

System Administration for the Solaris 10 OS Part 2

Student Guide

SA-202-S10 Rev D.1

D61738GC11
Edition 1.1
D65082 and D65083

ORACLE®

Copyright © 2009, 2010, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information, is provided under a license agreement containing restrictions on use and disclosure, and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except as expressly permitted in your license agreement or allowed by law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Sun Microsystems, Inc. Disclaimer

This training manual may include references to materials, offerings, or products that were previously offered by Sun Microsystems, Inc. Certain materials, offerings, services, or products may no longer be offered or provided. Oracle and its affiliates cannot be held responsible for any such references should they appear in the text provided.

Restricted Rights Notice

If this documentation is delivered to the U.S. Government or anyone using the documentation on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. UNIX is a registered trademark licensed through X/Open Company, Ltd.

This page intentionally left blank.

tim.jia (tim.jia@thomsonreuters.com) has a non-transferable license
to use this Student Guide.

This page intentionally left blank.

tim.jia (tim.jia@thomsonreuters.com) has a non-transferable license
to use this Student Guide.

Table of Contents

| | |
|---|------------------|
| About This Course | Preface-i |
| Course Goals | Preface-i |
| Course Map | Preface-ii |
| Topics Not Covered | Preface-iii |
| How Prepared Are You? | Preface-iv |
| Introductions | Preface-v |
| How to Use Course Materials | Preface-vi |
| Conventions | Preface-vii |
| Icons | Preface-vii |
| Typographical Conventions | Preface-viii |
| Managing Swap Space, Core Files and Crash Dumps..... | 1-1 |
| Objectives | 1-1 |
| Introducing Virtual Memory | 1-2 |
| Physical RAM | 1-2 |
| Swap Space | 1-3 |
| The swapfs File System | 1-4 |
| Paging | 1-5 |
| Configuring Swap Space | 1-7 |
| Displaying the Current Swap Configuration | 1-7 |
| Adding Swap Space | 1-9 |
| Removing Swap Space | 1-10 |
| What is a Core File? | 1-12 |
| Why Generate a Core File? | 1-12 |
| Managing Crash Dump Behavior | 1-13 |
| The Crash Dump | 1-13 |
| Displaying the Current Dump Configuration | 1-14 |
| Changing the Crash Dump Configuration | 1-15 |
| Managing Core File Behavior | 1-17 |
| Core Files | 1-17 |
| Displaying the Current Core File Configuration | 1-18 |
| Changing the Core File Configuration | 1-20 |
| Exercise 1: Managing swap Utility Configuration | 1-26 |
| Preparation | 1-26 |

| | |
|---|------------|
| Tasks | 1-28 |
| Exercise 2: Collecting the Crash Dump and Core Dump..... | 1-30 |
| Preparation..... | 1-30 |
| Task 1 – Using the dumpadm Command to Display the Core File Directory Location | 1-31 |
| Task 2 – Using the coreadm Command to Configure Core File Storage Locations | 1-31 |
| Exercise Summary..... | 1-33 |
| Exercise 1 Solutions: Managing swap Utility Configuration | 1-34 |
| Tasks and Solutions | 1-36 |
| Exercise 2 Solutions: Collecting the Crash Dump and Core Dump..... | 1-40 |
| Preparation..... | 1-40 |
| Task 1 – Using the dumpadm Command to Display the Core File Directory Location | 1-41 |
| Task 2 – Using the coreadm Command to Configure Core File Storage Locations | 1-43 |
| Configuring NFS | 2-1 |
| Objectives | 2-1 |
| Introducing the Benefits of NFS..... | 2-2 |
| Benefits of Centralized File Access..... | 2-3 |
| Benefits of Common Software Access..... | 2-3 |
| Introducing the Fundamentals of the NFS Distributed File System | 2-4 |
| NFS Server..... | 2-5 |
| NFS Client | 2-6 |
| Managing an NFS Server | 2-7 |
| The NFS Server Files..... | 2-7 |
| The NFS Server Daemons | 2-10 |
| Managing the NFS Server Daemons | 2-13 |
| NFS Server Commands | 2-15 |
| Configuring the NFS Server for Sharing Resources..... | 2-16 |
| Managing the NFS Client..... | 2-22 |
| NFS Client Files | 2-22 |
| NFS Client Daemons | 2-23 |
| Managing the NFS Client Daemons | 2-24 |
| NFS Client Commands..... | 2-25 |
| Configuring the NFS Client for Mounting Resources | 2-25 |
| Enabling NFS Server Logging | 2-31 |
| Fundamentals of NFS Server Logging | 2-31 |
| Configuring NFS Log Paths | 2-32 |
| Initiating NFS Logging..... | 2-34 |
| Configuring the nfslogd Daemon Behavior | 2-35 |
| NFS Version 4 (NFSv4)..... | 2-36 |
| Pseudo-File System | 2-36 |

| | |
|---|------------|
| Strong Security | 2-38 |
| Compound Procedures | 2-39 |
| Extended Attributes..... | 2-40 |
| File Handles | 2-40 |
| Delegation | 2-41 |
| Configuring an NFSv4 Server | 2-42 |
| Configuring an NFSv4 Client..... | 2-43 |
| Troubleshooting NFS Errors | 2-45 |
| The rpcbind failure Error..... | 2-45 |
| The server not responding Error..... | 2-46 |
| The NFS client fails a reboot Error | 2-46 |
| The service not responding Error | 2-47 |
| The program not registered Error | 2-47 |
| The stale NFS file handle Error..... | 2-48 |
| The unknown host Error | 2-48 |
| The mount point Error | 2-48 |
| The no such file Error | 2-49 |
| Exercise: Configuring NFS..... | 2-50 |
| Preparation..... | 2-50 |
| Tasks | 2-50 |
| Exercise Summary..... | 2-53 |
| Exercise Solutions: Configuring NFS | 2-54 |
| Preparation..... | 2-54 |
| Tasks and Solutions | 2-54 |
| Configuring AutoFS | 3-1 |
| Objectives | 3-1 |
| Introducing the Fundamentals of AutoFS..... | 3-2 |
| AutoFS File System..... | 3-3 |
| The automountd Daemon..... | 3-4 |
| The automount Command | 3-4 |
| Using Automount Maps | 3-5 |
| Configuring the Master Map..... | 3-6 |
| Identifying Mount Points for Special Maps | 3-7 |
| Adding Direct Map Entries | 3-8 |
| Adding Indirect Map Entries | 3-11 |
| Updating the Automount Maps | 3-13 |
| Stopping and Starting the Automount System..... | 3-16 |
| Exercise: Using the Automount Facility | 3-18 |
| Preparation..... | 3-18 |
| Tasks | 3-18 |
| Exercise Summary..... | 3-22 |
| Exercise Solutions: Using the Automount Facility..... | 3-23 |
| Preparation..... | 3-23 |
| Tasks and Solutions | 3-23 |

| | |
|--|------------|
| Describing RAID..... | 4-1 |
| Objectives | 4-1 |
| Introducing RAID | 4-2 |
| RAID 0 | 4-2 |
| RAID 1 | 4-5 |
| RAID 5 | 4-12 |
| Storage Configurations | 4-15 |
| Configuring Solaris Volume Manager Software..... | 5-1 |
| Objectives | 5-1 |
| Introducing Solaris Volume Manager Software Concepts | 5-2 |
| Logical Volume | 5-2 |
| Soft Partitions | 5-3 |
| Introducing the State Database | 5-4 |
| Introducing Hot Spares and Hot Spare Pools..... | 5-7 |
| Solaris Volume Manager Concepts | 5-8 |
| The State Database Replicas | 5-9 |
| Creating the State Database..... | 5-9 |
| Configuring RAID-0 | 5-11 |
| RAID-0 Striped Volumes | 5-12 |
| Creating a RAID-0 Volume | 5-13 |
| Configuring RAID-1 | 5-17 |
| Building a Mirror of the Root (/) File System..... | 5-20 |
| The Scenario..... | 5-21 |
| Creating the RAID-0 Volumes | 5-21 |
| Creating the RAID-1 Volume..... | 5-22 |
| Configuring an x86-Based System for Mirrored Failover. | 5-27 |
| Unmirroring the root (/) File System | 5-34 |
| Exercise: Mirroring the root (/) File System on SPARC-Based Systems | 5-36 |
| Preparation..... | 5-36 |
| Tasks | 5-36 |
| Exercise Summary..... | 5-40 |
| Exercise Solutions: Mirroring the root (/) File System on SPARC-Based Systems | 5-41 |
| Preparation..... | 5-41 |
| Tasks and Solutions | 5-41 |
| Exercise: Mirroring the root (/) File System on x86/x64-Based Systems..... | 5-49 |
| Preparation..... | 5-49 |
| Tasks | 5-50 |
| Exercise Summary..... | 5-54 |
| Exercise Solutions: Mirroring the root (/) File System on x86/x64-Based Systems | 5-55 |
| Preparation..... | 5-55 |
| Tasks and Solutions | 5-56 |

| | |
|---|------------|
| Configuring Role-Based Access Control (RBAC) | 6-1 |
| Objectives | 6-1 |
| Introducing RBAC Fundamentals | 6-2 |
| Key RBAC Files | 6-2 |
| Roles | 6-3 |
| Assigning Rights Profiles To Users | 6-4 |
| Assigning Rights Profiles To Roles | 6-8 |
| Assigning Roles to Users | 6-11 |
| Using Roles | 6-12 |
| Authorizations..... | 6-13 |
| Assigning Authorizations..... | 6-16 |
| Assigning Authorizations to User Accounts | 6-17 |
| Assigning Authorizations to Roles..... | 6-18 |
| Assigning Authorizations to Rights Profiles | 6-19 |
| Make root User Into a Role | 6-20 |
| RBAC Configuration File Summary..... | 6-22 |
| The /etc/user_attr File..... | 6-22 |
| The /etc/security/prof_attr File..... | 6-23 |
| The /etc/security/exec_attr File..... | 6-25 |
| The /etc/security/auth_attr File..... | 6-27 |
| Exercise: Configuring RBAC | 6-29 |
| Preparation..... | 6-29 |
| Tasks | 6-29 |
| Exercise Summary..... | 6-31 |
| Exercise Solutions: Configuring RBAC..... | 6-32 |
| Preparation..... | 6-32 |
| Tasks and Solutions | 6-32 |
| Configuring System Messaging..... | 7-1 |
| Objectives | 7-1 |
| Introducing the syslog Function | 7-2 |
| The syslog Concept..... | 7-2 |
| The /etc/syslog.conf File | 7-3 |
| The syslogd Daemon and the m4 Macro Processor | 7-8 |
| Configuring the /etc/syslog.conf File..... | 7-11 |
| Message Routing | 7-11 |
| Stopping and Starting the syslogd Daemon..... | 7-12 |
| Configuring syslog Messaging | 7-13 |
| Enabling TCP Tracing | 7-13 |
| Monitoring a syslog File in Real Time | 7-15 |
| Adding One-Line Entries to a System Log File | 7-17 |
| Exercise: Using the syslog Function and Auditing Utilities | 7-19 |
| Preparation..... | 7-19 |
| Tasks | 7-19 |
| Exercise Summary..... | 7-25 |

| | |
|---|------------|
| Exercise Solutions: Using the <code>syslog</code> Function and Auditing Utilities..... | 7-26 |
| Preparation..... | 7-26 |
| Tasks and Solutions | 7-26 |
| Using Name Services | 8-1 |
| Objectives | 8-1 |
| Introducing the Name Service Concept..... | 8-2 |
| Domain Name System (DNS) | 8-4 |
| Network Information Service (NIS) | 8-6 |
| Network Information Service Plus (NIS+) | 8-8 |
| Lightweight Directory Access Protocol (LDAP) | 8-9 |
| Name Service Features Summary..... | 8-11 |
| Introducing the Name Service Switch File | 8-12 |
| Database Sources..... | 8-14 |
| Status Codes..... | 8-15 |
| Actions | 8-15 |
| Configuring the Name Service Cache Daemon (<code>nscd</code>) | 8-17 |
| The <code>nscd</code> Daemon | 8-17 |
| Configuring the <code>nscd</code> Daemon | 8-17 |
| Stopping and Starting the <code>nscd</code> Daemon | 8-19 |
| Retrieving Name Service Information | 8-21 |
| The <code>getent</code> Command | 8-21 |
| Using the <code>getent</code> Command | 8-22 |
| Exercise: Reviewing Name Services..... | 8-23 |
| Preparation..... | 8-23 |
| Tasks | 8-23 |
| Exercise Summary..... | 8-25 |
| Exercise Solutions: Reviewing Name Services | 8-26 |
| Preparation..... | 8-26 |
| Tasks and Solutions | 8-26 |
| Configuring Name Service Clients | 9-1 |
| Objectives | 9-1 |
| Configuring a DNS Client | 9-2 |
| Configuring the DNS Client During Installation | 9-2 |
| Editing DNS Client Configuration Files | 9-5 |
| Setting Up an LDAP Client..... | 9-7 |
| Client Authentication | 9-7 |
| Client Profile and Proxy Account..... | 9-8 |
| Client Initialization | 9-8 |
| Configuring the LDAP Client During Installation..... | 9-9 |
| Initializing the Native LDAP Client..... | 9-12 |
| Copying the <code>/etc/nsswitch.ldap</code> File to the <code>/etc/nsswitch.conf</code> File | 9-14 |
| Listing LDAP Entries..... | 9-15 |
| Unconfiguring an LDAP Client | 9-16 |

| | |
|---|-------------|
| Exercise: Configuring a System to Use DNS and LDAP | 9-17 |
| Preparation..... | 9-17 |
| Tasks | 9-17 |
| Exercise Summary..... | 9-19 |
| Exercise Solutions: Configuring a System to Use DNS and LDAP .. | 9-20 |
| Preparation..... | 9-20 |
| Tasks and Solutions | 9-20 |
| Introduction to Zones..... | 10-1 |
| Objectives | 10-1 |
| Introducing Solaris Zones..... | 10-2 |
| Server Consolidation Solutions..... | 10-2 |
| Resource Sharing..... | 10-3 |
| Zone Features | 10-4 |
| Zone Concepts | 10-5 |
| Zone Types..... | 10-5 |
| Zone Daemons..... | 10-8 |
| Zone File Systems | 10-9 |
| Zone Networking..... | 10-11 |
| Zone States | 10-12 |
| Configuring Zones | 10-14 |
| Identifying Zone Components | 10-14 |
| Allocating File System Space..... | 10-14 |
| Using the zonecfg Command | 10-15 |
| The zonecfg Resources Parameters..... | 10-17 |
| Zone Configuration Walk-Through | 10-19 |
| Viewing the Zone Configuration..... | 10-21 |
| Using the zoneadm Command | 10-24 |
| Verifying a Configured Zone | 10-24 |
| Installing a Configured Zone | 10-25 |
| Booting a Zone..... | 10-25 |
| Halting a Zone | 10-26 |
| Rebooting a Zone | 10-27 |
| Logging Into and Working With the Zone..... | 10-27 |
| Moving a Zone | 10-29 |
| Migrating a Zone..... | 10-30 |
| Pre-Validating Zone Migration..... | 10-32 |
| Deleting a Zone | 10-33 |
| Installing Packages in Zones | 10-34 |
| Packaging for Sparse and Whole Root Zones | 10-34 |
| Listing Parameters for Packages..... | 10-35 |
| Package Operations Possible in the Global Zone | 10-36 |
| Package Operations Possible in a Non-Global Zone | 10-37 |
| Using 1x Branded Zone | 10-39 |
| Planning for 1x Branded Zone | 10-39 |

| | |
|--|-------------|
| Installing and Configuring an 1x Branded Zone | 10-41 |
| Exercise 1: Configuring Zones | 10-42 |
| Preparation..... | 10-42 |
| Task 1 – Creating and Installing a Non-Global Zone | 10-43 |
| Task 2 – Administering Users and Data Within Zones ... | 10-45 |
| Task 3 – Installing Software Packages in Zones | 10-45 |
| Task 4 - Removing Zones..... | 10-46 |
| Exercise Summary..... | 10-47 |
| Exercise 1 Solutions: Configuring Zones | 10-48 |
| Preparation..... | 10-48 |
| Task 1 – Creating and Installing a Non-Global Zone | 10-49 |
| Task 2 – Administering Users and Data Within Zones ... | 10-55 |
| Task 3– Installing Software Packages in Zones | 10-56 |
| Task 4 - Removing Zones..... | 10-60 |
| Exercise 2: Validating a Zone Migration Before the Migration Is Performed..... | 10-62 |
| Preparation..... | 10-62 |
| Task 1 - Configuring a Non-Global Zone | 10-62 |
| Task 2 - Creating and Transferring Zone Manifest | 10-63 |
| Task 3 - Validating Zone Migration Before the Actual Migration | 10-63 |
| Exercise Summary..... | 10-64 |
| Exercise 2 Solutions: Validating a Zone Migration Before the Migration Is Performed | 10-65 |
| Preparation..... | 10-65 |
| Task 1 - Configuring a Non-Global Zone | 10-65 |
| Task 2 - Creating and Transferring Zone Manifest | 10-66 |
| Task 3 - Validating Zone Migration Before the Actual Migration | 10-67 |
| Exercise 3: Using Solaris Containers for Linux Applications .. | 10-68 |
| Preparation..... | 10-68 |
| Task 1 - Configuring the 1x Branded Zone..... | 10-68 |
| Task 2 - Installing the 1x Branded Zone | 10-69 |
| Task 3 - Booting the 1x Branded Zone..... | 10-69 |
| Task 4 - Logging On to the 1x Branded Zone | 10-69 |
| Exercise Summary..... | 10-70 |
| Exercise 3 Solutions: Using Solaris Containers for Linux Applications..... | 10-71 |
| Preparation..... | 10-71 |
| Task 1 - Configuring the 1x Branded Zone..... | 10-71 |
| Task 2 - Installing the 1x Branded Zone | 10-72 |
| Task 3 - Booting the 1x Branded Zone..... | 10-72 |
| Task 4 - Logging On to the 1x Branded Zone | 10-73 |
| Introduction to LDAP | 11-1 |
| Objectives | 11-1 |

| | |
|--|-------------|
| LDAP As A Naming Service | 11-2 |
| The Lightweight Directory Access Protocol (LDAP)..... | 11-2 |
| Basic LDAP Concepts and Terminology | 11-3 |
| Defining a Directory Service | 11-4 |
| Directory Schema | 11-4 |
| LDAP | 11-7 |
| Four Defined LDAP Models..... | 11-8 |
| LDAP Directory: Information Model..... | 11-9 |
| LDAP Entries: Naming Model..... | 11-10 |
| LDAP Protocol: Functional Model | 11-11 |
| LDAP Request: Security Model | 11-13 |
| Defining LDAP Search Parameters | 11-15 |
| Utilizing LDIF..... | 11-18 |
| Default Directory Information Tree (DIT)..... | 11-19 |
| General LDAP Tools..... | 11-20 |
| DSMLv2 Over HTTP/SOAP | 11-22 |
| Directory Server Requirements..... | 11-23 |
| System Requirements | 11-23 |
| Operating Systems and Platforms | 11-24 |
| DSEE Components..... | 11-25 |
| DSEE Security Features | 11-26 |
| Configuring A Client System To Use LDAP | 11-27 |
| ldapclient Utility | 11-27 |
| /etc/nsswitch.conf File | 11-28 |
| Enabling DNS With LDAP | 11-30 |
| LDAP Service..... | 11-30 |
| Verifying Basic Client-Server Communication | 11-32 |
| Configuring JumpStart Installation Using the Solaris 10 Operating System..... | 12-1 |
| Objectives | 12-1 |
| Introducing JumpStart Configurations..... | 12-2 |
| Purpose of JumpStart | 12-2 |
| Boot Services | 12-3 |
| Identification Services | 12-5 |
| Configuration Services | 12-7 |
| Installation Services | 12-9 |
| Implementing a Basic JumpStart Server | 12-11 |
| Spooling the Operating System Image | 12-11 |
| Editing the sysidcfg File..... | 12-14 |
| Running the check Script | 12-22 |
| Running the add_install_client Script..... | 12-24 |
| Booting a SPARC JumpStart Client..... | 12-27 |
| Introducing the SPARC JumpStart Client | |
| Boot Sequence..... | 12-28 |
| Exercise 1: Configuring a JumpStart Server to Support | |

| | |
|--|-------------|
| One SPARC JumpStart Client | 12-35 |
| Preparation..... | 12-35 |
| Task Summary..... | 12-35 |
| Tasks | 12-36 |
| Booting and Installing x86/x64 Systems Over the Network | |
| With PXE | 12-42 |
| What is PXE?..... | 12-42 |
| Guidelines for Booting With PXE | 12-42 |
| Establishing DHCP Services for JumpStart Clients | 12-43 |
| Configuring DHCP Services on a JumpStart Server | 12-43 |
| Booting an x86/x64 JumpStart Client Using DHCP | 12-50 |
| Exercise 2: Configuring a JumpStart Server to Support | |
| One x86/x64 JumpStart Client (Demonstration)..... | 12-51 |
| Preparation..... | 12-51 |
| Task Summary..... | 12-51 |
| Tasks | 12-52 |
| Setting Up JumpStart Software Configuration Alternatives ... | 12-71 |
| Setting Up a Boot-Only Server for SPARC Clients | 12-71 |
| Setting Up Identification Service Alternatives | 12-76 |
| Setting Up Configuration Service Alternatives | 12-81 |
| Keywords and Arguments | 12-84 |
| Examples of Profile Files..... | 12-86 |
| Finish Scripts..... | 12-87 |
| Setting Up Installation Service Alternatives | 12-89 |
| New Solaris 10 Keywords..... | 12-92 |
| Adding Patches Using the patch Keyword..... | 12-98 |
| JumpStart Installation For A ZFS Root (/) File System..... | 12-100 |
| ZFS Root Flash Install..... | 12-101 |
| Creating Additional Boot Environments Using Live | |
| Upgrade And Jumpstart | 12-103 |
| Jumpstart And Zones | 12-104 |
| Exercise 3: (Optional) Creating a ZFS Mirrored Root Pool.... | 12-105 |
| Troubleshooting JumpStart | 12-111 |
| Resolving Boot Problems | 12-111 |
| Resolving Identification Problems | 12-114 |
| Resolving Configuration Problems | 12-116 |
| Resolving Installation Problems | 12-117 |
| Resolving Begin and Finish Script Problems | 12-118 |
| Identifying Log Files..... | 12-119 |
| Performing Live Upgrade Using the Solaris 10 Operating System | 13-1 |
| Objectives | 13-1 |
| Introducing Solaris Live Upgrade | 13-2 |
| Solaris Live Upgrade Process..... | 13-3 |
| Multiple Release Compatibility | 13-3 |

| | |
|--|-------|
| Solaris Live Upgrade System Requirements..... | 13-4 |
| Solaris Live Upgrade Disk Space Requirements | 13-4 |
| Guidelines for Selecting Slices for File Systems | 13-5 |
| Guidelines for Selecting a Slice for the root (/) | |
| File System | 13-5 |
| Guidelines for Selecting a Slice for a swap File System | 13-6 |
| Guidelines for Selecting Slices for Shareable File Systems | 13-7 |
| Installing Solaris Live Upgrade | 13-7 |
| Creating a Boot Environment..... | 13-8 |
| Live Upgrade Commands | 13-10 |
| Example Procedure: Creating A Base Master Flash Archive .. | 13-11 |
| Example Procedure: Cloning An Alternate Boot | |
| Environment From a Running System | 13-12 |
| Example Procedure: Modifying the State of the New | |
| Boot Environment | 13-19 |
| Example Procedure: Creating a Differential Archive Using | |
| Live Upgrade Boot Environments | 13-21 |
| Example Procedure: Applying a Differential Flash Archive | |
| Using Live Upgrade BE's..... | 13-23 |
| Example Procedure: Using Live Upgrade To Patch A System | 13-30 |
| Upgrading by Using a JumpStart Profile | 13-34 |
| Creating a Solaris Live Upgrade Profile | 13-36 |
| Upgrading a Boot Environment by Using a Custom | |
| JumpStart Profile (Command-Line Interface)..... | 13-38 |
| To Install a Solaris Flash Archive on a Boot | |
| Environment (Command-Line Interface) | 13-38 |
| To Install a Solaris Flash Archive With a Profile | |
| (Command-Line Interface) | 13-40 |
| Exercise: Patching and Upgrading Using Solaris | |
| Live Upgrade | 13-41 |
| Preparation..... | 13-41 |
| Task 1 - Creating a New UFS Boot Environment | 13-41 |
| Task 2 - Creating a New ZFS Boot Environment | 13-42 |
| Task 3 - Patching a UFS Boot Environment | 13-42 |
| Task 4 - Upgrading UFS Boot to ZFS Boot | 13-43 |
| Task 5 - Booting to a ZFS Boot Environment | 13-43 |
| Exercise Summary..... | 13-44 |
| Exercise Solution: Patching and Upgrading Using | |
| Live Upgrade | 13-45 |
| Preparation..... | 13-45 |
| Task 1 - Creating a New UFS Boot Environment | 13-45 |
| Task 2 - Creating a New ZFS Boot Environment | 13-46 |
| Task 3 - Patching a UFS Boot Environment | 13-47 |
| Task 4 - Upgrading UFS Boot to ZFS Boot | 13-47 |
| Task 5 - Booting to a ZFS Boot Environment | 13-47 |

tim jia (tim.jia@thomsonreuters.com) has a non-transferable license
to use this Student Guide.

