /etc/user_attr 9-3, 9-6
 auth_attr 9-9
 exec_attr 9-15
 prof_attr 9-12
 delimiters 9-4
 tools
 roleadd 9-20
 useradd 9-20
Remote Procedure Call
 (RPC) 2-9
Reverse Address Resolution
 Protocol 13-8
Reverse Address Resolution
 Protocol (RARP) 2-8
RIP 2-10
rlogin 2-10
roleadd command 9-20, 9-22
roleadd tool 9-20
role-based access control 9-2
Routing Information Protocol

 (RIP) 2-10
RPC 2-9
rpc.bootparamd daemon 13-9
rpc.yppupdated daemon 12-6
rpcinfo command 2-20
rules file 13-29

S

sd SCSI disk driver 4-3
selector field 3-5
server 1-3
 NFS 6-2
services
 rlogin 2-10
 telnet 2-10
share command 6-7, 6-9
shareall command 6-11
Simple Mail Transport Protocol
 (SMTP) 2-10
Simple Network Management
 Protocol (SNMP) 2-10
slave servers, NIS 12-3
SMC 10-3
SMTP 2-10
SNMP 2-10
Solaris AdminSuite 10-15
Solaris Management
 Console 10-3
Solstice DiskSuite 4-5, 4-6
StorEdge Volume Manager 4-5,
 4-6
su command 9-19
Sun StorEdge Volume
 Manager 4-5, 4-6
SunInstall 13-9
swap commands 5-9
swapfs file system 5-2
symbols
 ! 13-31
 # 13-31
 && 13-31
sysidcfg file 13-16
syslogd daemon 3-11

T

- TCP 2-9
- TCP/IP 2-7
- TCP/IP model layers 2-5
- telnet 2-10
- tmpfs file system 5-2
- Transmission Control Protocol (TCP) 2-9
- Transmission Control Protocol (TCP/IP) 2-7
- Transport layer 2-5

U

- UDP 2-9
- umount command 6-21
- umountall command 6-22
- unshare command 6-11
- unshareall command 6-12
- User Datagram Protocol (UDP) 2-9
- useradd tool 9-20

V

- virtual disk
 - access paths 4-5

Y

- ypbind daemon 12-5
- ypcat command 12-17
- ypinit command 12-11
- ypmatch command 12-17
- yppasswdd daemon 12-5
- ypserv daemon 12-5
- ypwhich command 12-18
- ypxfrd daemon 12-6

Copyright 2000 Sun Microsystems Inc., 901 San Antonio Road, Palo Alto, California 94303, Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a.

Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley 4.3 BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company Ltd.

Sun, Sun Microsystems, le logo Sun, JumpStart, Solaris, Solaris Management Console, Solstice AdminSuite, Solaris WebStart, Solstice DiskSuite, StorEdge Volume Manager, SunRay, Java, JDK, AnswerBook, Jumpstart, JavaStation, Sun-4, et SunInstall sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Toutes les marques SPARC sont utilisées sous licence sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays.

UNIX est une marques déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

L'accord du gouvernement américain est requis avant l'exportation du produit.

Le système X Window est un produit de X Consortium, Inc.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.



Please
Recycle



Adobe PostScript