Preface

About This Course

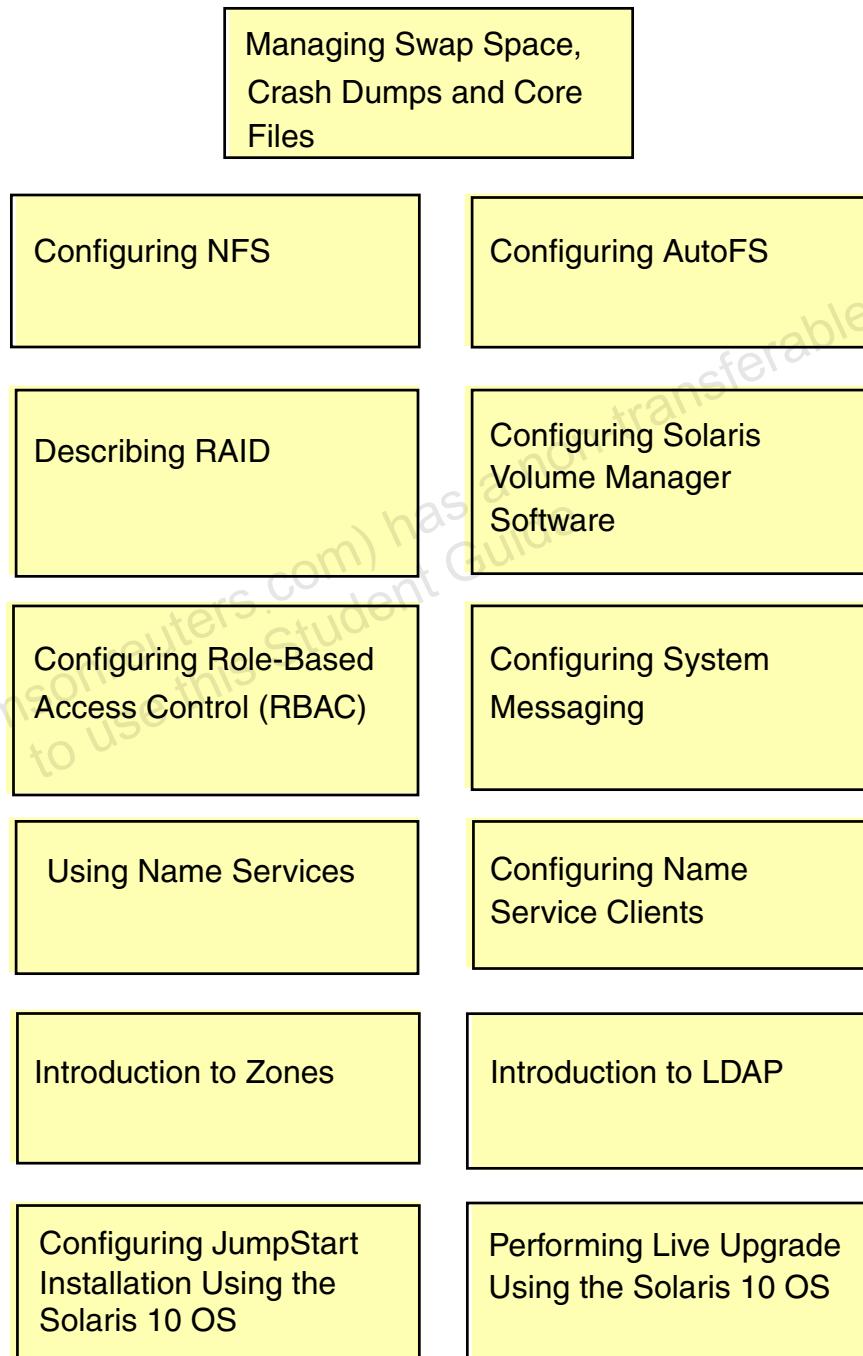
Course Goals

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

- Configure swap space
- Manage crash dumps and core files
- Manage NFS server and clients
- Use automount maps
- Describe RAID
- Manage storage volumes
- Manage RBAC
- Describe the syslog function and the /etc/syslog.conf file
- Set up name services
- Configure a DNS client and an LDAP client
- Configure, install, boot, move, migrate, and delete zones
- Describe basic LDAP concepts and terminology
- Implement a basic JumpStart server for SPARC® and x86/x64 clients
- Describe booting x86/x64 systems using the Preboot Execution Environment (PXE)
- Perform Live Upgrade using the Solaris 10 Operating System

Course Map

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



Topics Not Covered

This course does not cover the following topics. Many of these topics are covered in other courses offered by Sun Educational Services.

- Software package administration – Covered in SA-200-S10: *System Administration for the Solaris™ 10 Operating System, Part 1*
- Patch maintenance – Covered in SA-200-S10: *System Administration for the Solaris™ 10 Operating System, Part 1*
- Advanced file permissions – Covered in SA-200-S10: *System Administration for the Solaris™ 10 Operating System, Part 1*
- Backup and recovery – Covered in SA-200-S10: *System Administration for the Solaris™ 10 Operating System, Part 1*
- Hardware or software troubleshooting – Covered in ST-350: *Sun™ Systems Fault Analysis Workshop*
- System tuning – Covered in SA-400: *Solaris System Performance Management*
- Detailed shell programming – Covered in SA-245: *Shell Programming for System Administrators*
- Detailed shell programming – Covered in SA-245: *Shell Programming for System Administrators*
- Detailed network administration concepts – Covered in SA-300-S10: *Network Administration for the Solaris™ 10 Operating System*

Refer to the Sun 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 answer yes to the following questions?

- Can you install and boot the Solaris™ 10 Operating System (Solaris 10 OS) on a stand-alone workstation?
- Can you implement basic system security?
- Can you add users to the system using the Solaris Management Console software?
- Can you use the pkgadd command to add software packages?
- Can you monitor and mount file systems?
- Can you manage disk devices and processes?
- Can you perform backups and restorations?

Introductions

Now that you have been introduced to the course, introduce yourself to the other students and the instructor, addressing the following items:

- Name
- Company affiliation
- Title, function, and job responsibility
- Experience related to topics presented in this course
- Reasons for enrolling in this course
- Expectations for this course

How to Use Course Materials

To enable you to succeed in this course, these course materials use a learning module that is composed of the following components:

- Objectives – You should be able to accomplish the objectives after completing a portion of instructional content. Objectives support goals and can support other higher-level objectives.
- Lecture – The instructor will present information specific to the objective of the module. This information will help you learn the knowledge and skills necessary to succeed with the activities.
- Activities – The activities take on various forms, such as an exercise, self-check, discussion, and demonstration. Activities are used to facilitate the mastery of an objective.
- Visual aids – The instructor might use several visual aids to convey a concept, such as a process, in a visual form. Visual aids commonly contain graphics, animation, and video.

Note – Many system administration tasks for the Solaris OS can be accomplished in more than one way. The methods presented in the courseware reflect recommended practices used by Sun Educational Services.



Conventions

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

Icons



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



Note – Indicates additional information that can help students but is not crucial to their understanding of the concept being described. Students should be able to understand the concept or complete the task without this information. Examples of notational information include keyword shortcuts and minor system adjustments.



Caution – Indicates that there is a risk of personal injury from a nonelectrical hazard, or risk of irreversible damage to data, software, or the operating system. A caution indicates that the possibility of a hazard (as opposed to certainty) might happen, depending on the action of the user.



Power user – Indicates additional supportive topics, ideas, or other optional information.

Typographical Conventions

Courier is used for the names of commands, files, directories, user names, host names, programming code, and on-screen computer output; for example:

Use the `ls -al` command to list all files.

```
host1# cd /home
```

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

To list the files in this directory, type the following:

```
# ls
```

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

To delete a file, use the `rm filename` command.

Courier italic bold is used to represent variables whose values are to be entered by the student as part of an activity; for example:

Type `chmod a+rwx filename` to grant read, write, and execute rights for *filename*.

Palatino italics is used for book titles, new words or terms, or words that you want to emphasize; for example:

Read Chapter 6 in the *User's Guide*.

These are called *class* options.

Module 1

Managing Swap Space, Core Files and Crash Dumps

Objectives

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

- Configure swap space
- Manage core file behavior
- Manage crash dump behavior

Introducing Virtual Memory

Virtual memory combines RAM and dedicated disk storage areas known as swap space. Virtual memory management software maps copies of files on disk to virtual addresses. Programs use these virtual addresses rather than real addresses to store instructions and data. Virtual memory makes it possible for the operating system (OS) to use a large range of memory. However, the kernel must translate the virtual memory addresses into real address in RAM before the actual program instruction is performed on a central processing unit (CPU).

Physical RAM

Physical memory refers to the actual RAM installed on a computer. When working with swap space, RAM is the most critical resource in your system. The amount of physical memory varies depending on the system that runs the Solaris 10 OS. The code for each active process and any data required by each process must be mapped into physical memory before execution can take place.

Virtual and Physical Addresses

The Solaris 10 OS virtual memory management system maps the files on disk to virtual addresses in virtual memory. The virtual memory management system then translates the virtual addresses into real, physical addresses in physical memory, because programs require instructions or data in these files. The CPU uses the data and instructions when they are placed in physical memory.

Anonymous Memory Pages

Physical memory pages associated with a running process can contain private data or stack information that does not exist in any file system on disk. Since these memory pages contain information that is not also a named file on the disk, these pages are known as anonymous memory pages. Anonymous memory pages are backed by swap space; in other words, swap space is used as a temporary storage location for data while it is swapped out of memory.

Swap Space

A system's virtual memory is a combination of the available random access memory (RAM) and disk space. Portions of the virtual memory are reserved as swap space. Swap space can be defined as a temporary storage location that is used when system's memory requirements exceed the size of available RAM.

While the amount of physical memory in a system is constant, the use of the physical memory varies. Often processes conflict over which one gets access to physical memory space. Sometimes a process must give up some of its memory space allocation to another process. The process has some of its pages in RAM paged out. Anonymous memory pages are placed in a swap area, but unchanged file system pages are not placed in swap areas, because file system data exists as permanent storage on the disk, and can be removed from physical memory.

Swap Slices

The primary swap space on the system is a disk slice. In the Solaris 10 OS, the default location for the primary swap space is slice 1 of the boot disk, which by default, starts at cylinder 0. You can change the default location during a custom installation. Each time you reboot the system, an entry in the /etc/vfstab file determines the configuration of the swap partition. As additional swap space becomes necessary, you can configure additional swap slices. Plan your swap slice location carefully. If you have additional storage space outside of the system disk, place the swap slice on an additional drive to reduce the load on the system disk drive.

Swap Files

It is also possible to provide additional swap space on a system by using swap files. Swap files are files that reside on a file system, and that have been created using the mkfile command. These files might be useful in some cases. For example, swap files are useful when additional swap space is required, but there are no free disk slices and reslicing a disk to add more swap is not a practical solution. Swap files can be permanently included in the swap configuration by creating an entry for the swap file in the /etc/vfstab file.

The swapfs File System

When the kernel runs a process, swap space for any private data or stack space for the process must be allocated. The allocation occurs in case the stack information or private data might need to be paged out of physical memory, for example, if there are multiple processes contending for limited memory space.

Because of the virtual swap space provided by the swapfs file system in the Solaris 10 OS, there is less need for physical swap space on systems with a large available memory. The decreased need for physical swap space occurs because the swapfs file system provides virtual swap space addresses rather than real physical swap space addresses in response to swap space allocation requests. Therefore, you need physical swap space on disk, only in the event that the physical RAM pages containing private data need to be paged out.

Figure 1-1 shows that the swap space resides outside the physical RAM as a swap partition or as a swap file.

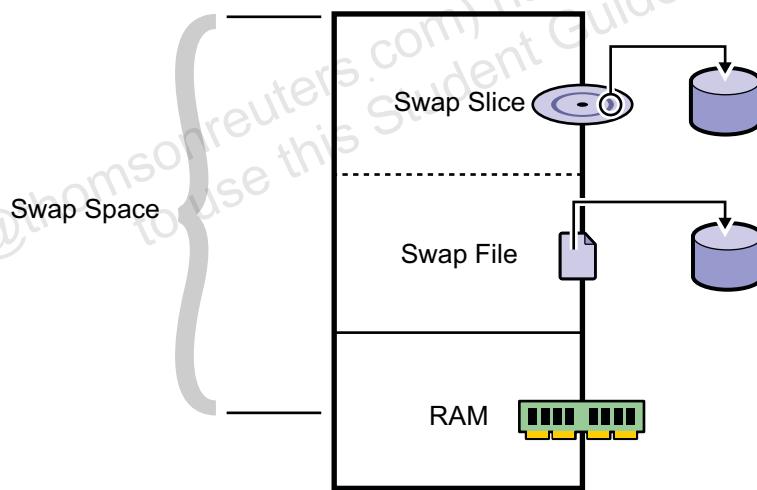


Figure 1-1 Swap Space

Paging

Paging is the transfer of selected memory pages between RAM and the swap areas. When you page private data to swap spaces, physical RAM is made available for other processes to use. If you need the pages that were paged out, you can retrieve them (page them in) from swap and map them back into physical memory. Moving these pages back into RAM might require more paging (page outs) of other process's pages to make room. Swapping is the movement of all modified data memory pages associated with a process, between RAM and a disk.

Use the `pagesize` command to display the size of a memory page in bytes. The default page size for SPARC-based systems running the Solaris 10 OS is 8192 bytes.

```
# pagesize  
8192
```

The default page size for Sun's x86-systems is 4096 bytes.

```
# pagesize  
4096
```

You can use the Multiple Page Size Support (MPSS) service to run legacy applications with larger memory page sizes. Using larger page sizes can significantly improve the performance of programs using large amounts of memory. Large pages must be mapped to addresses that are multiples of the page size. Use the `pagesize` command to display all supported page sizes.

For example, on SPARC-based systems:

```
# pagesize -a  
8192  
65536  
524288  
4194304
```

On x86-based systems:

```
# pagesize -a  
4096  
2097152
```

Swapping does not *typically* occur in the Solaris OS. The required amount of swap space varies from system to system. The amount of available swap space must satisfy two criteria:

- Swap space must be sufficient to supplement physical RAM to meet the needs of concurrently running processes.
- Swap space must be sufficient to hold a crash dump (in a single slice), unless `dumpadm(1m)` has been used to specify a dump device outside of swap space.

Configuring Swap Space

The swap command provides a method of adding, deleting, and monitoring the swap areas used by the kernel. Swap area changes made from the command line are not permanent and are lost after a reboot. To create permanent additions to the swap space, create an entry in the /etc/vfstab file. The entry in the /etc/vfstab file is added to the swap space at each reboot.

Displaying the Current Swap Configuration

Figure 1-2 shows the relationship between the used swap space, which consists of allocated and reserved swap spaces, and the available swap space.

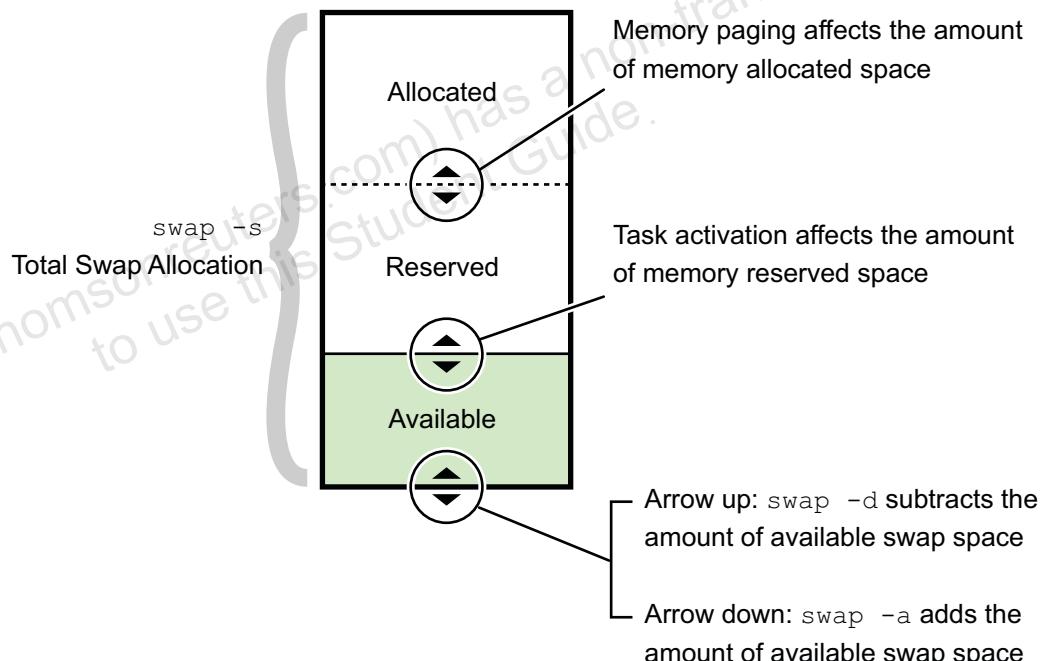


Figure 1-2 Swap Space Allocation

Configuring Swap Space

To view the current swap space allocation, complete the following steps:

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

```
# swap -s  
total: 41776k bytes allocated + 5312k reserved = 47088k used, 881536k  
available
```

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

```
# swap -l  
swapfile          dev  swaplo blocks   free  
/dev/dsk/c0t0d0s1    136,9      16 1048304 1048304
```

Note – There can be a discrepancy in available and free swap space size between the swap -s and swap -l outputs. The swap -s output does not take into account pre-allocated swap space that has not yet been used by a process.

The swap -l command reports values expressed in 512-byte blocks. The swap -s command reports values expressed in Kbytes.

The swaplo value reported by swap -l reflects the number of 512-byte blocks that match the system page size. The swaplo value prevents paging or swap operations in the disk blocks that might contain the disk VTOC and boot block.

Adding Swap Space

When the swap space requirements of the system exceed the current swap space available, you can use the following procedures to add additional swap space to your system.

Adding Swap Slices

To add a swap slice, complete the following steps:

1. Edit the /etc/vfstab file to add information describing the swap slice.

```
# vi /etc/vfstab
#device    device      mount      FS      fsck      mount      mount
#to mount   to fsck    point      type     pass      at boot    options
...
2. Add the following line to create the swap slice.
/dev/dsk/c1t3d0s1      -      -      swap      -      no      -
3. Use the swap -a command to add additional swap area.
# swap -a /dev/dsk/c1t3d0s1
```

 **Note** – When the system is subsequently rebooted, the new swap slice /dev/dsk/c1t3d0s1, is automatically included as part of the swap space as a result of adding the entry to the /etc/vfstab file.

Adding Swap Files

To add a swap file, complete the following steps:

1. Identify a file system that has adequate space to create an additional swap file, preferably on another drive.
2. Make a directory to hold the swap file.

```
# mkdir -p /usr/local/swap
3. Create a 20-Mbyte swap file named swapfile in the
/usr/local/swap directory.
# mkfile 20m /usr/local/swap/swapfile
```

Configuring Swap Space

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

```
# swap -a /usr/local/swap/swapfile
```

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

```
# swap -l
```

| swapfile | dev | swaplo | blocks | free |
|--------------------------|-------|--------|---------|---------|
| /dev/dsk/c0t0d0s1 | 136,9 | 16 | 1048304 | 1048304 |
| /usr/local/swap/swapfile | - | 16 | 40944 | 40944 |

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

```
# swap -s
```

total: 41672k bytes allocated + 5416k reserved = 47088k used, 901200k available

7. To use a swap file when the system is subsequently rebooted, add an entry for the swap file in the /etc/vfstab file.

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

| #device #to mount | device to fsck | mount point | FS type | fsck pass | mount at boot | mount options |
|--------------------------|-------------------|----------------|------------|--------------|------------------|------------------|
| ... | | | | | | |
| /usr/local/swap/swapfile | - | - | swap | - | no | - |

Removing Swap Space

If you no longer need the additional swap space, you can delete the swap space by removing the additional swap slices and swap files.

Removing Swap Slices

To remove a swap slice, complete the following steps:

1. Delete a swap slice from the current swap configuration.

```
# swap -d /dev/dsk/c1t3d0s1
```

2. To prevent the swap slice from being configured as part of the swap configuration during a reboot or change of run level, edit the /etc/vfstab file, and remove the swap slice entry from the file.

Removing Swap Files

To remove a swap file, complete the following steps:

1. Delete a swap file from the current swap configuration.

```
# swap -d /usr/local/swap/swapfile
```

2. Remove the file to free the disk space that it is occupying.

```
# rm /usr/local/swap/swapfile
```

3. To prevent the swap file from being configured as part of the swap configuration during a reboot or change of run level, edit the /etc/vfstab file, and remove the swap file entry.

Note – The output of the df -h command shows the space used by the swap file until it is removed.



What is a Core File?

When software encounters a fatal error, a core file is generated. For user processes, this is often referred to as a core dump. For the kernel, you will refer to the resulting core file as a crash dump.

The core file is an on-disk file that represents the memory image of the software that encountered the fatal error.

For user processes, the address space of the process contains various segments that include the text, data and stack of the process. The core file generated when the process encounters a fatal error is a snapshot of what the memory space of the process looked like when the error occurred.

The same is true for the kernel. A fatal error in the kernel software will result in a system crash, (typically a "panic"). The kernel core image is a snapshot of the kernel's memory space when the error occurred.

In both cases, a debugger must be used to examine the core file. For user processes, `dbx(1)` or `mdb(1)` can be used. For the kernel, `mdb(1)` is used to analyze the core file.

Why Generate a Core File?

Core files are generated so you can perform dump analysis (user processes), or crash dump analysis (the kernel) and determine the root-cause of the error so the problem can be corrected. Often times the only possible way to debug a problem that has caused a fatal software crash is an analysis of the core file.

Benefits of Generating Core Files

You generate core files so that you can effectively troubleshoot software problems to the root-cause of the crashes, and get problems corrected.

Drawbacks of Generating Core Files

The only drawback to generating core files is that these files can take up a great deal of disk space, and consequently would need to be managed.

Managing Crash Dump Behavior

If a fatal operating system error occurs, the operating system prints a message to the console, describing the error. The operating system then generates a crash dump by writing some of the contents of the physical memory to a predetermined dump device, which must be a local disk slice. You can configure the dump device by using the `dumpadm` command. After the operating system has written the crash dump to the dump device, the system reboots. The crash dump is saved for future analysis to help determine the cause of the fatal error.

The Crash Dump

If the Solaris OS kernel encounters a problem that might endanger the integrity of data or when the kernel encounters an unexpected hardware fault, the `panic` routine is executed. Despite its name, a system panic is a well-controlled event where memory contents are copied to a disk partition defined as a dump device. Whatever the cause, the crash dump itself provides valuable information to help your support engineer diagnose the problem.

When an operating system crashes, the `savecore` command is automatically executed during a boot. The `savecore` command retrieves the crash dump from the dump device and then writes the crash dump to a pair of files in your file system:

- The `savecore` command places kernel core information in the `/var/crash/nodename/vmcore.X` file, where `nodename` is the name returned by `uname -n`, and `X` is an integer identifying the dump.
- The `savecore` command places name list information and symbol table information in the `/var/crash/nodename/unix.X` file.



Note – Within the crash dump directory, a file named `bounds` is created. The `bounds` file holds a number that is used as a suffix for the next dump to be saved.

Together, these data files form the saved crash dump. You can use the `dumpadm` command to configure the location of the dump device and the `savecore` directory.

A dump device is usually disk space that is reserved to store system crash dump information. By default, a system's dump device is configured to be a swap slice. If possible, you should configure an alternate disk partition as a dedicated dump device to provide increased reliability for crash dumps and faster reboot time after a system failure.

Displaying the Current Dump Configuration

To view the current dump configuration, enter the `dumpadm` command without arguments, as shown in the following example:

```
# dumpadm
      Dump content: kernel pages
      Dump device: /dev/dsk/c0t0d0s1 (swap)
Savecore directory: /var/crash/sys-02
  Savecore enabled: yes
```

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/sys-02`
- The savecore command is set to run automatically on reboot

The following example shows that the current configuration is located in the `/etc/dumpadm.conf` file:

```
# 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_SAVDIR=/var/crash/sys-02
DUMPADM_CONTENT=kernel
DUMPADM_ENABLE=yes
```

Changing the Crash Dump Configuration

The dumpadm command manages the configuration of the operating system crash dump facility.



Note – Perform all modifications to the crash dump configuration by using the dumpadm command, rather than attempting to edit the /etc/dumpadm.conf file. Editing the file might result in an inconsistent system dump configuration.

The syntax of the dumpadm command is:

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

where:

- n Modifies the dump configuration so it does not run the savecore command automatically on reboot.
- u Forcibly updates the kernel dump configuration based on the contents of the /etc/dumpadm.conf file.
- y Modifies the dump configuration so that the savecore command is run automatically on reboot. This is the default.
- c *content-type* Specifies the contents of the crash dump. The *content-type* can be kernel, all, or curproc. The curproc content type includes the kernel memory pages and the memory pages of the currently executing process.
- d *dump-device* Modifies the dump configuration to use the specified dump device. The dump device can be an absolute path name or swap.

Managing Crash Dump Behavior

-m *mink*
-m *mirm*
-m *min%*

Creates a *minfree* file in the current *savecore-dir* directory indicating that the *savecore* command should maintain at least the specified amount of free space in the file system in which the *savecore-dir* 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 that the *minfree* value is computed as the specified percentage of the total, current size of the file system that contains the *savecore-dir* directory.

-r *root-dir*

Specifies an alternative root directory relative to which the *dumpadm* command should create files. If the **-r** argument is not specified, the default root directory “*/*” is used.

-s *savecore-dir*

Modifies the dump configuration to use the specified directory to save files written by the *savecore* command. The default *savecore-dir* directory is */var/crash/hostname*, where *hostname* is the output of the *uname -n* command.

Managing Core File Behavior

When a process terminates abnormally, it typically produces a core file. You can use the `coreadm` command to specify the name or location of core files produced by abnormally terminated processes.

Core Files

A core file is a point-in-time copy (snapshot) of the RAM allocated to a process. The copy is written to a more permanent medium, such as a hard disk. A core file is useful in analyzing why a particular program crashed.

A core file is also a disk copy of the address space of a process, at a certain point-in-time. This information identifies items, such as the task name, task owner, priority, and instruction queue, in execution at the time that the core file was created.

When a core file occurs, the operating system generates two possible copies of the core files, one copy known as the global core file and the other copy known as 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 files with these permissions.

Ordinary per-process core files are created in mode 600 under the credentials of the process. The owner of the process can examine files with these permissions.

Displaying the Current Core File Configuration

You use the coreadm command without arguments to display the current configuration.

```
# coreadm
1   global core file pattern:
2   global core file content: default
3       init core file pattern: core
4       init core file content: default
5           global core dumps: disabled
6           per-process core dumps: enabled
7           global setid core dumps: disabled
8 per-process setid core dumps: disabled
9   global core dump logging: disabled
```



Note – The line numbers in the example are not part of the configuration. They are part of the example only to assist with the following description of the file.

Line 1 of the output identifies the name to use for core files placed in a global directory.

Line 2 of the output identifies that the content of core files is the default setting. The resultant core file contains all the process information pertinent to debugging.

Line 3 of the output identifies the default name that per-process core files must use. This name is set for the `init` process, meaning it is inherited by all other processes on the system.

Line 4 of the output indicates that the `init` core file content is the default content structure.

Line 5 indicates that global core files are disabled.

Line 6 indicates that core file generation in the current working directory of a process is enabled.

Line 7 indicates that generation of global core files with `setuid` or `setgid` permissions are disabled.

Line 8 indicates that generation of per process core files with setuid or setgid permissions are disabled.

Line 9 indicates if global core dump logging is enabled.



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

By viewing the /etc/coreadm.conf file, you can verify the same configuration parameters that were displayed with the coreadm command.

```
# 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_GLOB_CONTENT=default
COREADM_INIT_PATTERN=core
COREADM_INIT_CONTENT=default
COREADM_GLOB_ENABLED=no
COREADM_PROC_ENABLED=yes
COREADM_GLOB_SETID_ENABLED=no
COREADM_PROC_SETID_ENABLED=no
COREADM_GLOB_LOG_ENABLED=no
```

Changing the Core File Configuration

The coreadm command allows you to control core file generation behavior. 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. The flexibility of this configuration makes it easier to track problems by examining the core files in a specific directory whenever a process or daemon terminates abnormally. This flexibility also makes it easy to locate and remove core files on a system.



Note – You should make all modifications to the coreadm configuration at the command line by using the coreadm command instead of editing the /etc/coreadm.conf file.

You can enable or disable two configurable core file paths, per-process and global, separately. 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/core directory.



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

Users can run the coreadm command with the -p option to specify the file name pattern for the operating system to use when generating a per-process core file.

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

Only the root user can run the following coreadm command options to configure system-wide core file options.

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



The coreadm Command Options

The following are some options to the coreadm command.

Note – A regular user can only use the `-p` option, the superuser can use all options.

| | |
|-------------------------|--|
| <code>-i pattern</code> | Sets the per-process core file name pattern from <code>init</code> to <code>pattern</code> . This option is the same as the <code>coreadm -p pattern 1</code> command, except that the setting is persistent after a reboot. |
| <code>-e option</code> | Enables the specified core file option, where <code>option</code> is: <ul style="list-style-type: none"> • <code>global</code> – Enables core dumps by using the global core pattern. • <code>process</code> – Enables core dumps by using the per-process core pattern. • <code>global-setid</code> – Enables setid core dumps by using the global core pattern. • <code>proc-setid</code> – Enables setid core dumps by using the per-process core pattern. • <code>log</code> – Generates a <code>syslog (3)</code> message when a user attempts to generate a global core file. |
| <code>-d option</code> | Disables the specified core file option; see the <code>-e option</code> for descriptions of possible options. You can specify multiple <code>-e</code> and <code>-d</code> options by using the command line. |
| <code>-u</code> | Updates system-wide core file options from the contents of the configuration file <code>/etc/coreadm.conf</code> . 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. |
| <code>-g pattern</code> | Sets the global core file name pattern to <code>pattern</code> . The pattern must start with a forward slash (/), and can contain any of the special embedded variables described in Table 1-1 on page 1-22. |

- p *pattern*** Sets the per-process core file name pattern to *pattern* for each of the specified process IDs (PIDs). The pattern can contain any of the special embedded variables described in Table 1-1 and does not have to begin with a forward slash (/). If *pattern* does not begin with “/”, it is evaluated relative to the current directory in effect when the process generates a core file.
- A non-privileged user can only apply the -p option to processes owned by that user. The superuser can apply the -p option to any process.
- G *content*** Set the global core file content. You specify *content* by using pattern options listed in Table 1-1.

A core file named *pattern* is a file system path name with embedded variables. The embedded variables are specified with a leading percent (%) character. The operating system expands these variables from values in effect when the operating system generates a core file. The possible variables are listed in Table 1-1.

Table 1-1 Pattern Options for the coreadm Command

| Option | Meaning |
|--------|--|
| %p | PID |
| %u | Effective user ID (EUID) |
| %g | Effective group ID (EGID) |
| %f | Executable file name |
| %n | System node name (uname -n) |
| %m | Machine hardware name (uname -m) |
| %t | The time in seconds since midnight January 1, 1970 |
| %d | Executable file directory/name |
| %z | Zonename |
| %% | Literal% |

Table 1-2 shows the pattern options for the global core file content.

Table 1-2 Pattern Options for the Global Core File Content

| Option | Meaning |
|--------|---|
| anon | Anonymous private mappings, including thread stacks that are not main thread stacks |
| ctf | CTF type information sections for loaded object files |
| data | Writable private file mappings |
| dism | DISM mappings |
| heap | Process heap |
| ism | ISM mappings |
| rodata | Read-only file mappings |
| shanon | Anonymous shared mappings |
| shfile | Shared mappings that are backed by files |
| shm | System V shared memory |
| stack | Process stack |
| syntab | Symbol table sections for loaded object |
| text | Readable and executable file mappings |

Examples of 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 file, the following entry sets the core file name pattern for all processes run during the login session:

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

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



Example 2 – Dumping a User's Core Files Into a Subdirectory

The following command places all of the user's core files into the corefiles subdirectory of the user's home directory, differentiated by the system node name. This example is useful for users who use many different systems, but share a single home directory across multiple systems.

```
$ 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 core file that is created:

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

For example, the core file name pattern /var/core/core.%f.%p causes the xyz program with PID 1234 to generate the core file /var/core/core.xyz.1234.

Note – In the above coreadm examples, the corefiles and core directories must be created manually. The coreadm command does not create them automatically.



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

```
# coreadm
global core file pattern: /var/core/core.%f.%p
global core file content: default
init core file pattern: core
init core file content: default
    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 PIDs

Running the coreadm command with a list of PIDs reports each process's per-process core file name pattern, for example:

```
# coreadm 228 507
228:   core   default
507:   /usr/local/swap/corefiles/%n.%f.%p   default
```

Only the owner of a process or the superuser can query a process by using the coreadm command with a list of PIDs.

Example 5 – Setting up the System to Produce Core Files in the Global Repository Only if the Executables Were Run From /usr/bin or /usr/sbin

```
# mkdir -p /var/core/usr/bin
# mkdir -p /var/core/usr/sbin
# coreadm -G all -g /var/core/%d/%f%p%n
```

When using the *all* option in the previous command, examples of the core file content include:

- anon = anonymous private maps
- data = writable private file mapping
- stack = process stack
- syms = symbol table sections for loaded object files

Exercise 1: Managing swap Utility Configuration

In this exercise, you add and remove a swap space.

Preparation

Before you perform the following task, here is an overview on how to access your lab equipment.

Sun uses the Sun Secure Global Desktop (SGD) to provide you with access to the remote lab environment. To access the SGD, open the following path in your Web browser:

<http://rldc.sun.net>

Click the Log In link and enter the user name and password provided by your instructor. Figure 1-3 shows a typical SGD session.

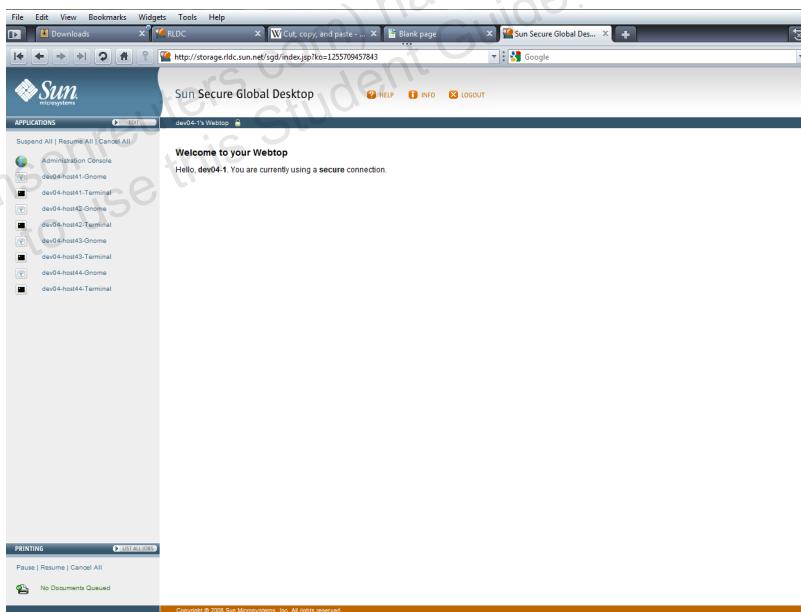


Figure 1-3 Sun Secure Global Desktop

The left panel contains tools that allow you to access your lab systems. The *Gnome* tool opens a remote desktop on the lab system. The *Terminal* tool opens a terminal window on the lab system. The *Console* tool opens a remote console on the lab system. The *Administration Console* is for shadowing your lab partner.

To prepare for this exercise:

- Each student should unconfigure the additional swap space before exiting the lab exercise.
- All students use disk slice 1 of the spare disk on their systems for this exercise.
- This exercise assumes students know how to use the format utility to create Solaris fdisk partitions, and disk slices.
- Device examples in this exercises are from a SPARC-based system. Be certain to use device names that are correct for the system architecture you are using.
- On x86/x64 systems, use the fdisk menu in the format utility to create one Solaris fdisk partition that uses the entire spare disk. Be certain that the Solaris fdisk partition you create on the spare disk *is not marked* as the active partition.



Note – The actual swap statistics will vary depending on the configuration of each system.

Partition the spare disk, or the Solaris fdisk partition on the spare disk, using the information in Table 1-3.

Table 1-3 Partition Information

| Slice | Size | Use |
|-------|---|-------------|
| 0 | 2048 Mbytes | Swap /dump |
| 1 | 2048 Mbytes | Swap /dump |
| 3 | 16 Mbytes | Unassigned |
| 4 | 16 Mbytes | Unassigned |
| 5 | 10240 Mbytes (or remainder of disk) | File system |
| 6 | 0 Mbytes | Unassigned |
| 7 | 0 Mbytes | Unassigned |

Tasks

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Run the `swap -s` command.
 - a. What is the total number of bytes actually allocated and currently in use?
 - _____
 - b. What is the number of bytes allocated and not currently in use, but reserved by processes for possible future use?
 - _____
 - c. What is the total amount of swap space, both allocated and reserved?
 - _____
 - d. What is the total swap space currently available for future reservation and allocation?
 - _____
3. Run the `swap -l` command.

List the physical swap device configured on your system.

- a. How much total swap space is in the listed swap device?
- _____
- b. How much space is available for the listed device?
- _____
4. Create the `/usr/local/swap` directory if it does not already exist.
5. Create a 20-Mbyte swap file in the `/usr/local/swap` directory and add it to the system swap space.
6. Use the `swap -l` command to verify that the new swap space is available.
7. Use the `swap -s` command to verify that the new swap space is available.

How does the output differ between the `swap -l` command and the `swap -s` command?

8. Remove the swap file created in step 5.
9. Use the swap utility to verify that the swap space is no longer available.
10. Add partition 1 of your second disk to your existing swap space and use the correct device name for your system.

11. Verify that the new swap space has been added.

12. Add the new swap partition to the /etc/vfstab file to make the partition permanent.

To verify this change, you must reboot the system.

13. After the reboot, verify that the additional swap space exists by using the swap utility.

Is the newly listed swap partition the same as the one you added to the /etc/vfstab file?

-
14. Verify the additional swap space exists using the df -h command.

Why is the newly created swap space listed in the /etc/vfstab file not listed in the output of the df -h command?

15. Remove the additional swap space using the swap -d command to return the system to its initial swap configuration.

16. Remove the additional swap space entry from the /etc/vfstab file so that the system maintains its initial swap configuration after rebooting.

17. Verify that the additional swap space was unconfigured using the swap -l command.

Exercise 2: Collecting the Crash Dump and Core Dump

In this exercise, you configure crash dumps and core files.

Preparation

This exercise requires specific disk partitions on the spare disk. The required partitions may already exist as a result of previously-run exercises.

- This exercise assumes students know how to use the format utility to create Solaris fdisk partitions, and disk slices, if required.
- Device examples in this exercises are from an x86/x64-based system. Be certain to use device names that are correct for the system architecture you are using.
- On x86/x64 systems, verify that the spare disk contains one Solaris fdisk partition that uses the entire disk. If not, use the fdisk menu in the format utility to create one Solaris fdisk partition that uses the entire spare disk. Be certain that the Solaris fdisk partition you create on the spare disk *is not marked* as the active partition.

Verify that the spare disk has the partitions defined in Table 1-4. If it does not, partition the spare disk, or the Solaris fdisk partition on the spare disk, using the information in Table 1-4.

Table 1-4 Partition Information

| Slice | Size | Use |
|-------|---|-------------|
| 0 | 2048 Mbytes | Swap /dump |
| 1 | 2048 Mbytes | Swap /dump |
| 3 | 16 Mbytes | Unassigned |
| 4 | 16 Mbytes | Unassigned |
| 5 | 10240 Mbytes (or remainder of disk) | File system |
| 6 | 0 Mbytes | Unassigned |
| 7 | 0 Mbytes | Unassigned |

Task 1 – Using the `dumpadm` Command to Display the Core File Directory Location

Complete the following steps:

1. Use the `dumpadm` command with no arguments to view the current dump configuration.
2. Record the configuration parameters that the `dumpadm` command displayed in the previous step:

Dump content:

Dump device:

The savecore directory:

Is savecore enabled?

3. Use the `dumpadm` command to change the dump device to the second disk drive slice 5.
4. Run the `sync` command to flush all previously unwritten system buffers out to disk, ensuring that all file modifications up to that point will be saved.
5. Force the kernel to save a live snapshot of the running system and write out a new set of crash dump files by using the `savecore -L` command.
6. Make sure the crash dump succeeded by using the `file` command to identify the files in the savecore directory.
7. Use the `dumpadm` command to set the dump device to its original value.

Task 2 – Using the `coreadm` Command to Configure Core File Storage Locations

Complete the following steps:

1. Use the `coreadm` command to display the default core file configuration.
2. Create a directory to hold core files, and enable a global core file path that uses the directory you created.
3. Turn on global core dump logging to generate a message when the system creates a global core file.

Exercise 2: Collecting the Crash Dump and Core Dump

4. Display the core file configuration information to verify your changes.
5. Open another terminal window. In the new terminal window, create a new directory named /var/tmp/dir, and change directory to it.
6. Verify that your current working directory is /var/tmp/dir.
7. Run the ps command to display the PID of the shell associated with the new terminal window, and send a SIGFPE signal (Signal 8) to the shell by using the kill command. (SIGFPE causes a core file to be created.)

Note – The kill -8 command terminates the shell and the terminal window in which it is executed.

8. 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 that the core file is from the old shell.
9. Use the ls command to check for a core file in the /var/core directory.
10. Observe the messages generated in the console window and the /var/adm/messages file due to coreadm logging being enabled.



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 1 Solutions: Managing swap Utility Configuration

In this exercise you add and remove a swap space.

Preparation

Before you perform the following task, here is an overview on how to access your lab equipment.

Sun uses the Sun Secure Global Desktop (SGD) to provide you with access to the remote lab environment. To access the SGD, open the following path in your Web browser:

<http://rldc.sun.net>

Click the Log In link and enter the user name and password provided by your instructor. Figure 1-3 shows a typical SGD session.

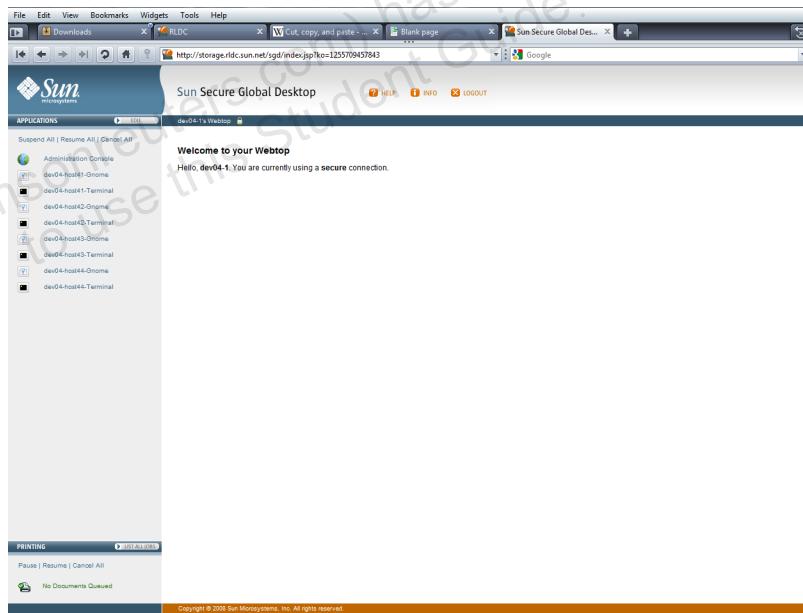


Figure 1-4 Sun Secure Global Desktop

The left panel contains tools that allow you to access your lab systems. The *Gnome* tool opens a remote desktop on the lab system. The *Terminal* tool opens a terminal window on the lab system. The *Console* tool opens a remote console on the lab system. The *Administration Console* is for shadowing your lab partner.

To prepare for this exercise:

- Each student should unconfigure the additional swap space before exiting the lab exercise.
- All students use disk slice 1 of the spare disk on their systems for this exercise.
- This exercise assumes students know how to use the format utility to create Solaris fdisk partitions, and disk slices.
- Device examples in this exercises are from a SPARC-based system. Be certain to use device names that are correct for the system architecture you are using.
- On x86/x64 systems, use the fdisk menu in the format utility to create one Solaris fdisk partition that uses the entire spare disk. Be certain that the Solaris fdisk partition you create on the spare disk *is not marked* as the active partition.



Note – The actual swap statistics will vary depending on the configuration of each system.

Partition the spare disk, or the Solaris fdisk partition on the spare disk, using the information in Table 1-5.

Table 1-5 Partition Information

| Slice | Size | Use |
|-------|---|-------------|
| 0 | 2048 Mbytes | Swap /dump |
| 1 | 2048 Mbytes | Swap /dump |
| 3 | 16 Mbytes | Unassigned |
| 4 | 16 Mbytes | Unassigned |
| 5 | 10240 Mbytes (or remainder of disk) | File system |
| 6 | 0 Mbytes | Unassigned |
| 7 | 0 Mbytes | Unassigned |

Tasks and Solutions

This section describes the steps you must perform, and lists the solutions to these steps. Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Run the `swap -s` command.

```
# swap -s
total: 49024k bytes allocated + 4704k reserved = 53728k used, 875280k
available
```

- a. What is the total number of bytes actually allocated and currently in use?
49,024 kilobytes (Kbytes) in this example.
- b. What is the number of bytes allocated and not currently in use, but reserved by processes for possible future use?
4,704 Kbytes in this example.
- c. What is the total amount of swap space, both allocated and reserved?
53,728 Kbytes in this example.
- d. What is the total swap space currently available for future reservation and allocation?
875,280 Kbytes in this example.

3. Run the `swap -l` command.

```
# swap -l
swapfile          dev  swaplo  blocks   free
/dev/dsk/c0t0d0s1  136,1      16  1049312  1049312
```

List the physical swap device configured on your system.

/dev/dsk/c0t0d0s1 in this example.

- a. How much total swap space is in the listed swap device?

1,049,312 blocks in this example.

- b. How much space is available for the listed device?

1,049,312 blocks in this example.

4. Create the /usr/local/swap directory if it does not already exist.

```
# mkdir -p /usr/local/swap
```

5. Create a 20-Mbyte swap file in the /usr/local/swap directory and add it to the system swap space.

```
# mkfile 20m /usr/local/swap/swapfile
```

```
# swap -a /usr/local/swap/swapfile
```

6. Use the swap -l command to verify that the new swap space is available.

```
# swap -l
```

| swapfile | dev | swaplo | blocks | free |
|--------------------------|-------|--------|---------|---------|
| /dev/dsk/c0t0d0s1 | 136,1 | 16 | 1049312 | 1049312 |
| /usr/local/swap/swapfile | - | 16 | 40944 | 40944 |

7. Use the swap -s command to verify that the new swap space is available.

```
# swap -s
```

| total: | bytes allocated | reserved | = | used, | available |
|--------|-----------------|----------|---|--------|-----------|
| 49104k | + 4624k | | = | 53728k | 895040k |

How does the output differ between the swap -l command and the swap -s command?

The swap -l command output is a listing of each space, whereas the swap -s command output only produces a cumulative report.

8. Remove the swap file created in step 5.

```
# swap -d /usr/local/swap/swapfile
```

```
# rm /usr/local/swap/swapfile
```

9. Use the swap utility to verify that the swap space is no longer available.

```
# swap -l
```

| swapfile | dev | swaplo | blocks | free |
|-------------------|-------|--------|---------|---------|
| /dev/dsk/c0t0d0s1 | 136,1 | 16 | 1049312 | 1049312 |

```
# swap -s
```

| total: | bytes allocated | reserved | = | used, | available |
|--------|-----------------|----------|---|--------|-----------|
| 49088k | + 4640k | | = | 53728k | 874568k |

Exercise 1 Solutions: Managing swap Utility Configuration

10. Add partition 1 of your second disk to your existing swap space and use the correct device name for your system.

```
# swap -a /dev/dsk/c0t1d0s1
```

11. Verify that the new swap space has been added.

```
# swap -l
```

| swapfile | dev | swaplo | blocks | free |
|-------------------|-------|--------|---------|---------|
| /dev/dsk/c0t0d0s1 | 136,1 | 16 | 1049312 | 1049312 |
| /dev/dsk/c0t1d0s1 | 136,9 | 16 | 1049312 | 1049312 |

```
# swap -s
```

```
total: 49120k bytes allocated + 4608k reserved = 53728k used, 1399224k available
```

12. Add the new swap partition to the /etc/vfstab file to make the partition permanent.

To verify this change, you must reboot the system.

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

(add entry that matches your configuration)

```
/dev/dsk/c0t1d0s1 - - swap - no -
```

```
# init 6
```

13. After the reboot, verify that the additional swap space exists by using the swap utility.

```
# swap -l
```

| swapfile | dev | swaplo | blocks | free |
|-------------------|-------|--------|---------|---------|
| /dev/dsk/c0t0d0s1 | 136,1 | 16 | 1049312 | 1049312 |
| /dev/dsk/c0t1d0s1 | 136,9 | 16 | 1049312 | 1049312 |

Is the newly listed swap partition the same as the one you added to the /etc/vfstab file?

Yes.

14. Verify the additional swap space exists using the df -h command.

Why is the newly created swap space listed in the /etc/vfstab file not listed in the output of the df -h command?

```
# df -h
```

| Filesystem | size | used | avail | capacity | Mounted on |
|-------------------|------|------|-------|----------|-------------------|
| /dev/dsk/c0t0d0s0 | 470M | 194M | 229M | 46% | / |
| /devices | 0K | 0K | 0K | 0% | /devices |
| ctfs | 0K | 0K | 0K | 0% | /system/contract |
| proc | 0K | 0K | 0K | 0% | /proc |
| mnttab | 0K | 0K | 0K | 0% | /etc/mnttab |
| swap | 1.3G | 968K | 1.3G | 1% | /etc/svc/volatile |
| objfs | 0K | 0K | 0K | 0% | /system/object |
| /dev/dsk/c0t0d0s6 | 4.8G | 2.9G | 1.9G | 61% | /usr |

```

fd          0K    0K    0K    0% /dev/fd
/dev/dsk/c0t0d0s3 479M   57M   375M  14% /var
swap        1.3G   0K   1.3G   0% /tmp
swap        1.3G   40K   1.3G   1% /var/run
/dev/dsk/c0t0d0s7 2.1G   2.1M   2.0G  1% /export
#

```

The df -h output does not produce an entry for the additional swap devices. However, the added swap space is reflected in the total swap space.

15. Remove the additional swap space using the swap -d command to return the system to its initial swap configuration.

```
# swap -d /dev/dsk/c0t1d0s1
```

16. Remove the additional swap space entry from the /etc/vfstab file so that the system maintains its initial swap configuration after rebooting.

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

17. Verify that the additional swap space was unconfigured using the swap -l command.

```
# swap -l
```

| swapfile | dev | swaplo | blocks | free |
|-------------------|-------|--------|---------|---------|
| /dev/dsk/c0t0d0s1 | 136,1 | 16 | 1049312 | 1049312 |

Exercise 2 Solutions: Collecting the Crash Dump and Core Dump

In this exercise, you configure crash dumps and core files.

Preparation

This exercise requires specific disk partitions on the spare disk. The required partitions may already exist as a result of previously-run exercises.

- This exercise assumes students know how to use the `format` utility to create Solaris `fdisk` partitions, and disk slices, if required.
- Device examples in this exercises are from an x86/x64-based system. Be certain to use device names that are correct for the system architecture you are using.
- On x86/x64 systems, verify that the spare disk contains one Solaris `fdisk` partition that uses the entire disk. If not, use the `fdisk` menu in the `format` utility to create one Solaris `fdisk` partition that uses the entire spare disk. Be certain that the Solaris `fdisk` partition you create on the spare disk *is not marked* as the active partition.

Verify that the spare disk has the partitions defined in Table 1-5. If it does not, partition the spare disk, or the Solaris `fdisk` partition on the spare disk, using the information in Table 1-6.

Table 1-6 Partition Information

| Slice | Size | Use |
|-------|---|-------------|
| 0 | 2048 Mbytes | Swap /dump |
| 1 | 2048 Mbytes | Swap /dump |
| 3 | 16 Mbytes | Unassigned |
| 4 | 16 Mbytes | Unassigned |
| 5 | 10240 Mbytes (or remainder of disk) | File system |
| 6 | 0 Mbytes | Unassigned |
| 7 | 0 Mbytes | Unassigned |

Task 1 – Using the dumpadm Command to Display the Core File Directory Location

Complete the following steps:

1. Use the `dumpadm` command with no arguments to view the current dump configuration.

```
# dumpadm
Dump content: kernel pages
Dump device: /dev/dsk/c1d0s1 (swap)
Savecore directory: /var/crash/sys01
Savecore enabled: yes
#
```

2. Record the configuration parameters that the `dumpadm` command displayed in the previous step:

```
Dump content: kernel pages
Dump device: /dev/dsk/c1d0s1 (swap)
The savecore directory: /var/crash/sys01
Is savecore enabled? yes
```

3. Use the `dumpadm` command to change the dump device to the second disk drive slice 5.

```
# dumpadm -d /dev/dsk/c2d0s5
Dump content: kernel pages
Dump device: /dev/dsk/c2d0s5 (dedicated)
Savecore directory: /var/crash/sys01
Savecore enabled: yes
#
```

4. Run the `sync` command to flush all previously unwritten system buffers out to disk, ensuring that all file modifications up to that point will be saved.

```
# sync
```

Exercise 2 Solutions: Collecting the Crash Dump and Core Dump

5. Force the kernel to save a live snapshot of the running system and write out a new set of crash dump files by using the savecore -L command.

```
# savecore -L
dumping to /dev/dsk/c2d0s5, offset 65536, content: kernel
100% done: 63380 pages dumped, compression ratio 4.28, dump succeeded
System dump time: Fri Apr 20 13:40:20 2007
Constructing namelist /var/crash/sys01/unix.0
Constructing corefile /var/crash/sys01/vmcore.0
100% done: 63380 of 63380 pages saved
#
```

6. Make sure the crash dump succeeded by using the file command to identify the files in the savecore directory.

The output shown should be similar to the following if you are using a x86/x64-based system:

```
# cd /var/crash/sys01
# ls
bounds      unix.0      vmcore.0
# file vmcore.0
vmcore.0:           SunOS 5.10 Generic_118855-33 64-bit Intel crash dump from
'
#
#
```

The output shown should be similar to the following if you are using a SPARC-based system:

```
vmcore.0:SunOS 5.10 Generic_137138-09 64-bit SPARC crash dump from ''
```

7. Use the dumpadm command to set the dump device to its original value.

```
# dumpadm -d /dev/dsk/c1d0s1
Dump content: kernel pages
Dump device: /dev/dsk/c1d0s1 (swap)
Savecore directory: /var/crash/sys01
Savecore enabled: yes
#
```

Task 2 – Using the coreadm Command to Configure Core File Storage Locations

Complete the following steps:

1. Use the coreadm command to display the default core file configuration.

```
# coreadm
global core file pattern:
global core file content: default
init core file pattern: core
init core file content: default
    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
#
```

2. Create a directory to hold core files, and enable a global core file path that uses the directory you created.

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

3. Turn on global core dump logging to generate a message when the system creates a global core file.

```
# coreadm -e log
```

4. Display the core file configuration information to verify your changes.

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

Exercise 2 Solutions: Collecting the Crash Dump and Core Dump

5. Open another terminal window. In the new terminal window, create a new directory named /var/tmp/dir, and change directory to it.

```
# mkdir /var/tmp/dir  
# cd /var/tmp/dir
```

6. Verify that your current working directory is /var/tmp/dir.

```
# pwd  
/var/tmp/dir
```

7. Run the ps command to display the PID of the shell associated with the new terminal window, and send a SIGFPE signal (Signal 8) to the shell by using the kill command. (SIGFPE causes a core file to be created.)

```
# ps  
PID TTY TIME CMD  
1204 pts/2 0:00 ksh  
1208 pts/2 0:00 ps  
# kill -8 1204
```

Note – The kill -8 command terminates the shell and the terminal window in which it is executed.

8. 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 that the core file is from the old shell.

```
# cd /var/tmp/dir  
# ls  
core  
# file core  
core: ELF 32-bit LSB core file 80386 Version 1, from 'ksh'  
A SPARC system displays a result similar to the following:  
core: ELF 32-bit MSB core file SPARC Version 1, from 'ksh'
```

9. Use the ls command to check for a core file in the /var/core directory.

```
# ls /var/core  
core.sh.1204
```

10. Observe the messages generated in the console window and the /var/adm/messages file due to coreadm logging being enabled.

```
# tail /var/adm/messages
```

```
...
```

```
Apr 20 13:58:46 sys01 genunix: [ID 603404 kern.notice] NOTICE: core_log:  
sh[1204] core dumped: /var/core/core.sh.1204
```

Module 2

Configuring NFS

Objectives

The Network File System (NFS) is a client-server service that lets users view, store, and update files on a remote computer as though they were on their own local computer.

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

- Describe the benefits of NFS
- Describe the fundamentals of the NFS distributed file system
- Manage an NFS server
- Manage an NFS client
- Enable the NFS server logging
- Troubleshoot NFS errors

Introducing the Benefits of NFS

The NFS service enables computers of different architectures running different operating systems to share file systems across a network.

You can implement the NFS environment on different operating systems (OS) because NFS defines an abstract model of a file system. Each operating system applies the NFS model to its file system semantics. For example, NFS file system operations, such as reading and writing, work as if they were accessing a local file.

Some of the benefits of the NFS service are that it:

- Allows multiple computers to use the same files, because all users on the network can access the same data
- Reduces storage costs by sharing applications on computers instead of allocating local disk space for each user application
- Provides data consistency and reliability, because all users can read the same set of files
- Supports heterogeneous environments, including those found on a personal computer (PC)
- Reduces system administration overhead

Note – The NFS was developed by Sun Microsystems and is recognized as a file server standard. Its protocol uses the Remote Procedure Call (RPC) method of communication between computers on the Internet. Other sources of information for NFS are found at
<http://docs.sun.com/app/docs/prod/solaris.10#hic>, and
<http://www.citi.umich.edu> for information about porting NFSV4 to Linux.





Benefits of Centralized File Access

The NFS service lets you share a whole or partial directory tree or a file hierarchy. Instead of placing copies of commonly used files on every system, the NFS service enables you to place one copy of the files on one computer's hard disk. All other systems can then access the files across the network. When using the NFS service, remote file systems are almost indistinguishable from local file systems.

Note – In most UNIX environments, a file hierarchy that can be shared corresponds to a file system. Because NFS functions across operating systems, and the concept of a file system might be meaningless in non-UNIX environments, the use of the term file system refers to a file hierarchy that can be shared and mounted over NFS environments.

The files are centrally located, making the same files accessible to many users and systems simultaneously. This accessibility feature is useful when giving a user access to a single home directory across multiple systems or when providing access to various applications.

Benefits of Common Software Access

Systems can share one or more centrally located software packages, reducing the disk space requirements for individual systems.

Remote file sharing is almost transparent to the user and to any application, because these resources appear as if they exist on the local system.

Introducing the Fundamentals of the NFS Distributed File System

The Solaris 10 OS supports the sharing of remote file resources and presents them to users as if they were local files and directories. The primary distributed file system (DFS) type supported by the Solaris 10 OS is NFS.

The NFS environment contains the following components:

- NFS server
- NFS client

The Solaris 10 OS supports versions 2, 3, and 4 NFS simultaneously. The default is to use NFSv4 software when sharing a directory or accessing a shared file. Version-related checks are applied whenever a client host attempts to access a server's file share. If all hosts in the network are installed with Solaris 10 OS, then all hosts should, by default, use the NFSv4 protocols.

NFS Server

The NFS server contains file resources shared with other systems on the network. A computer acts as a server when it makes files and directories on its hard disk available to the other computers on the network.

“NFS Server Configuration” shows how files and directories on an NFS server are made available to NFS clients. The NFS server is sharing the /export/rdbms directory over NFS, as shown in Figure 2-1.

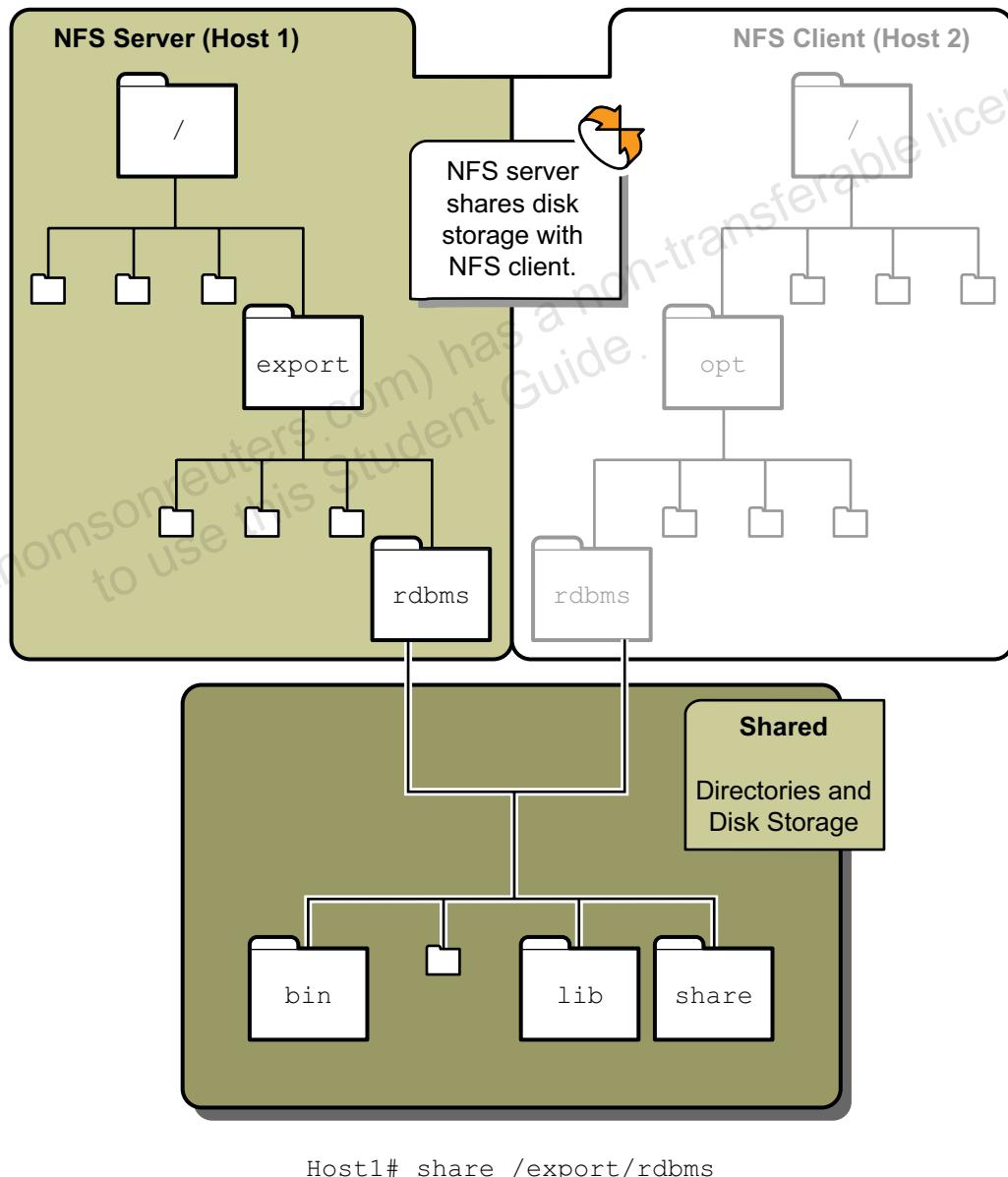
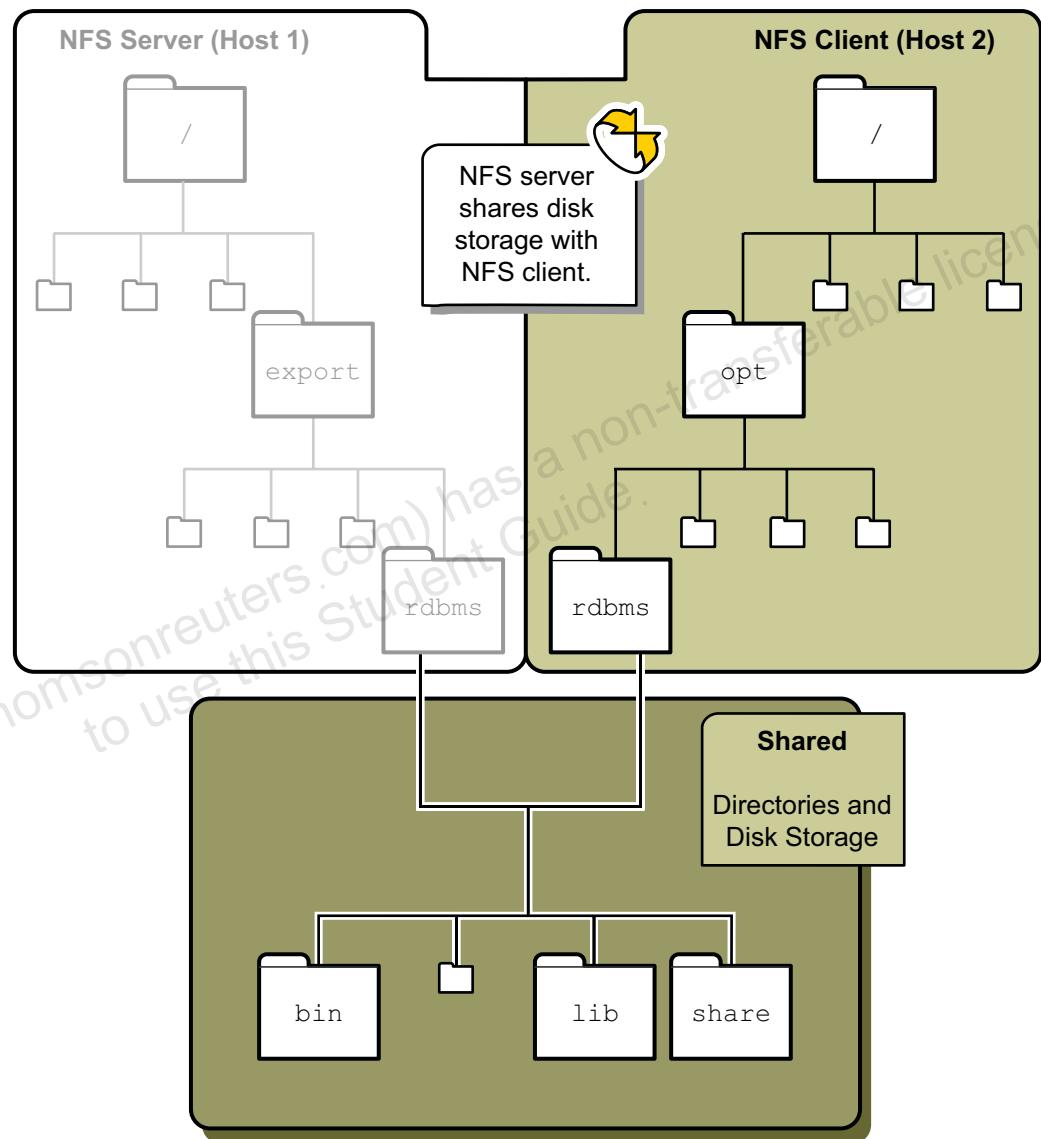


Figure 2-1 NFS Server Configuration

NFS Client

The NFS client system mounts file resources shared over the network and presents the file resources to users as if they were local files.

“NFS Client Configuration” shows how an NFS client uses the files and directories shared by an NFS server. The /export/rdbms directory, shared by the NFS server, is mounted on the NFS client on the /opt/rdbms mount point. The resource mount point exists on the NFS client, and the NFS server shares the file resources with other computers on the network, as shown in Figure 2-2.



```
Host2# mount Host1:/export/rdbms /opt/rdbms
```

Figure 2-2 NFS Client Configuration

Managing an NFS Server

You use NFS server files, NFS server daemons, and NFS server commands to configure and manage an NFS server.

The NFS Server Files

You need several files to support NFS server activities on any computer. Table 2-1 lists these files and their functions.

Table 2-1 NFS Server Files

| File | Description |
|----------------------|--|
| /etc/dfs/dfstab | Lists the local resources to share at boot time. |
| /etc/dfs/sharetab | Lists the local resources currently being shared by the NFS server. Do not edit this file. |
| /etc/dfs/fstypes | Lists the default file system types for remote file systems. |
| /etc/rmtab | Lists file systems remotely mounted by NFS clients. Do not edit this file. |
| /etc/nfs/nfslog.conf | Lists information defining the location of configuration logs used for NFS server logging. |
| /etc/default/nfslogd | Lists configuration information describing the behavior of the nfslogd daemon for NFSv2/3. |
| /etc/default/nfs | Contains parameter values for NFS protocols and NFS daemons. |

The /etc/dfs/dfstab File

The /etc/dfs/dfstab file contains the commands that share local directories. 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 'svcadm enable network/nfs/server' 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

share -F nfs -o ro -d "Shared data files" /usr/local/data
share -F nfs -o rw,root=sys-01 -d "Database files" /rdbms_files
```



Note – If the svc:/network/nfs/server service does not find any share commands in the /etc/dfs/dfstab file, it does not start the NFS server daemons.

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

- The system enters the multi-user-server milestone.
- The superuser runs the shareall command. The NFS daemons must be running to share directories.
- The superuser enables the svc:/network/nfs/server service.

The /etc/dfs/sharetab File

The /etc/dfs/sharetab file stores the results of the share commands. This file contains a table of local resources currently being shared. The following example shows that two nfs resources are shared in read-only mode.

```
# cat /etc/dfs/sharetab
/usr/local/data - nfs ro Shared data files
/rdbms_files - nfs ro,root=sys01 Database files
```

The /etc/dfs/fstypes File

The /etc/dfs/fstypes file lists a system's distributed file system types. For each distributed file system type, there is a line beginning with the file system type, which is used with the -F option of the share and mount commands. The file system type listed on the first line of this file is the default file system type when entering DFS administration commands without the -F fstypes option.

```
# cat /etc/dfs/fstypes
nfs NFS Utilities
autofs AUTOFS Utilities
cachefs CACHEFS Utilities
```

The /etc/rmtab File

The /etc/rmtab file contains a table of file systems remotely mounted by NFS clients. After a client successfully completes an NFS mount request, the mountd daemon on the server makes an entry in the /etc/rmtab file. This file also contains a line entry for each remotely mounted directory that has been successfully unmounted, except that the mountd daemon replaces the first character in the entry with the (#) character. For example:

```
# The format of this file follows the syntax
# hostname:fsname

# cat /etc/rmtab
sys-03:/usr/local/data
sys-02:/export/config
#ys-02:/export/config
```

The entries for unmounted directories, indicated with the (#) mark in the first character of the system name, are removed by the mountd daemon during a system startup.

The /etc/default/nfs File

The /etc/default/nfs file lists parameters that can be set for NFS daemon and NFS protocols. Each entry has a description of the item and the default value if the item is commented out.

Details of each entry in this file can be found in `man nfs`.

The NFS Server Daemons

You need several daemons to support NFS activities. These daemons can support both NFS client and NFS server activity, NFS server activity alone, or logging of the NFS server activity.

To start the NFS server daemons or to specify the number of concurrent NFS requests that can be handled by the `nfsd` daemon, enable the `svc:/network/nfs/server` service.

```
# svcadm -v enable nfs/server
svc:/network/nfs/server:default enabled.
```

If a system has entries in its `/etc/dfs/dfstab` file, these server daemons start when the system enters multi-user-server milestone. Table 2-2 lists the NFS server daemons.

Table 2-2 NFS Server Daemons

| Daemon | Description | NFSv4 |
|----------|--|-------|
| mountd | Handles file system mount requests from remote systems, and provides access control. | No |
| nfsd | Handles client file system requests. | Yes |
| statd | Works with the <code>lockd</code> daemon to provide crash recovery functions for the lock manager. | No |
| lockd | Supports record locking operations on NFS files. | No |
| nfslogd | Provides operational logging for NFSv2 and 3. | No |
| nfsmapid | NFS user and group ID mapping daemon | Yes |

In NFSv4, the features provided by the `mountd` and `lockd` daemons are integrated into the NFSv4 protocol. This reduces the number of daemons required on the server and makes the NFS server implementation and management easier.

In NFSv2 and NFSv3, the mount protocol is implemented by the separate `mountd` daemon which did not use an assigned, well-known port number. This made it very hard to use NFS through a firewall. NFSv4 includes the mount protocol and uses the well-known port number 2049 which improves support for NFS use through a firewall.

The mountd Daemon

The `mountd` daemon handles NFS file system mount requests from remote systems and provides access control. The `mountd` daemon checks the `/etc/dfs/sharetab` file to determine whether a particular file or directory is being shared and whether the requesting client has permission to access the shared resource.

When an NFS client issues an NFS mount request, the `mount` command on the client contacts the `mountd` daemon on the server. The `mountd` daemon provides a file handle to the client. File handles are client references that uniquely identify a file or directory on the server. File handles encode a file's inode number, inode generation number, and disk device number.

In NFSv4 file handle and path name mapping is implemented into the NFSv4 protocol, removing the need for a separate `mountd` daemon. The `mountd` daemon is only required for NFSv3 and NFSv2.

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

The `mountd` daemon is started by the `svc:/network/nfs/server` service.

The nfsd Daemon

When a client process attempts to access a remote file resource, the `nfsd` daemon on the NFS server receives the request and the resource's file handle, and then performs the requested operation. This daemon returns any data to the requesting client process.

The `nfsd` daemon also handles file system data requests from clients. Only the superuser can start the `nfsd` daemon. The `nfsd` daemon is started by the `svc:/network/nfs/server` service.

The statd Daemon

The statd daemon works with the lock manager lockd daemon to provide crash recovery functions for the lock manager. The server's statd daemon tracks the clients that are holding locks on an NFS server. When the NFS server reboots after a crash, the statd daemon on the server contacts the statd daemon on the client, which informs the lockd daemon to reclaim any locks on the server. When an NFS client reboots after a crash, the statd daemon on the client system contacts the statd daemon on the server, which invokes the lockd daemon to clear any previous client process locks on the server.

The statd daemon is started by the `svc:/network/nfs/status` service. The statd daemon is not used by NFSv4.

The lockd Daemon

The lockd daemon supports record locking operations for NFS files. The daemon sends locking requests from the NFS client to the NFS server. The server's lockd daemon enables local locking on the NFS server.

The lockd daemon is started by the `svc:/network/nfs/lockmgr` service.

NFSv4 is stateful, unlike NFSv3 and NFSv2. File locking support is integrated into the NFSv4 protocol. The lockd daemon is not used with NFSv4.

The nfslogd Daemon

The nfslogd daemon provides operational logging for an NFS server. NFS logging is enabled when the share is made available. For all file systems for which logging is enabled, the NFS kernel module records all operations in a buffer file. The nfslogd daemon periodically processes the contents of the buffer files to produce American Standard Code for Information Interchange (ASCII) log files, as defined by the contents of the `/etc/default/nfslogd` file.

The nfslogd daemon also handles the mapping of file handles to path names. The daemon keeps track of these mappings in a *file-handle-to-path* mapping table. After post-processing, the ASCII log files store the records. NFS logging is not supported in NFSv4. The nfslogd daemon is started by the `svc:/network/nfs/server` service.

The nfsmapid Daemon

The nfsmapid daemon is implemented in NFSv4. The nfsmapid daemon maps owner and group identification that both the NFSv4 client and server use. There is no user interface to this daemon, but parameters can be set in the /etc/default/nfs file.

The nfsmapid daemon is started by the svc:/network/nfs/mapid service.

Managing the NFS Server Daemons

The NFS daemons start conditionally when the system transitions through the run levels, or they start manually when enabling the svc:/network/nfs/server service.

Note – The nfssd and mounted daemons are started if there is an uncommented share statement in the system's /etc/dfs/dfstab file.

The svcs command can be used to show the dependencies of the nfs/server service.

```
# svcs | grep nfs
online          15:35:24 svc:/network/nfs/client:default
online          15:35:29 svc:/network/nfs/status:default
online          15:35:30 svc:/network/nfs/nlockmgr:default
online          15:35:30 svc:/network/nfs/server:default
online          15:35:31 svc:/network/nfs/mapid:default
online          15:35:32 svc:/network/nfs/rquota:default
```

```
# svcs -l nfs/server
fmri           svc:/network/nfs/server:default
name          NFS server
enabled        true
state          online
next_state    none
state_time    Mon Feb 28 15:35:30 2005
logfile       /var/svc/log/network-nfs-server:default.log
restarter     svc:/system/svc/restart:default
contract_id   44
dependency   require_any/error svc:/milestone/network (online)
dependency   require_all/error svc:/network/nfs/nlockmgr (online)
dependency   optional_all/error svc:/network/nfs/mapid (online)
dependency   require_all/restart svc:/network/rpc/bind (online)
dependency   optional_all/none svc:/network/rpc/keyserv (disabled)
dependency   optional_all/none svc:/network/rpc/gss (online)
dependency   require_all/error svc:/system/filesystem/local (online)
```

Starting the NFS Server Daemons

The `svc:/network/nfs/server` service starts the NFS server daemons when the system enters run level 3.

To start the NFS server daemons manually, place an entry in the `/etc/dfs/dfstab` file and perform the command:

```
# svcadm enable svc:/network/nfs/server
```

Stopping the NFS Server Daemons

By default, the NFS server daemons are shut down by the service management facility (SMF) when it leaves the multi-user-server milestone.

To stop the NFS server daemons manually, perform the command:

```
# svcadm disable svc:/network/nfs/server
```

NFS Server Commands

Table 2-3 lists the NFS server commands.

Table 2-3 NFS Server Commands

| Commands | Description |
|------------|---|
| share | Makes a local directory on an NFS server available for mounting. It also displays the contents of the /etc/dfs/sharetab file. |
| unshare | Makes a previously available directory unavailable for client side mount operations. |
| shareall | Reads and executes share statements in the /etc/dfs/dfstab file. |
| unshareall | Makes previously shared resources unavailable. |
| dfshares | Lists available shared resources from a remote or local NFS server. |
| dfmounts | Displays a list of NFS server directories that are currently mounted. |

Configuring the NFS Server for Sharing Resources

The following sections describe the basic functionality of the NFS server commands. These commands configure shared remote resources.

Making File Resources Available for NFS Mounting

When the `mountd` and `nfsd` daemons are running, you can use the `share` command to make file resources available:

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

where:

| | |
|------------------------------------|--|
| <code>-F <i>nfs</i></code> | Specifies the file system type. This option is not typically required, because NFS is the default remote file system type. |
| <code>-o <i>options</i></code> | Controls a client's access to an NFS shared resource. |
| <code>-d <i>description</i></code> | Describes the shared file resource. |
| <code><i>pathname</i></code> | Specifies the absolute path name of the resource for sharing. |

Note – Unless you specify an option to the `share` command, for example, `-F nfs`, the system uses the file system type from the first line of the `/etc/dfs/fstypes` file.



To share a file resource from the command line, you can use the `share` command. For example, to share the `/usr/local/data` directory as a read-only shared resource, perform the command:

```
# share -o ro /usr/local/data
```

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

The following share command options shown in Table 2-4 restrict the read and write capabilities for NFS clients and enable superuser access to a mounted resource.

Table 2-4 The share Command Options

| Options | Definitions |
|--------------------------|---|
| ro | Informs clients that the server accepts only read requests |
| rw | Allows the server to accept read and write requests from the client |
| root= <i>access-list</i> | Informs clients that the root user on the specified client system or systems can perform superuser-privileged requests on the shared resource |
| ro= <i>access-list</i> | Allows read requests from the specified access list |
| rw= <i>access-list</i> | Allows read and write requests from the specified access list, as shown in Table 2-5 |
| anon= <i>n</i> | Sets <i>n</i> to be the effective user ID (EUID) of anonymous users. By default, anonymous users are given the EUID 60001 (UID_NOBODY). If <i>n</i> is set to -1, access is denied. |

Table 2-5 lists access list options.

Table 2-5 Access List Options

| Option | Description |
|----------------------------------|--|
| <i>access-list=client:client</i> | Allows access based on a colon-separated list of one or more clients. |
| <i>access-list=@network</i> | Allows access based on a network number (for example, @192.168.100) or a network name (for example, @mynet.com). The network name must be defined in the /etc/networks file. |
| <i>access-list=.domain</i> | Allows access based on a Domain Name System (DNS) domain; the dot (.) identifies the value as a DNS domain. |
| <i>access-list=netgroup_name</i> | Allows access based on a configured net group (Network Information Service [NIS] or Network Information Service Plus [NIS+] only). |

You can combine these options by separating each option with commas, which forms intricate access restrictions. The following examples show some of the more commonly used options:

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

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

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

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

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

By setting the option anon=0, the EUID for access to shared resources by an anonymous user is set to 0. The access is also set to read-only.

```
# share -F nfs \
-o ro=client1:client2,rw=client3:client4,root=client4 directory
```

This command shares the directory to the four named hosts only. The hosts, client1 and client2, have read-only access. The hosts client3 and client4 have read-write access. The root user from host client4 has root privilege access to the shared directory and its contents.

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



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

```
# share  
- /usr/local/data ro "Shared data files"  
- /rdbms_files rw,root=sys01 "Database files"
```

Making File Resources Unavailable for Mounting

Use the unshare command to make file resources unavailable for mount operations. This command reads the /etc/dfs/sharetab file.

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

where:

-F nfs Specifies NFS as the file system type. Because NFS is the default remote file system type, you do not have to specify this option.

pathname Specifies the path name of the file resource to unshare.

For example, to make the /export/sys44_data directory unavailable for client-side mount operations, perform the command:

```
# unshare /usr/local/data
```

Sharing and Unsharing All NFS Resources

Use the shareall and unshareall commands to share and unshare all NFS resources.

The shareall command, when used without arguments, shares all resources listed in the /etc/dfs/dfstab file.

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

The `unshareall` command, when used without arguments, unshares currently shared file resources listed in the `/etc/dfs/sharetab` file.

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

Displaying Currently Shared NFS Resources

The **dfshares** command uses the NFS daemon, **mountd**, to display currently shared NFS resources.

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

The **dfshares** command displays resources currently being shared by the local server when used without a *host* argument.

```
# share -F nfs -o ro /usr/local/data
# dfshares
```

| RESOURCE | SERVER | ACCESS | TRANSPORT |
|------------------------|--------|--------|-----------|
| sys-02:/usr/local/data | sys-02 | - | - |

By specifying one or more server names as arguments, the **dfshares** command also displays file resources being shared by other servers. For example:

```
# dfshares sys-01
RESOURCE
    sys-01:/usr/share/man
```

| RESOURCE | SERVER | ACCESS | TRANSPORT |
|-----------------------|--------|--------|-----------|
| sys-01:/usr/share/man | sys-01 | - | - |

Displaying NFS Mounted Resources

The **dfmounts** command displays remotely mounted NFS resource information.

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

The **dfmounts** command, when used without arguments, displays a list of directories on the local server that are currently mounted and also displays a list of the client systems that currently have the shared resource mounted.

```
# dfmounts
RESOURCE
    -
    SERVER PATHNAME
    sys-02 /usr/local/data
```

| RESOURCE | SERVER | PATHNAME | CLIENTS |
|----------|--------|-----------------|---------|
| - | sys-02 | /usr/local/data | sys-03 |

Note – Since the **dfmounts** command uses the **mountd** daemon to display currently shared NFS resources, it will not display NFSv4 shares.



Managing the NFS Client

NFS client files, NFS client daemons, and NFS client commands work together to manage the NFS client.

NFS Client Files

You need several files to support NFS client activities on any computer. Table 2-6 lists the files that support NFS client activities.

Table 2-6 NFS Client Files

| File | Description |
|------------------|---|
| /etc/vfstab | Defines file systems to be mounted locally. |
| /etc/mnttab | Lists currently mounted file systems, including automounted directories. The contents of this file are maintained by the kernel and cannot be edited. |
| /etc/dfs/fstypes | Lists the default file system types for remote file systems. |
| /etc/default/nfs | Contains parameters used by NFS protocols and daemons. |

The /etc/vfstab File

To mount remote file resources 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
#
sys-02:/usr/local/data    -   /usr/remote_data    nfs     -     yes        soft,bg
```

The /etc/mnttab File

The /etc/mnttab file system provides read-only access to the table of mounted file systems for the current host. Mounting a file system adds an entry to the table of mounted file systems. Unmounting a file system removes an entry from the table of mounted file systems.

Remounting a file system updates the information in the mounted file system table. The kernel maintains a chronological list in the order of the mount time. The first mounted file system is first on the list and the most recently mounted file system is last. Although the /etc/mnttab file is a mount point for the mntfs file system, it appears as a regular file containing the current mount table information. The /lib/svc/method/fs-user script establishes the mntfs file system during the boot process.

The /etc/dfs/fstypes File

As with an NFS server, NFS clients use the /etc/dfs/fstypes file to determine distributed file system support.

```
# cat /etc/dfs/fstypes
nfs NFS Utilities
autofs AUTOFS Utilities
cachefs CACHEFS Utilities
```

NFS Client Daemons

The NFS client daemons are started using the svc:/network/nfs/client service. Table 2-7 lists the NFS client daemons.

Table 2-7 NFS Client Daemons

| Daemon | Description |
|---------|--|
| statd | Works with the lockd daemon to provide crash recovery functions for the lock manager |
| lockd | Supports record-locking operations on NFS files |
| nfs4cbd | NFSv4 callback daemon. |

Managing the NFS Client Daemons

Two NFS daemons, the statd daemon and the lockd daemon, run on NFS servers and the NFS clients. These daemons start automatically when a system enters the network milestone. This can be seen by examining the dependencies for the network milestone.

```
# svcs -D milestone/network
STATE          STIME      FMRI
disabled       15:34:35  svc:/network/dns/client:default
disabled       15:34:37  svc:/network/nfs/cbd:default
disabled       15:34:38  svc:/network/rpc/bootparams:default
disabled       15:34:39  svc:/network/rarp:default
disabled       15:34:51  svc:/network/dns/server:default
disabled       15:34:52  svc:/network/slp:default
disabled       15:35:20  svc:/network/shell:kshell
online         15:35:03  svc:/milestone/single-user:default
online         15:35:04  svc:/network/initial:default
online         15:35:13  svc:/network/inetd:default
online         15:35:24  svc:/network/nfs/client:default
online         15:35:26  svc:/network/shell:default
online         15:35:30  svc:/network/nfs/server:default
online         15:35:31  svc:/network/nfs/mapid:default
online        16:31:18  svc:/network/nfs/nlockmgr:default
online        16:33:12  svc:/network/nfs/status:default
```

Both the statd and lockd daemons provide crash recovery and locking services for NFS version 2 and 3. If a server crashes, clients can quickly re-establish connections with files they were using. Therefore, the server has a record of the clients that were using its NFS resources. It contacts each client for information about which files were in use, which helps to provide continuous operation. You can start both of these daemons using the svcadm command.

The lockd daemon is started by the SMF service nfs/nlockmgr.

```
# svcadm -v enable nfs/nlockmgr
svc:/network/nfs/nlockmgr:default enabled.
```

The statd daemon is started by the SMF service nfs/status.

```
# svcadm -v enable nfs/status
svc:/network/nfs/status:default enabled.
```

Neither daemon requires administrative intervention.

Restarting the NFS Client Daemons

The service management facility automatically starts the NFS client daemons when the system enters the network milestone, and shuts down NFS client daemons when the system enters the single-user milestone.

To manually restart these daemons, perform the command:

```
# svcadm -v restart nfs/status  
Action restart set for svc:/network/nfs/status:default.  
# svcadm -v restart nfs/nlockmgr  
Action restart set for svc:/network/nfs/nlockmgr:default.  
#
```

NFS Client Commands

Table 2-8 lists the NFS client commands.

Table 2-8 NFS Client Commands

| Command | Description |
|-----------|--|
| dfshares | Lists available shared resources from a remote or local NFS server |
| mount | Attaches a file resource (local or remote) to a specified local mount point |
| umount | Unmounts a currently mounted file resource |
| mountall | Mounts all file resources or a specific group of file resources listed in the /etc/vfstab file with a mount at boot value of yes |
| umountall | Unmounts all non-critical local and remote file resources |

Configuring the NFS Client for Mounting Resources

The following sections describe some of the functions of the NFS client utilities.

Displaying a Server's Available Resources

You can use the dfshares command to list resources made available by an NFS server. To verify the resources that an NFS server is currently making available, run the dfshares command with the server name as an argument.

```
# dfshares sys-02
RESOURCE                                SERVER ACCESS      TRANSPORT
sys-02:/usr/local/data                  sys-02   -
sys-02:/rdbms_files                     sys-02   -
```

Accessing the Remote File Resource

Enter the /usr/sbin/mount command to attach a local or remote file resource to the file system hierarchy.

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

where:

-F nfs

Specifies NFS as the file system type. The *-F nfs* option is not necessary, because NFS is the default remote file system type specified in the /etc/dfs/fstypes file.

-o options

Specifies a comma-separated list of file-system specific options, such as *rw*. The *rw* option mounts the file resource as read, write. The *ro* option mounts the file resource as read-only. (The default is *rw*.)

server:pathname

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

mount_point

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

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

```
# mount sys-02:/rdbms_files /rdbms_files
```

When mounting a read-only remote resource, you can specify a comma-separated list of sources for the remote resource, which are then used as a list of failover resources. This process works if the resource mounted from all of the servers in the list is the same. For example:

```
# mount -o ro sys-45,sys-43,sys-41:/multi_homed_data /remote_shared_data
```

In this example, if the sys-45 server is unavailable, the request passes to the next server on the list, sys-43, and then to the sys-41 server.

Unmounting the Remote File Resources From the Client

Use the `umount` command to detach local and remote file resources from the file system hierarchy. This command reads the `/etc/mnttab` file on the client.

```
umount server:pathname | mount_point
```

The command can specify either the `server:pathname` option or the `mount_point` option.

```
# umount /rdbms_files
```

Mounting All File Resources

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.

```
mountall -r [ -F nfs ]  
# mountall -r
```

Unmounting All Currently Mounted File Resources

When you use the `umountall` command without any arguments, it unmounts all currently mounted file resources except for the root (/), /usr, /var, /var/adm, /var/run, /proc, and /dev/fd directories. To restrict the unmounting to only remote file systems, use the `-r` option.

```
umountall -r [ -F nfs ]
# umountall -r
```



Note – Use the `-F` FSType with the `mountall` and `umountall` commands to specify FSType as the file system type. You do not have to specify the `-F nfs` option, because NFS is listed as the default remote file system type.

Mounting Remote Resources at Boot Time

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

| #device #to mount # | device to fsck | mount point | FS type | fsck pass | mount at boot | mount options |
|---------------------------|-------------------|------------------|------------|--------------|------------------|------------------|
| sys-02:/usr/local/data | - | /usr/remote_data | nfs | - | yes | soft,bg |

where the fields in the `/etc/vfstab` file are:

| | |
|--------------------|--|
| device to mount | The name of the server and the path name of the remote file resource. The server host name and share name are separated by a (:). |
| device to fsck | NFS resources are not checked by the client because the file system is not local to the client. This field is always (-) for NFS resources. |
| mount point | The mount point for the resource. |
| FS type | This field specifies the type of file system to be mounted. |
| fsck pass | NFS resources are not checked by the client, because the file system is not local to the client. This field is always (-) for NFS resources. |



| | |
|---------------|---|
| mount at boot | This field can contain either of two values, yes or no. If the field is set to the value yes, the specified resource is mounted every time the mountall command is run. |
| mount options | A comma-separated list of mount options. See Table 2-9 on page 2-30 for a description of each option. |

Note – If the /etc/vfstab file contains the file resource, 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.

Table 2-9 The mount Command Options

| Option | Description |
|-------------|---|
| rw ro | Specifies whether the resource is mounted as read/write or read-only. The default is read/write. |
| bg fg | During an NFS mount request, if the first mount attempt fails, retry in the background or foreground. The default is to retry in the foreground. |
| soft hard | <p>When the number of retransmissions has reached the number specified in the retrans=n option, a file system mounted with the soft option reports an error on the request, and stops trying. A file system mounted with the hard option prints a warning message and continues to try to process the request. The default is a hard mount.</p> <p>Although the soft option and the bg option are not the default settings, combining them usually results in the fastest client boot when NFS mounting problems occur.</p> |

Table 2-9 The mount Command Options (Continued)

| Option | Description |
|-------------------|---|
| intr nointr | <p>Enables or disables the use of keyboard interrupts to kill a process that hangs while waiting for a response on a hard-mounted file system. The default is <code>intr</code>.</p> <p>The <code>intr</code> option is not specifically required with the <code>soft</code> option as this option allows the NFS mount to time out and fail if the mount is unsuccessful over the <code>retry/retrans</code> limits.</p> <p>If the <code>intr</code> option is applied with the <code>hard</code> option, this allows the user to interrupt a manually executed mount instruction, that is currently failing to mount, by using the Ctrl-C interrupt. If the <code>nointr</code> option is applied with the <code>hard</code> option, the mount takes as long as is required to successfully mount.</p> <p>The <code>intr</code> option is not applicable at boot time as the mount operation is being performed by a daemon process that cannot send a Ctrl-C character to the NFS mount process.</p> |
| suid nosuid | Indicates whether to enable setuid execution. The default enables setuid execution. |
| timeo= <i>n</i> | Sets the timeout to <i>n</i> tenths of a second. The default timeout is 11, measured in one-tenth of a second (0.1 second) for User Datagram Protocol (UDP) transports, and 600 tenths of a second for Transmission Control Protocol (TCP). |
| retry= <i>n</i> | Sets the number of times to retry the mount operation. The default is 10,000 times. |
| retrans= <i>n</i> | Sets the number of NFS retransmissions to <i>n</i> . The default is 5 for UDP. For the connection-oriented TCP, this option has no effect. |

Enabling NFS Server Logging

Maintain an NFS activity log to:

- Track remote file accesses on your network
- Assist in debugging NFS failures

Fundamentals of NFS Server Logging

Note – Server logging is not supported in NFSv4.

The NFS server logging feature records NFS transactions on the file system. The nfslogd daemon provides operational logging.

When you enable NFS server logging, the NFS kernel module writes records of all NFS operations on the file system into a buffer file. The data includes a time stamp, the client IP address, the UID of the requester, the file handle of the resource being accessed, and the type of operation that occurs.

The nfslogd Daemon

The functions of the nfslogd daemon are that it:

- Converts the raw data from the logging operation into ASCII records, and stores the raw data in ASCII log files.
- Resolves IP addresses to host names and UIDs to login names.
- Maps the file handles to path names, and records the mappings in a file-handle-to-path mapping table. Each tag in the /etc/nfs/nfslog.conf file corresponds to one mapping table.

Note – If the nfslogd daemon is not running, changes are not tracked to the mappings in the file-handle-to-path table.



Configuring NFS Log Paths

The `/etc/nfs/nfslog.conf` file defines the path, file names, and type of logging that the `nfslogd` daemon must use. There is a tag corresponding to each definition.

To configure NFS server logging, identify or create the tag entries for each of the server's shared resources. The `global` tag defines the default values.

The following is an example an `nfslog.conf` file:

```
# cat /etc/nfs/nfslog.conf
#ident  "@(#)nfslog.conf          1.5      99/02/21 SMI"
#
# Copyright (c) 1999 by Sun Microsystems, Inc.
# All rights reserved.
#
# 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:

| | |
|---------------------------------------|--|
| <code>defaultdir=dir_path</code> | Specifies the default parent directory. All relative path entries to this log can be seen. |
| <code>log=logfile_path</code> | Specifies the relative or absolute path and the file name for the ASCII log file. |
| <code>fhtable=table_path</code> | Specifies relative or absolute path and the file name for the file-handle-to-path database file. |
| <code>buffer=bufferfile_path</code> | Specifies the relative and absolute path and the file name for the raw buffer file. |
| <code>logformat=basic/extended</code> | Specifies the format when creating user-readable log files. The basic format produces a log file similar to the FTPdaemon. The extended format gives a more detailed view. |

If you do not specify an absolute path in the parameters, the nfslogd daemon appends the name given to the path specified by the defaultdir parameter. To override the value specified by the defaultdir parameter, use an absolute path.

To easily identify the log files for different shared resources, place them 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
public defaultdir=/var/nfs/public \
        log=nfslog fhtable=fhtable buffer=nfslog_workbuffer
```

Note – Create the /var/nfs/public directory before starting NFS server logging.

In the previous example, any file system shared with log=public uses the following values:

- The default directory is the /var/nfs/public directory.
- The log is stored in the /var/nfs/public/nfslog file.
- The /var/nfs/public/fhtables file stores the *file-handle-to-path* database.
- The /var/nfs/public/nfslog_workbuffer file stores the buffer.



Initiating NFS Logging

To initiate NFS server logging, complete the following steps:

1. Become superuser.
2. Optional: Change the file system configuration settings. In the /etc/nfs/nfslog.conf file, either:
 - Edit the default settings for all file systems by changing the data corresponding to the global tag.
 - Add a new tag for the specific file system.
 If you do not need these changes, do not edit this file.
3. To share file systems using NFS server logging, you must first enable NFS server logging. Edit the /etc/dfs/dfstab file to add an entry for file systems for which you want to enable NFS server logging. Either:
 - Specify a tag by entering the tag to use with the `log=tag` option in the /etc/dfs/dfstab file.
 - Use the `log` option without specifying a tag, which causes the option to use the `global` tag as a default. The following example uses the default settings in the `global` tag:

```
share -F nfs -o log /export/sys44_data
```

4. Check that the NFS service is running on the server.

To start or restart the `mountd`, `nfsd`, and `nfslogd` daemons if they are not running, perform the command:

```
# svcadm enable svc:/network/nfs/server
```

If the /etc/nfs/nfslog.conf file exists and you execute the `nfs.server` script, the `nfs.server` script starts the `nfslogd` daemon.

5. Run the `share` command to verify that the correct options are listed.

```
# share
-          /export/sys44_data    ro,log    "
```

6. If you add the additional entries to the /etc/dfs/dfstab file, share the file system by rebooting the system or entering the `shareall` command.

```
# shareall
```

Configuring the nfslogd Daemon Behavior

The configuration information in the /etc/default/nfslogd file controls the logging behavior of the nfslogd daemon.

The /etc/default/nfslogd file defines default parameters used for NFS server logging. Table 2-10 describes some of the NFS logging parameters.

Table 2-10 NFS Logging Parameters

| Parameter | Description |
|---------------------|---|
| IDLE_TIME | Sets the amount of time that the nfslogd daemon sleeps before checking the buffer file for more information. 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. |
| 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 is 524,288 bytes. Increasing this number can improve performance by reducing the number of times that the buffer file is processed. The MIN_PROCESSING_SIZE and the IDLE_TIME parameters determine how often the buffer file is processed. |
| UMASK | Specifies the permissions for the log files set by the nfslogd daemon. The default value is 0137. |
| CYCLE_FREQUENCY | Determines the time that must pass before the log files are cleared. The default value is 24 hours. Use the CYCLE_FREQUENCY parameter to prevent the log files from becoming too large. |
| MAX_LOGS_PRESERVE | Determines the number of log files to save. The default value is 10. |

NFS Version 4 (NFSv4)

The Solaris 10 OS includes NFSv4 in addition to NFSv3 and NFSv2.

NFSv4 includes features that were not in the previous versions of NFS. These features include the following:

- Stateful connections.
- Single protocol, reducing the number of service-side daemons.
- Improved Firewall Support. NFSv4 uses the well-known port number 2049.
- Pseudo file systems which ensure the NFS client has seamless access to all exported objects on the server and that portions of a server file system that are not explicitly exported are not visible to the client.
- Strong security.
- Extended attributes
- Delegation. In the Solaris 10 NFSv4 release, the NFS server can hand over delegation of management of a shared file to the client requesting that file. It is the server that decides whether or not to apply delegation. By delegating read or write management control to the client, this can greatly reduce the amount of network traffic otherwise caused by clients making requests for the current state of a shared file.

Pseudo-File System

Previous versions of NFS required use of the `mountd` protocol, which does not use assigned ports. This made NFS hard to use through a firewall. Implementation of NFSv4 must support Transmission Control Protocol/Internet Protocol (TCP/IP) to provide congestion control. NFSv4 uses the well-known port 2049, thus improving firewall support.

NFSv4 maps file handles to path names, which the `mountd` protocol did in previous NFS versions. In NFSv4, the server provides a root file handle that represents the top of the file system that the server exported. The NFS server maintains a pseudo -file system which contains the full path of each exported file system.

The pseudo-file system, created and maintained by NFSv4 servers, provides clients with seamless access to all exported objects on the server. Before NFSv4, this pseudo-file system did not exist. Clients had to mount each shared server file system for access. Figure 2-3 shows an example.

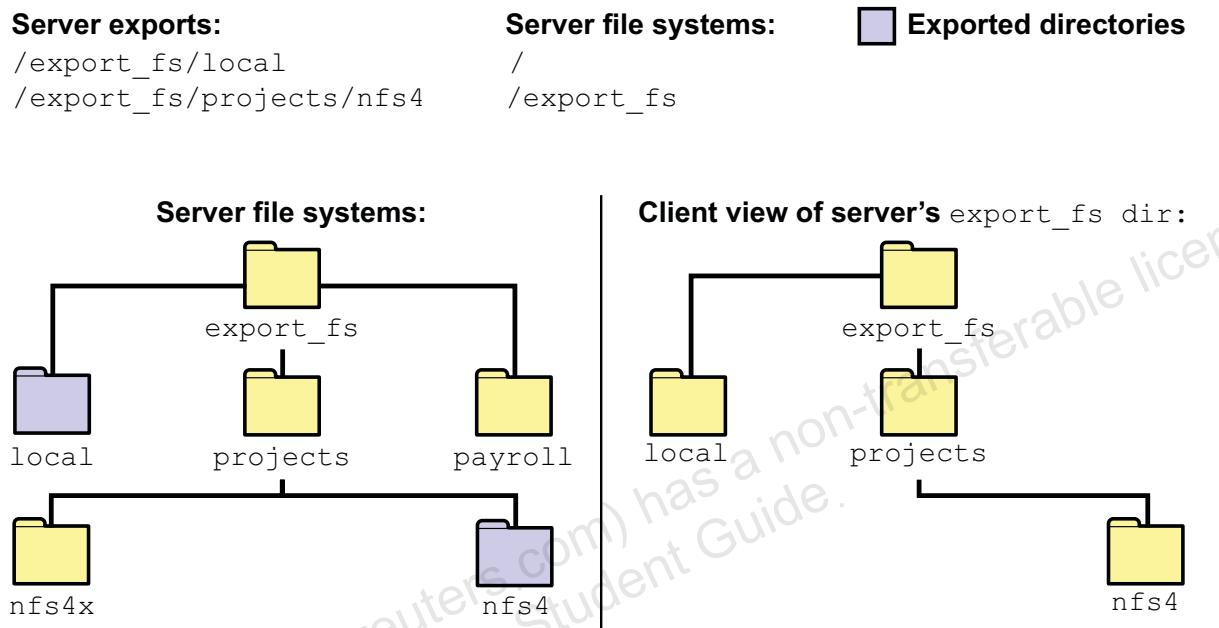


Figure 2-3 Views of the Server File System and Client File System

In Figure 2-3 the client cannot see the payroll directory and the nfs4x directory because these directories are not exported and do not lead to exported directories. However, the client can see the local directory because local is an exported directory. The projects directory is visible to the client because the projects directory leads to the exported directory, nfs4. Thus, portions of the server namespace that are not explicitly exported are bridged with a pseudo-file system that views only the exported directories and those directories that lead to server exports.

This pseudo-file system is a structure that contains only directories, and is created by the server. The pseudo-file system permits a client to browse the hierarchy of exported file systems. Thus, the client's view of the pseudo-file system is limited to paths that lead to exported file systems.

Previous versions of NFS did not permit a client to traverse server file systems without mounting each file system. However, in NFSv4, the server namespace does the following:

- Restricts the client's file-system view to directories that lead to server exports.
- Provides clients with seamless access to server exports without requiring that the client mount each underlying file system.

See the example in Figure 2-3 on page 2-38. However, different operating systems might require the client to mount each server file system.

NFSv4 is the default NFS version in the Solaris 10 OS. The `nfs(4)` file in the `/etc/default` directory configures the client or server to use NFS versions 2, 3, or 4. In addition, the `mount` command (`mount_nfs (1M)`) can use the `vers=version_number` option to mount a file system using only the version specified.

Strong Security

NFSv4 uses the remote procedure call (RPC) implementation of the General Security Service (GSS) framework to extend the basic security of RPC. This provides mechanisms for authentication, integrity, and privacy between the client and server.

Traditional RPC implementations included `AUTH_NONE`, `AUTH_SYS`, `AUTH_DH`, and `AUTH_KRB4` as security flavors. An additional security method of `RPCSEC_GSS` is introduced that uses the functionality of Generic Security Services Application Programming Interface (GSSAPI). This allows the RPC layer to use various security mechanisms without the additional implementation overhead of adding new security flavors.



Note – `RPCSEC_GSS` security mandated for NFSv4 is the same as that released on Solaris 2.6. The main `RPCSEC_GSS` security model is Kerberos V5.

For NFSv4, the `RPCSEC_GSS` security method must be used to enable the mandatory security mechanism. Other flavors, such as `AUTH_NONE`, `AUTH_SYS`, and `AUTH_DH` may be implemented as well.

The NFS client negotiates with the NFS server to determine the security mechanism that meets the requirements for the server and client. The RPCSEC_GSS framework delivers Sun Enterprise Authentication Mechanism™ (SEAM) software authentication.

You can mix the security mechanisms on a single server, which allows security to be applied on a per-share basis.

To configure a Solaris 10 OS NFSv4 server to use the RPCSEC_GSS security flavor with SEAM software, the administrator first edits the /etc/nfssec.conf file using the nfssec security modes described in the nfssec(5) man page to enable the necessary security mode needed.

The NFSv4 server then shares the file system with the sec=mode option. The system must also be configured as a Kerberos client. For example:

```
# share -F nfs -o sec=krb5 /export/home
```

Compound Procedures

To improve performance and Internet access, the NFSv4 client combines multiple operations which go in to one remote procedure call. By using compound procedures, clients can combine LOOKUP, OPEN, and READ operations in a single request. The server breaks the request into a list of separate requests. The server iterates through the list and performs each operation in the list until it reaches the end of the list or fails. The server then returns the results of the operations to the client.

The following is a simplified example of compound procedures. When reading the /export/testdata file, NFS versions 3 and 4 generate the following RPC calls.

| NFS version 3 | NFS version 4 |
|----------------------|--------------------------|
| -> LOOKUP "export" | ->OPEN "export/testdata" |
| <- OK | READ |
| ->LOOKUP "testdata" | <- OPEN OK |
| <- OK | READ OK |
| -> ACCESS "testdata" | (sends data) |
| <- OK | |
| -> READ "testdata" | |
| <- OK | |

NFS version 3

(sends data)

NFS version 4

Fewer RPC calls result in faster NFS response. This allows the client to tailor its request to appropriately match the operating environment of the client, thus enhancing cross-platform interoperability.

Extended Attributes

Earlier NFS versions used a fixed set of file and file system attributes that were modeled on the UNIX® files and file systems. A non-UNIX-like server or client had to simulate those attributes, making implementation on a non-UNIX system difficult. NFSv4 introduces three categories of attributes: mandatory, recommended, and named. All NFSv4 clients and servers supported the mandatory attributes to ensure a minimum level of interoperability.

Not all clients or servers have to support the recommended attributes. This allows a server to support the attributes that apply to its operating environment. The client determines how to proceed if the server does not support a particular recommended attribute.

The named attribute is in the form of a byte stream that is associated with a file or file system and is referred to by a string name. This allows the client to associate data with a specific file or file system.

File Handles

File handles are created on the server and contain information that uniquely identifies files and directories. In NFS versions 2 and 3, the server returned persistent file handles. This meant the client could guarantee that the server would generate a file handle that always referred to the same file. The following is an example:

- If a file was deleted and replaced with a file of the same name, the server would generate a new file handle for the new file. If the client used the old file handle, the server would return an error that the file handle was stale.
- If a file was renamed, the file handle would remain the same.
- If you had to reboot the server, the file handles would remain the same.

When the server received a request from a client that included a file handle, the resolution was straightforward, and the file handle always referred to the correct file.

This method of identifying files and directories for NFS operations worked well for most UNIX-based servers, but could not be implemented on servers that relied on other methods of identification, such as a file's path name. To resolve this problem, the NFSv4 protocol permits a server to declare that its file handles are volatile. Thus, a file handle could change. If the file handle does change, the client must find the new file handle.

Like NFS versions 2 and 3, the Solaris OS NFS version 4 server always provides persistent file handles. However, Solaris OS NFSv4 clients that access non-Solaris OS NFSv4 servers must support volatile file handles if the server uses them. Specifically, when the server tells the client that the file handle is volatile, the client must cache the mapping between path name and file handle. The client uses the volatile file handle until it expires. Upon expiration, the client does the following:

- Flushes the cached information that refers to that file handle
- Searches for that file's new file handle
- Retries the operation

Delegation

NFSv4 provides both client support and server support for delegation. Delegation is a technique by which the server delegates the management of a file to a client.

For example, the server could grant either a read delegation or a write delegation to a client. You can grant read delegations to multiple clients at the same time, because these read delegations do not conflict with each other. A write delegation can be to only one client, because a write delegation conflicts with any file accessed by any other client.

While holding a write delegation, the client would not send various operations to the server because the client is guaranteed exclusive access to a file. Similarly, the client would not send various operations to the server while holding a read delegation because the server guarantees that no client can open the file in write mode.

The server alone decides whether to grant a delegation. A client does not request a delegation. The server decides based on the access patterns for the file. If several clients recently accessed a file in write mode, the server might not grant a delegation because this access pattern indicates the potential for future conflicts.

A conflict occurs when a client accesses a file in a manner that is inconsistent with the delegations that are currently granted for that file. For example, if a client holds a write delegation on a file and a second client opens that file for read or write access, the server recalls the first client's write delegation. Similarly, if a client holds a read delegation and another client opens the same file for writing, the server recalls the read delegation.

One server does not resolve access conflicts for a file that is stored on another server. Thus, an NFS server resolves only conflicts for files that it stores. Furthermore, in response to conflicts that are caused by clients that are running various versions of NFS, an NFS server can initiate only recalls to the client that is running NFSv4. An NFS server cannot initiate recalls for clients that are running earlier versions of NFS.

Note – By default, server delegation is enabled when NFSv4 is started.



Configuring an NFSv4 Server

When configuring the NFS server in the Solaris 10, Update 3 environment, the first step is to add the appropriate entries in the /etc/default/nfs file. This file allows NFS to be configured without making changes to the service management facility service properties.

You must log in as superuser or assume an equivalent role to edit the file.

1. Edit the /etc/default/nfs file.
2. Make the following entries to configure an NFSv4 only server:

```
NFS_SERVER_VERSMAX=4  
NFS_SERVER_VERSMIN=4
```

While numerous parameters are supported, only those used to configure the NFSv4 server are considered here.

See the nfs(4) man page for a complete list of possible parameters.

`NFS_SERVER_VERSMIN=num`
`NFS_SERVER_VERSMAX=num`

The NFS server uses only NFS versions in the range these variables specify. Valid values or versions are: 2, 3, and 4. By default these variables are unspecified (commented out) and the client's default minimum is Version 2. The default maximum is Version 4.

3. If required, make the following entry:

`NFS_SERVER_DELEGATION=off`

By default, this variable is commented out and the NFS server does provide delegations to clients. The user can turn off delegations for all exported file systems by setting this variable to `off` (case sensitive). This variable applies only to NFSv4.

4. If required, make the following entry:

`NFSMAPID_DOMAIN=my.comany.com`

By default, the `nfsmapid` daemon uses the Domain Name Service (DNS) domain of the system. This setting overrides the default. This domain is used for identifying user and group attribute strings in the NFSv4 protocol. Clients and servers must match with this domain for operation to proceed normally. This variable applies only to NFSv4.

5. Determine if the NFS server is running:

`# svcs network/nfs/server`

If enabled is returned, you must use the following command to stop the service:

`# svcadm disable network/nfs/server`

6. To enable the NFS service:

`# svcadm enable network/nfs/server`

Configuring an NFSv4 Client

When configuring the NFSv4 client, the first step is to add the appropriate entries in the `/etc/default/nfs` file. This file allows NFS to be configured without making changes to the service management facility scripts.

You must login as superuser or assume an equivalent role to edit the file.

1. Edit the `/etc/default/nfs` file.
2. Insert the following lines to configure a NFSv4 only client:

```
NFS_CLIENT_VERSMAX=4  
NFS_CLIENT_VERSMIN=4
```

While numerous parameters are supported, only those used to configure the NFSv4 client are considered here.

See the `nfs(4)` man page for a complete list of possible parameters.

The NFS client only uses NFS versions in the range specified by these variables. Valid values or versions are: 2, 3, and 4. By default these variables are unspecified (commented out) and the client's default minimum is Version 2. The default maximum is Version 4.

3. Mount a file system.

```
# mount server_name:share_point local_dir
```

- **server_name** – Provides the name of the server
- **share_point** – Provides the path of the remote directory to be shared
- **local_dir** – Provides the path of the local mount point

Note – Enable the client service if you are putting NFS mounts in `/etc/vfstab`. You do this by running the `svcadm enable network/nfs/client` command. Otherwise, they will not get mounted if the OS is *Secure By Default*.



Troubleshooting NFS Errors

You can detect most NFS problems from console messages or from certain symptoms that appear on a client system. Some common errors are:

- The rpcbind failure error
- The server not responding error
- The NFS client fails a reboot error
- The service not responding error
- The program not registered error
- The stale file handle error
- The unknown host error
- The mount point error
- The no such file error

The rpcbind failure Error

The following example shows the message that appears on the client system during the boot process or in response to an explicit mount request.

```
nfs mount: server1:: RPC: Rpcbind failure  
RPC: Timed Out  
nfs mount: retrying: /mntpoint
```

The error in accessing the server is 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.
- The hosts database file that supports the client has the correct server node, but the server node temporarily stops due to an overload.

To solve the rpcbind failure error condition when the server node is operational, determine if the server is out of critical resources (for example, memory, swap, or disk space).

The server not responding Error

The following message appears during the boot process or in response to an explicit mount request, and this message indicates a known server that is inaccessible.

```
NFS server server2 not responding, still trying
```

Possible causes for the server not responding error are:

- The network between the local system and the server is down. To verify that the network is down, enter the ping command (ping *server2*).
- The server (*server2*) is down.

The NFS client fails a reboot Error

If you attempt to boot an NFS client and the client-node stops, waits, and echoes the following message:

```
Setting default interface for multicast: add net 224.0.0.0: gateway:  
client_node_name.
```

these symptoms might indicate that a client is requesting an NFS mount using an entry in the /etc/vfstab file, specifying a foreground mount from a non-operational NFS server.

To solve this error, complete the following steps:

1. To interrupt the failed client node press Stop-A, and boot the client into single-user mode.
2. Edit the /etc/vfstab file to comment out the NFS mounts.
3. To continue booting to the default run level (normally run level 3), press Control-D.
4. Determine if all the NFS servers are operational and functioning properly.
5. After you resolve problems with the NFS servers, remove the comments from the /etc/vfstab file.



Note – If the NFS server is not available, an alternative to commenting out the entry in the /etc/vfstab file is to use the bg mount option so that the boot sequence can proceed in parallel with the attempt to perform the NFS mount.

The service not responding Error

The following message appears during the boot process or in response to an explicit mount request, and indicates that an accessible server is not running the NFS server daemons.

```
nfs mount: dbserver: NFS: Service not responding  
nfs mount: retrying: /mntpoint
```

To solve the service not responding error condition, complete the following steps:

1. Enter the who -r command on the server to see if it is at run level 3. If the server is not, change to run level 3 by entering the init 3 command.
2. Enter the ps -e command on the server to check whether the NFS server daemons are running. If they are not, start them with the svcadm enable svc:/network/nfs/server command.

The program not registered Error

The following message appears during the boot process or in response to an explicit mount request and indicates that an accessible server is not running the mountd daemon.

```
nfs mount: dbserver: RPC: Program not registered  
nfs mount: retrying: /mntpoint
```

To solve the program not registered error condition, complete the following steps:

1. Enter the who -r command on the server to check that it is at run level 3. If the server is not, change to run level 3 by performing the init 3 command.
2. Enter the pgrep -f1 mountd command. If the mountd daemon is not running, start it using the svcadm enable svc:/network/nfs/server command.
3. Check the /etc/dfs/dfstab file entries.

The stale NFS file handle Error

The following message appears when a process attempts to access a remote file resource with an out-of-date file handle.

```
stale NFS file handle
```

A possible cause for the stale NFS file handle error is that the file resource on the server moved. To solve the stale NFS file handle error condition, unmount and mount the resource again on the client.

The unknown host Error

The following message indicates that the host name of the server on the client is missing from the hosts table.

```
nfs mount: sserver1:: RPC: Unknown host
```

To solve the unknown host error condition, verify the host name in the hosts database that supports the client node.

Note – The preceding example misspelled the node name server1 as sserver1.



The mount point Error

The following message appears during the boot process or in response to an explicit mount request and indicates a non-existent mount point.

```
mount: mount-point /DS9 does not exist.
```

To solve the mount point error condition, check that the mount point exists on the client. Check the spelling of the mount point on the command line or in the /etc/vfstab file on the client, or comment out the entry and reboot the system.

The no such file Error

The following message appears during the boot process or in response to an explicit mount request, which indicates that there is an unknown file resource name on the server.

No such file or directory

To solve the no such file error condition, check that the directory exists on the server. Check the spelling of the directory on the command line or in the /etc/vfstab file.

Exercise: Configuring NFS

In this exercise, you configure an NFS server and client to share and mount the /usr/share/man file.

Preparation

Choose a partner for this lab. Determine which systems to configure as the NFS server and the NFS client. Verify that entries for both systems exist in the /etc/hosts file on both systems. Refer to your lecture notes as necessary to perform the following steps.

Tasks

Complete the following tasks.

Task 1 – On the NFS Server

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Edit the /etc/dfs/dfstab file. Add an entry to share the directory that holds man pages.
3. Start the NFS server daemons.
4. Verify that the NFS server service is online.
5. Verify that the /usr/share/man directory is shared and that no NFS mounts are present.

Task 2 – On the NFS Client

Complete the following steps:

1. Rename the /usr/share/man directory so that you can no longer access the man pages on the client system. Verify that the man pages are not available.

What message does the man command report?

Exercise: Configuring NFS

2. Create a new man directory (`/usr/share/man`) to use as a mount point.
 3. Verify that the NFS client service is online.
 4. Mount the `/usr/share/man` directory from the server.
 5. Verify that the man pages are available.
Are the man pages available?

 6. Use the `mount` command to display the mount options for the `/usr/share/man` directory. Record the read and write options that the `mount` command displays.
 7. Write a file into the NFS-mounted file system.
What is the result of trying to write to the NFS-mounted file system?

- What conclusion can be reached by this exercise?

8. Unmount the `/usr/share/man` directory.

Task 3 – On the NFS Server

Complete the following steps:

1. Unshare the `/usr/share/man` directory.
2. Change the share statement in the `/etc/dfs/dfstab` file for the `/usr/share/man` directory to read:

```
share -o ro=bogus /usr/share/man
```

3. Share the `/usr/share/man` directory.

Task 4 – On the NFS Client

Complete the following step:

Attempt to mount the `/usr/share/man` directory again.

What happens?

Task 5 – On the NFS Server

Complete the following steps:

1. Unshare the /usr/share/man directory.
2. Edit the /etc/dfs/dfstab file to remove the entry for the /usr/share/man directory.

Task 6 – On the NFS Client

Complete the following steps:

1. Return the /usr/share/man directory to its original configuration.
2. Verify that the man pages are now available.

Exercise Summary

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

- Experiences
- Interpretations
- Conclusions
- Applications



Exercise Solutions: Configuring NFS

In this exercise, you configure an NFS server and client to share and mount the /usr/share/man file.

Preparation

Choose a partner for this lab. Determine which systems to configure as the NFS server and the NFS client. Verify that entries for both systems exist in the /etc/hosts file on both systems. Refer to your lecture notes as necessary to perform the following steps.

Tasks and Solutions

This section contains the solutions to this exercise.

Task 1 – On the NFS Server

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Edit the /etc/dfs/dfstab file. Add an entry to share the directory that holds man pages.

```
share -o ro /usr/share/man
```

3. Start the NFS server daemons.

```
# svcadm enable svc:/network/nfs/server
```

4. Verify that the NFS server service is online.

```
# svcs -a | grep nfs
```

| | | |
|----------|-------|-----------------------------------|
| disabled | 09:04 | svc:/network/nfs/cbd:default |
| online | 09:04 | svc:/network/nfs/client:default |
| online | 09:04 | svc:/network/nfs/status:default |
| online | 09:04 | svc:/network/nfs/nlockmgr:default |
| online | 09:04 | svc:/network/nfs/mapid:default |
| online | 09:04 | svc:/network/nfs/rquota:default |
| online | 09:04 | svc:/network/nfs/server:default |

Exercise Solutions: Configuring NFS

5. Verify that the /usr/share/man directory 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

There is no output for the dfmounts command.

Task 2 – On the NFS Client

Complete the following steps:

1. Rename the /usr/share/man directory so that you can no longer access the man pages on the client system. Verify that the man pages are not available.

```
# mv /usr/share/man /usr/share/man.orig
# man ls
No manual entry for ls.
#
```

What message does the man command report?

No manual entry for ls.

2. Create a new man directory (/usr/share/man) to use as a mount point.

```
# cd /usr/share
# mkdir man
```

3. Verify that the NFS client service is online.

```
# svcs -a | grep nfs
disabled          18:10:23 svc:/network/nfs/rquota:ticlts
disabled          18:10:24 svc:/network/nfs/rquota:udp
disabled          18:10:56 svc:/network/nfs/server:default
online            18:10:12 svc:/network/nfs/cbd:default
online            18:10:12 svc:/network/nfs/mapid:default
online            18:10:12 svc:/network/nfs/status:default
online            18:10:14 svc:/network/nfs/nlockmgr:default
online            18:10:32 svc:/network/nfs/client:default
```

4. Mount the /usr/share/man directory from the server.

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

5. Verify that the man pages are available.

```
# man ls
```

Are the man pages available?

Yes

6. Use the mount command to display the mount options for the /usr/share/man directory. Record the read and write options that the mount command displays.

```
# mount
```

The ro | rw option for the mount command is read/write (rw) by default.

7. Write a file into the NFS-mounted file system.

```
# touch /usr/share/man/test
```

```
touch: /usr/share/man/test cannot create
```

What is the result of trying to write to the NFS-mounted file system?

You cannot write to the file system.

What conclusion can be reached by this exercise?

Even though the file system mount is read/write, by default, the actual ro | rw permission is read-only, as defined when the directory was shared on the NFS server.

8. Unmount the /usr/share/man directory.

```
# umount /usr/share/man
```

Task 3 – On the NFS Server

Complete the following steps:

1. Unshare the /usr/share/man directory.

```
# unshareall
```

2. Change the share statement in the /etc/dfs/dfstab file for the /usr/share/man directory to read:

```
share -o ro=bogus /usr/share/man
```

3. Share the /usr/share/man directory.

```
# shareall
```

Task 4 – On the NFS Client

Complete the following step:

Attempt to mount the /usr/share/man directory again.

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

What happens?

The client reports the error message:

```
nfs mount: mount: /usr/share/man: Permission denied
```

Task 5 – On the NFS Server

Complete the following steps:

1. Unshare the /usr/share/man directory.

```
# unshareall
```

2. Edit the /etc/dfs/dfstab file to remove the entry for the /usr/share/man directory.

Task 6 – On the NFS Client

Complete the following steps:

1. Return the /usr/share/man directory to its original configuration.

```
# cd /usr/share  
# rmdir man  
# mv man.orig man
```

2. Verify that the man pages are now available.

```
# man ls
```

Module 3

Configuring AutoFS

Objectives

The AutoFS file system provides a mechanism for automatically mounting NFS file systems on demand and for automatically unmounting these file systems after a predetermined period of inactivity. The mount points are specified using local or distributed automount maps.

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

- Describe the fundamentals of the AutoFS file system
- Use automount maps

Introducing the Fundamentals of AutoFS

AutoFS is a file system mechanism that provides automatic mounting using the NFS protocol. AutoFS is a client-side service. The AutoFS file system is initialized by the svc:/system/filesystem/autofs:default services, which runs automatically when a system is booted. This service executes the /lib/svc/method/svc-autofs script. This script runs the automount command, which reads the AutoFS configuration files and also starts the automount daemon automountd. The automountd daemon runs continuously, mounting and unmounting remote directories on an as-needed basis.

Whenever a user on a client computer running the automountd daemon tries to access a remote file or directory, the daemon mounts the remote file system to which that file or directory belongs. This remote file system remains mounted for as long as it is needed. If the remote file system is not accessed for a defined period of time, the automountd daemon automatically unmounts the file system.

The AutoFS service mounts and unmounts file systems as required without any user intervention. The user does not need to use the mount and umount commands and does not need to know the superuser password.

The AutoFS file system enables you to do the following:

- Mount file systems on demand
- Unmount file systems automatically
- Centralize the administration of AutoFS mounts through the use of a name service, which can dramatically reduce administration overhead time
- Create multiple mount resources for read/write or read-only file systems

The automount facility contains three components, as shown in Figure 3-1:

- The AutoFS file system
- The automountd daemon
- The automount command

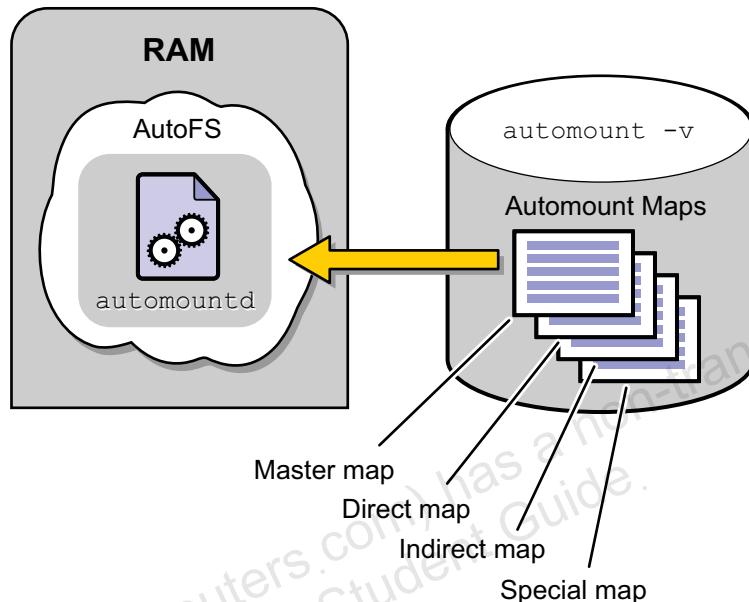


Figure 3-1 The AutoFS Features

AutoFS File System

An AutoFS file system's mount points are defined in the automount maps on the client system. After the AutoFS mount points are set up, activity under the mount points can trigger file systems to be mounted under the mount points. If the automount maps are configured, the AutoFS kernel module monitors mount requests made on the client. If a mount request is made for an AutoFS resource not currently mounted, the AutoFS service calls the automountd daemon, which mounts the requested resource.

The automountd Daemon

The /lib/svc/method/svc-autofs script starts the automountd daemon. The automountd daemon mounts file systems on demand and unmounts idle mount points.



Note – 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.

The automount Command

The automount command, called at system startup time, reads the master map to create the initial set of AutoFS mounts. These AutoFS mounts are not automatically mounted at startup time, they are the points under which file systems are mounted on demand.

Using Automount Maps

The file system resources for automatic mounting are defined in automount maps. Figure 3-2 shows maps defined in the /etc directory.

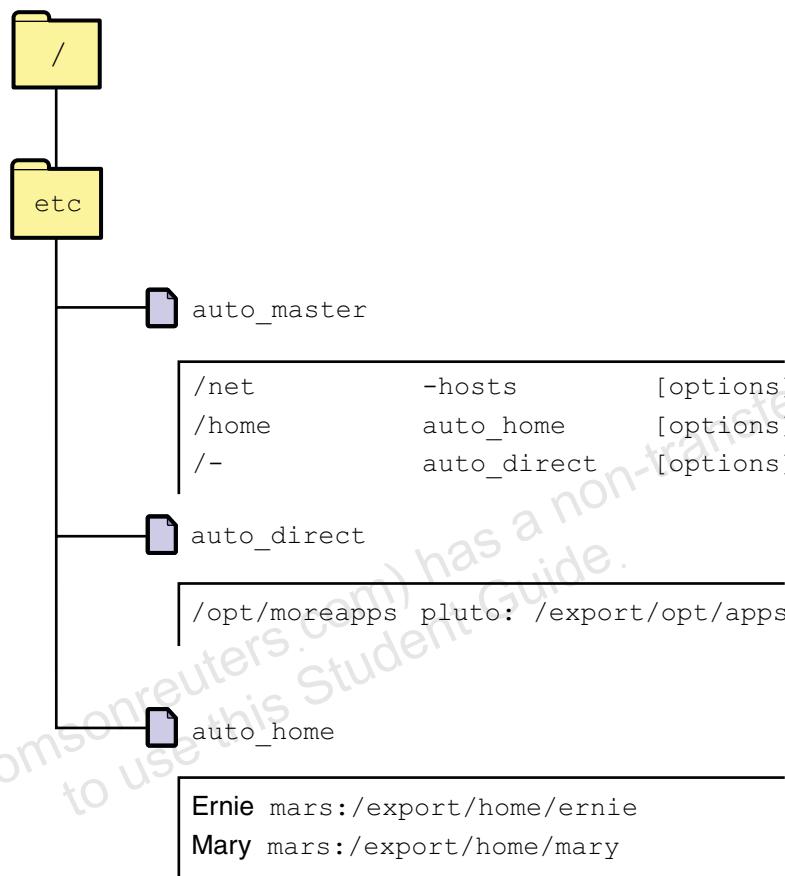


Figure 3-2 Configuring AutoFS Mount Points

The AutoFS map types are:

- Master map – Lists the other maps used for establishing the AutoFS file system. The `automount` command reads this map at boot time.
- 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 NFS servers by using their host names.

Using Automount Maps

The automount maps can be obtained from ASCII data files, NIS maps, NIS+ tables, or from an LDAP database. Together, these maps describe information similar to the information specified in the /etc/vfstab file for remote file resources.

The source for automount maps is determined by the automount entry in the /etc/nsswitch.conf file. For example, the entry:

```
automount: files
```

tells the automount command that it should look in the /etc directory for its configuration information. Using *nis* instead of *files* tells automount to check the NIS maps for its configuration information.

Configuring the Master Map

The auto_master map associates a directory, also called a mount point, with a map. The auto_master map is a master list specifying all the maps that the AutoFS service should check. Names of direct and indirect maps listed in this map refer to files in the /etc directory or to name service databases.

Associating a Mount Point With a Map

The following example shows an /etc/auto_master file.

```
# cat /etc/auto_master
# Master map for automounter
#
+auto_master
/net           -hosts          -nosuid,nobrowse
/home          auto_home       -nobrowse
```

The general syntax for each entry in the auto_master map is:

mount point *map name* *mount options*

where:

mount point

The full path name of a directory. If the directory does not exist, the AutoFS service creates one, if possible.

map name

The name of a direct or indirect map. These maps provide mounting information. A relative path name in this field requires AutoFS to consult the /etc/nsswitch.conf file for the location of the map.

mount options

The general options for the map. The mount options are similar to those used for standard NFS mounts. However, the nobrowse option is an AutoFS-specific mount option.



Note – The plus (+) symbol at the beginning of the +auto_master line in this file directs the automountd daemon to look at the NIS, NIS+, or LDAP databases before it reads the rest of the map. If this line is commented out, only the local files are searched unless the /etc/nsswitch.conf file specifies that NIS, NIS+, or LDAP should be searched.

Identifying Mount Points for Special Maps

There are two mount point entries listed in the default /etc/auto_master file.

```
# cat /etc/auto_master
#
# Copyright 2003 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)auto_master      1.8      03/04/28 SMI"
#
# Master map for automounter
#
+auto_master
/net            -hosts          -nosuid,nobrowse
/home           auto_home       -nobrowse
```

The two special maps are:

| | |
|-------------------|---|
| The -hosts map | Provides access to all resources shared by NFS servers. The resources being shared by a server are mounted below the <code>/net/hostname</code> directory, or, if only the server's IP address is known, below the <code>/net/IPaddress</code> directory. The server does not have to be listed in the hosts database for this mechanism to work. |
| The auto_home map | This map provides the mechanism to allow users to access their centrally located \$HOME directories. |

Using the /net Directory

Shared resources associated with the hosts map entry are mounted below the `/net/hostname` directory. For example, a shared resource named `/documentation` on host `sys42` is mounted by the command:

```
# cd /net/sys42/documentation
```

Using the `cd` command to trigger the automounting of `sys42`'s resource eliminates the need to log in to the system. Any user can mount the resource by executing the command to change to the directory that contains the shared resource. The resource remains mounted until a predetermined time period of inactivity has occurred.

The `-nobrowse` option prevents all the potential mount points from being visible. Only those resources that are actually mounted are visible.

Adding Direct Map Entries

The `/-` entry in the example master map defines a mount point for direct maps.

```
# cat /etc/auto_master
# Master map for automounter
#
+auto_master
/net           -hosts          -nosuid, nobrowse
/home          auto_home        -nobrowse
/-             auto_direct     -ro
```

The `/-` mount point is a pointer that informs the automount facility that the full path names are defined in the file specified by *map_name* (the `/etc/auto_direct` file in this example).



Note – The `/-` entry is not an entry in the default master map. This entry has been added here as an example. The other entries in this example already exist in the `auto_master` file.

Even though the *map_name* entry is specified as `auto_direct`, the automount facility automatically searches for all map-related files in the `/etc` directory; therefore, based upon the automount entry in the `/etc/nsswitch.conf` file, the `auto_direct` file is the `/etc/auto_direct` file. If the `auto_direct` file is to be stored in another directory, the absolute path name to the file should be used.



Note – An NIS or NIS+ master map can have only one direct map entry. A master map that is a local file can have any number of entries.

Creating a Direct Map

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,v6.0
/opt/local     -ro,soft   server2:/export/unbundled
/usr/share/man -ro,soft   server3,server4,server5:/usr/share/man
```

The syntax for direct maps is:

key [*mount-options*] *location*

where:

key The full path name of the mount point for the direct maps.

mount-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 automount feature—multiple locations for a file resource.

The automountd daemon automatically mounts the /usr/share/man directory as needed, from servers server3, server4, or server5, with server proximity and administrator-defined weights determining server selection. If the nearest server fails to respond within the specified timeout period, the next server which responds first is selected.

Map entries that specify multiple file system sources must include the read-only (-ro) option.

Note – Selection criteria for multiple servers, such as server proximity and administrator-defined weights, is defined in the “Replicated File Systems” section of the *automount* man page.



Adding Indirect Map Entries

The /home entry defines a mount point for an indirect map. The map auto_home lists relative path names only. Indirect maps obtain the initial path of the mount point from the master map.

```
# cat /etc/auto_master
# Master map for automounter
#
+auto_master
/net           -hosts          -nosuid, nobrowse
/home          auto_home       -nobrowse
```

The Solaris 2.6 through the Solaris 10 OS releases support browsing of indirect maps and special maps with the -browse option. This support 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 does not provide browser functions for any directory other than those that are currently mounted. The default for this option is -browse.

Creating an Indirect Map

Use the auto_home indirect map to list the location of home directories across the network. 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
mary         mars:/export/home/mary
```

The example /etc/auto_home file implies the following mount points: /home/stevenu, /home/johnnyd, /home/wkd, and /home/mary. Figure 3-3 on page 3-12 shows the /home/mary mount point.

The following describes the syntax for indirect maps:

key [*mount-options*] *location*

Using Automount Maps

where:

- key* Specifies the path name of the mount point relative to the beginning of the path name specified in the master map.
- mount-options* Specifies the options for a given entry.
- location* Specifies the location of the file resource specified in *server:pathname* notation.

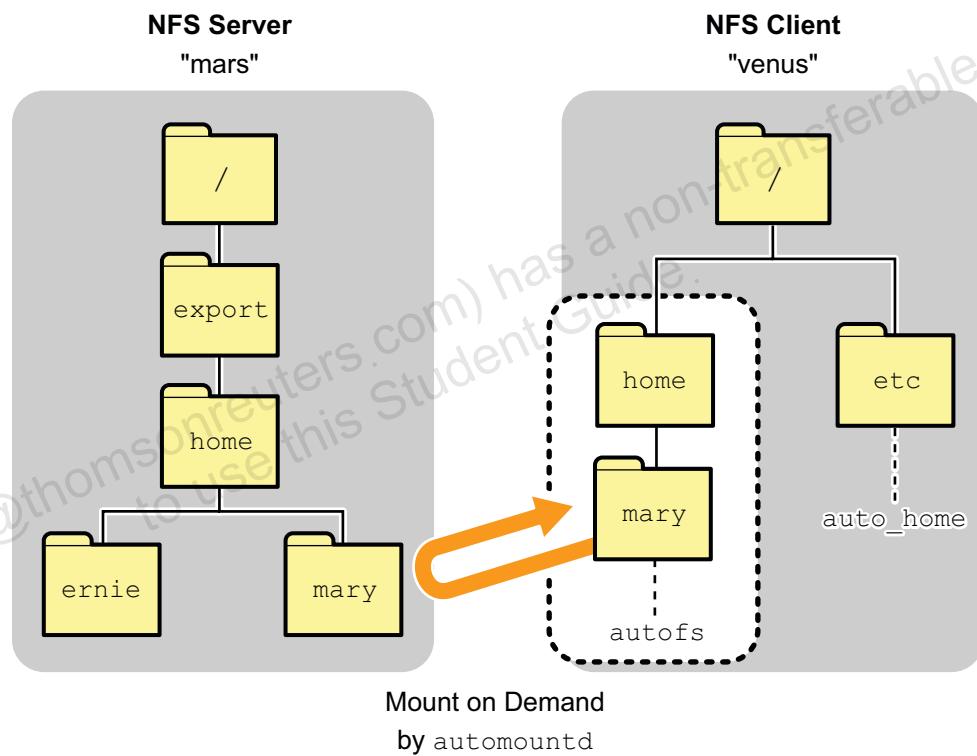


Figure 3-3 The Mount Points

Reducing the auto_home Map to a Single Line

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`, as shown in Figure 3-3.

Figure 3-4 shows that this entry uses the wildcard character (*) to match any key. The wildcard character cannot be used in combination with any other character. The substitution character (&) at the end of the location is replaced with the matched key field. Using wildcard and substitution characters works only when all home directories are on a single server (in this example, server1).

```
*      server1:/export/home/&
```

Figure 3-4 Mounting a Directory on a Local Mount Point

Updating the Automount Maps

When making changes to the master map or creating a direct map, run the `automount` command to make the changes effective.

Running the `automount` Command

The syntax of the command is:

```
automount [-t duration] [-v]
```

where:

`-t duration` Specifies a time, in seconds, that the file system remains mounted when not in use. The default is 600 seconds (10 minutes).

`-v` Specifies verbose mode, which displays output as the `automount` command executes.

You can modify the master map entries or add entries for new maps. However, you must run the `automount` command to make these changes effective.

Using Automount Maps

You do not have to stop and restart the automountd daemon after making changes to existing entries in a direct map, because the daemon is stateless. You can modify existing entries in the direct map at any time. The new information is used when the automountd daemon next accesses the map entry to perform a mount.

Note – If you change the mount point, the first field of the direct map, you must stop and start automountd. If you change any other entry in the direct map then you do not have to restart automountd.

Any modifications to indirect maps are automatically used by the automountd daemon.

A modification is a change to options or resources. A change to the key (the mount point) or a completely new line is an added entry, a deleted entry, or both.

Use Table 3-1 to determine whether you should run (or rerun) the automount command.

Table 3-1 When to Run the automount Command

| Automount Map | Run if the Entry Is Added or Deleted | Run if the Entry Is Modified |
|---------------|--------------------------------------|------------------------------|
| Master map | Yes | Yes |
| Direct map | Yes | No |
| Indirect map | No | No |

Verifying AutoFS Entries in the /etc/mnttab File

The /etc/mnttab file is a file system that provides read-only access to the table of mounted file systems for the current host. Mounting a file system adds an entry to this table. Unmounting a file system removes the entry from this table. Each entry in the table is a line of fields separated by spaces in the form of:

special *mount_point* *fstype* *options* *time*

where:

special The name of the resource to be mounted

| | |
|--------------------|--|
| <i>mount_point</i> | The path name of the directory on which the file system is mounted |
| <i>fstype</i> | The type of file system |
| <i>options</i> | The mount options |
| <i>time</i> | The time at which the file system was mounted |

Using Automount Maps

You can display the /etc/mnttab file to obtain a snapshot of the mounted file systems, including those mounted as an AutoFS file system type.

```
# grep autofs /etc/mnttab
-hosts /net    autofs nosuid,indirect,ignore,nobrowse,dev=4e00001
1099678245
auto_home      /home   autofs indirect,ignore,nobrowse,dev=4e00002
1099678245
-hosts /net/sys-02/rdbms_files autofs
nosuid,ignore,nest,nobrowse,dev=4e000031099679619
-hosts /net/sys-02/usr autofs nosuid,ignore,nest,nobrowse,dev=4e00004
1099679619
```

Stopping and Starting the Automount System

The autofs service is enabled or disabled automatically as the system transitions between run levels, or you can enable or disable the service manually from the command line.

Stopping the Automount System

When the autofs service is disabled, it performs a forced unmount of all AutoFS file systems, and it then kills the automountd daemon.

The autofs service is disabled automatically when transitioning to the single-user milestone.

To disable the service, become superuser, and kill the automountd daemon by typing the following command:

```
# svcadm disable svc:/system/filesystem/autofs
```

Starting the Automount System

When the `autofs` service is enabled, the service management facility starts the `automountd` daemon, and then it runs the `automount` utility as a background task.

The service starts automatically when transitioning to multi-user milestone.

To enable the service manually, become superuser, and start the `automountd` daemon by performing the command:

```
# svcadm enable svc:/system/filesystem/autofs
```

Exercise: Using the Automount Facility

In this exercise, you use the automount facility to automatically mount man pages and to mount 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 system. Refer to the lecture notes as necessary to perform the steps listed.

Tasks

Perform the following tasks.

Task 1 – On the Server System

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Edit the /etc/dfs/dfstab file, and add a line to share the man pages.
3. Use the pgrep command to check if the mounted daemon is running.
 - If the mounted daemon is not running, start it.
 - If the mounted daemon is running, share the new directory.

Task 2 – On the Client System

Complete the following steps:

1. Rename the /usr/share/man directory so that you cannot view the man pages installed on the client system.
2. Make a backup copy of the /etc/auto_master file called /etc/_auto_master and then edit the /etc/auto_master file and add an entry for a direct map.
3. Use the vi editor to create a new file called /etc/auto_direct, and add an entry to the file to share the man pages.
4. Run the automount command to update the list of directories managed by the automountd daemon.

5. Test the configuration, and verify that a mount for the /usr/share/man directory exists after accessing the man pages.

What did you observe to indicate that the automount operation was successful?

Task 3 – On the Server System

Complete the following steps:

1. Verify that the /export/home directory exists. If it does not exist, create it.
2. Add a user account with the following characteristics:
 - User ID: 3001
 - Primary group: 10
 - Home directory: /export/home/usera
 - Login shell:/bin/ksh
 - User name: usera
3. Configure the password mechanism for usera so that this user must assign a new password (123pass) at next login. Do this by executing the passwd -f usera command.

Task 4 – On the Client System

Complete the following steps:

1. Verify that the /export/home directory exists. If it does not, create it.
2. Add a user account with the following characteristics:
 - User ID: 3001
 - Primary group: 10
 - Home directory: /export/home/usera
 - Login shell: /bin/ksh
 - User name: usera
3. Configure the password mechanism for usera so that this user must assign a new password (123pass) at next login. Do this by executing the passwd -f usera command.

Task 5 – On Both Systems

Complete the following steps:

1. Edit the /etc/passwd file, and change the home directory for usera from the /export/home/usera directory to /home/usera.
2. Make a backup copy of the /etc/auto_home file and call it /etc/_auto_home.
3. Edit the /etc/auto_home file and add the following line:
usera server:/export/home/usera

Task 6 – On the Server System

Complete the following steps:

1. Edit the /etc/dfs/dfstab file, and add a line to share the /export/home directory.
2. Use the pgrep command to check if the mountd daemon is running.
 - If the mountd daemon is not running, start it.
 - If the mountd daemon is running, share the new directory.

Task 7 – On Both Systems

Complete the following step:

Log in as the new user.

Do both systems automatically mount the new user's home directory?

Exit from the usera login.

Which directory is mounted, and what is the mount point:

- On the server?
-

- On the client?
-

Task 8 – On the Client System

Complete the following steps:

1. Remove the entry for usera from the /etc/auto_home map.
2. Remove the entry for the auto_direct map from the /etc/auto_master map and remove the /etc/auto_direct file you created earlier.
3. Reboot the client.
4. Remove the /usr/share/man directory.
5. Rename the /usr/share/man.orig directory to /usr/share/man.

Task 9 – On the Server System

Complete the following steps:

1. After the client reboots as described in step 3 of “Task 8 – On the Client System”, remove the entry for usera from the /etc/auto_home map.
2. Remove the entries from /etc/dfs/dfstab file.
3. Unshare mounted directories.

Exercise Summary

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

- Experiences
- Interpretations
- Conclusions
- Applications



Exercise Solutions: Using the Automount Facility

In this exercise, you use the automount facility to automatically mount man pages and to mount 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 system. Refer to the lecture notes as necessary to perform the steps listed.

Tasks and Solutions

The following section provides the tasks with their solutions.

Task 1 – On the Server System

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Edit the /etc/dfs/dfstab file, and add a line to share the man pages.
`share -o ro /usr/share/man`
3. Use the pgrep command to check if the mountd daemon is running.

```
# pgrep -f1 mountd
820 /usr/lib/autofs/automountd
819 /usr/lib/autofs/automountd
1021 /usr/lib/nfs/mountd
```

- If the mountd daemon is not running, start it.

```
# svcadm enable svc:/network/nfs/server
● If the mountd daemon is running, share the new directory.
```

```
# shareall
```

Task 2 – On the Client System

Complete the following steps:

Exercise Solutions: Using the Automount Facility

1. Rename the /usr/share/man directory so that you cannot view the man pages installed on the client system.

```
# cd /usr/share/  
# mv man man.orig
```

2. Make a backup copy of the /etc/auto_master file called /etc/_auto_master and then edit the /etc/auto_master file and add an entry for a direct map.

```
# cp /etc/auto_master /etc/_auto_master  
# vi /etc/auto_master  
/- auto_direct
```

3. Use the vi editor to create a new file called /etc/auto_direct, and add an entry to the file to share the man pages.

```
# vi /etc/auto_direct  
/usr/share/man    server:/usr/share/man
```

4. Run the automount command to update the list of directories managed by the automountd daemon.

```
# automount -v
```

5. Test the configuration, and verify that a mount for the /usr/share/man directory exists after accessing the man pages.

```
# man ls  
<-- output from man command -->  
# mount | grep man  
/usr/share/man on sys44:/usr/share/man  
remote/read/write/setuid/dev=42c0003 on Thu Jan 6 08:07:26 2005
```

What did you observe to indicate that the automount operation was successful?

This operation should automatically mount the directory in which the manuals are stored. In other words, the man command should work.

Task 3 – On the Server System

Complete the following steps:

1. Verify that the /export/home directory exists. If it does not exist, create it.

```
# ls /export/home
```



Note – Perform the next command if the /export/home directory does not exist.

```
# mkdir /export/home
```

2. Add a user account with the following characteristics:

- User ID: 3001
- Primary group: 10
- Home directory: /export/home/usera
- Login shell:/bin/ksh
- User name: usera

```
# useradd -u 3001 -g 10 -m -d /export/home/usera -s /bin/ksh usera
```

3. Configure the password mechanism for usera so that this user must assign a new password (123pass) at next login. Do this by executing the passwd -f usera command.

Task 4 – On the Client System

Complete the following steps:

1. Verify that the /export/home directory exists. If it does not, create it.

```
# ls /export
# mkdir /export/home
```

2. Add a user account with the following characteristics:

- User ID: 3001
- Primary group: 10
- Home directory: /export/home/usera
- Login shell: /bin/ksh
- User name: usera

```
# useradd -u 3001 -g 10 -m -d /export/home/usera -s /bin/ksh usera
```

3. Configure the password mechanism for usera so that this user must assign a new password (123pass) at next login. Do this by executing the passwd -f usera command.

Task 5 – On Both Systems

Complete the following steps:

Exercise Solutions: Using the Automount Facility

1. Edit the /etc/passwd file, and change the home directory for usera from the /export/home/usera directory to /home/usera.

```
# vi /etc/passwd
```

2. Make a backup copy of the /etc/auto_home file and call it /etc/_auto_home.

```
# cp /etc/auto_home /etc/_auto_home
```

3. Edit the /etc/auto_home file and add the following line:

```
usera      server:/export/home/usera
```

Task 6 – On the Server System

Complete the following steps:

1. Edit the /etc/dfs/dfstab file, and add a line to share the /export/home directory.

```
share /export/home
```

2. Use the pgrep command to check if the mountd daemon is running.

```
# pgrep -fl mountd
```

```
820 /usr/lib/autofs/automountd  
819 /usr/lib/autofs/automountd  
1021 /usr/lib/nfs/mountd
```

- If the mountd daemon is not running, start it.

```
# svcadm enable svc:/network/nfs/server
```

- If the mountd daemon is running, share the new directory.

```
# shareall
```

Task 7 – On Both Systems

Complete the following step:

Log in as the new user.

```
# su - usera
```

Do both systems automatically mount the new user's home directory?

Yes, this should work.

Exit from the usera login.

```
# exit
```

Which directory is mounted, and what is the mount point:

mount

- On the server?

The /home/username directory is mounted on the /export/home/username directory.

- On the client?

The /home/username directory is mounted on the server:/export/home/username directory.

Task 8 – On the Client System

Complete the following steps:

1. Remove the entry for usera from the /etc/auto_home map.
2. Remove the entry for the auto_direct map from the /etc/auto_master map and remove the /etc/auto_direct file you created earlier.
3. Reboot the client.

init 6

4. Remove the /usr/share/man directory.

rmdir /usr/share/man

5. Rename the /usr/share/man.orig directory to /usr/share/man.

mv /usr/share/man.orig /usr/share/man

Task 9 – On the Server System

Complete the following steps:

1. After the client reboots as described in step 3 of “Task 8 – On the Client System”, remove the entry for usera from the /etc/auto_home map.
2. Remove the entries from /etc/dfs/dfstab file.
3. Unshare mounted directories.

unshareall

Notes:

Module 4

Describing RAID

Objectives

A redundant array of independent disks (RAID) configuration enables you to expand the characteristics of a storage volume beyond the physical limitations of a single disk. You can use a RAID configuration to increase storage capacity as well as to improve disk performance and fault tolerance.

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

- Describe RAID

Introducing RAID

RAID is a classification of methods to back up and to store data on multiple disk drives. There are six levels of RAID as well as a non-redundant array of independent disks (RAID 0). The Solaris Volume Manager software uses metadevices, which are product-specific definitions of logical storage volumes, to implement RAID 0, RAID 1, RAID 1+0 and RAID 5:

- RAID 0: Non-redundant disk array (concatenation and striping)
- RAID 1: Mirrored disk array
- RAID 5: Block-interleaved striping with distributed-parity

RAID 0

RAID-0 volumes, including both stripes and concatenations, are composed of slices and let you expand disk storage capacity. You can either use RAID-0 volumes directly or use the volumes as the building blocks for RAID-1 volumes (mirrors). There are two types of RAID-0 volumes:

- Concatenated volumes (or concatenations)
A concatenated volume writes data to the first available slice. When the first slice is full, the volume writes data to the next available slice.
- Striped volumes (or stripes)
A stripe distributes data equally across all slices in the stripe.

RAID-0 volumes allow you to expand disk storage capacity efficiently. These volumes do not provide data redundancy. If a single slice fails on a RAID-0 volume, the entire volume is inaccessible.

Concatenated Volumes

Figure 4-1 shows that in a concatenated RAID 0 volume, data is organized across disk slices, forming one logical storage unit.

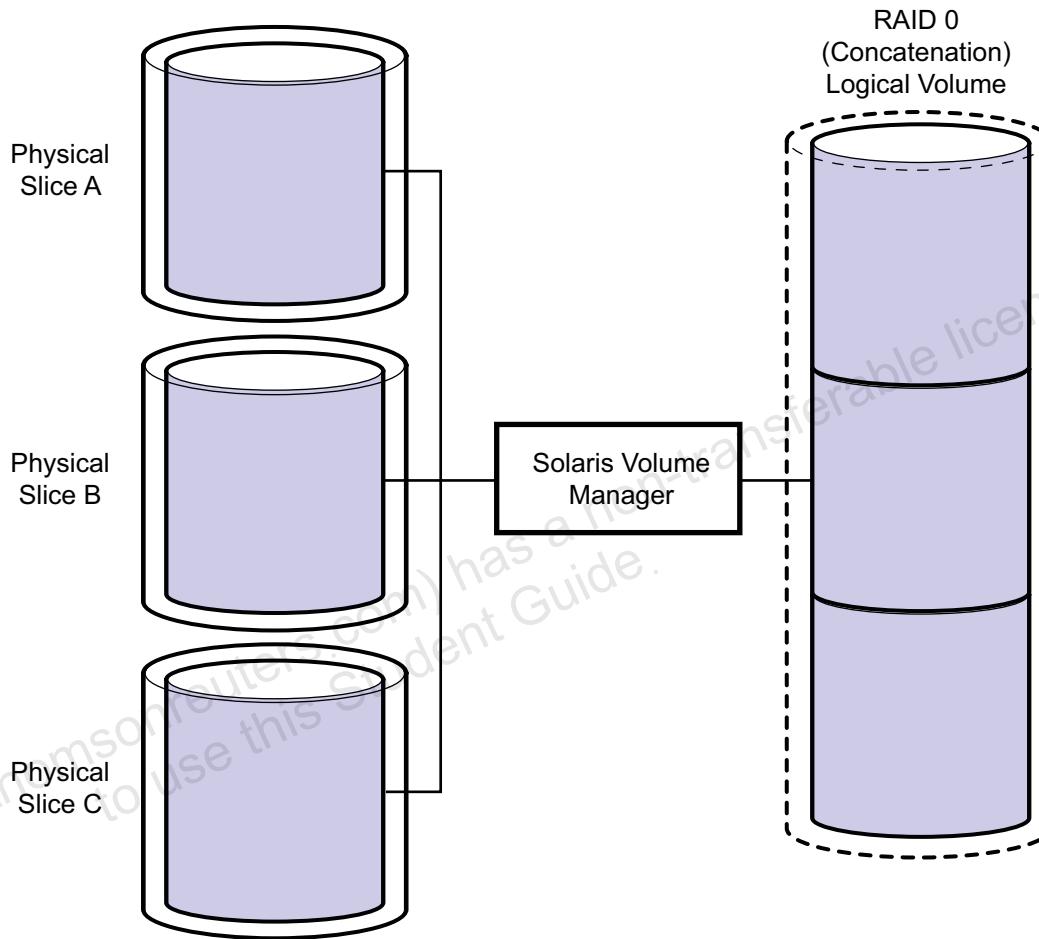


Figure 4-1 RAID-0 Concatenation

A concatenation combines the capacities of several slices to get a larger storage capacity. You can add more slices to the concatenation as the demand for storage increases. You can add slices at anytime, even if other slices are currently active.

The default behavior of concatenated RAID-0 volumes is to fill a physical component within the volume before beginning to store data on subsequent components within the concatenated volume. However, the default behavior of UFS file systems within the Solaris OS is to distribute the load across devices assigned to the volume containing a file system. This behavior makes it seem that concatenated RAID-0 volumes distribute data across the components of the volume in a round-robin, interlaced fashion.

This interlacing of data is a function of the UFS file system that is mounted in the concatenated volume and is not a function of the concatenated volume itself.

You can also use a concatenation to expand any active and mounted UFS file system without having to bring down the system. The capacity of a concatenation is the total size of all the slices in the concatenation.

Striped Volumes

Figure 4-2 shows the arrangement of a striped RAID-0 volume. A RAID 0 volume configured as a stripe arranges data across two or more slices. Striping alternates equally-sized segments of data across two or more slices, forming one logical storage unit. These segments are interleaved round-robin, so that the combined space is created alternately from each slice.

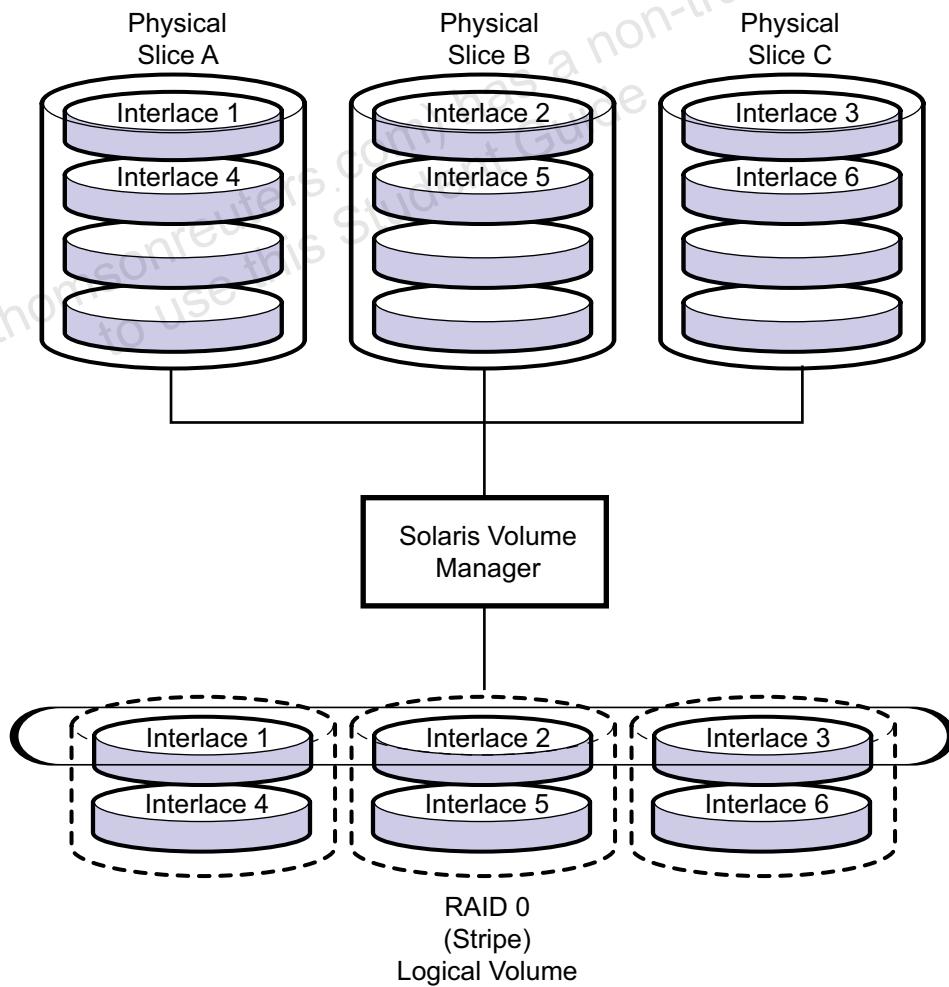


Figure 4-2 RAID-0 Stripe

Striping enables parallel data access because data can be retrieved from multiple disks at the same time. Parallel access increases input/output (I/O) throughput because multiple disks in the volume are busy servicing I/O requests simultaneously.

You cannot convert an existing file system directly to a stripe. You must first back up the file system, create the stripe, and then restore the file system to the stripe.

For sequential I/O operations on a stripe, the Solaris Volume Manager software reads all the blocks in an *interlace*. An interlace is a grouped segment of blocks on a particular slice. The Solaris Volume Manager software then reads all the blocks in the interlace on the second slice, and so on.

An interlace is the size of the logical data chunks on a stripe. Depending on the application, different interlace values can increase performance for your configuration. The performance increase comes from several disk head-arm assemblies (HDAs) concurrently executing I/O operations. When the I/O request is larger than the interlace size, you might get better performance.

When you create a stripe, you can set the interlace value. After you create the stripe, you cannot change the interlace value. You could back up the data on it, delete the stripe, create a new stripe with a new interlace value, and then restore the data.

RAID 1

RAID-1 volumes, also known as mirror volumes in the Solaris Volume Manager software, are typically composed of RAID-0 volumes and provide the advantage of data redundancy. The disadvantage is the higher cost incurred by requiring two RAID-1 devices wherever a single RAID-0 device is mirrored. Typical topics to be considered when configuring mirrors are:

- Trade-offs when using mirrors
- Uses of multiple submirrors
- RAID 0+1
- RAID 1+0
- Mirror read, write, and synchronization options
- Mirror configuration guidelines

Trade-Offs When Using Mirrors

A RAID-1 (mirror) volume maintains identical copies of the data in RAID-0 volumes. Mirroring requires more disks. You need at least twice as much disk space as the amount of data to be mirrored.

After configuring a mirror, you can use it as if it were a physical slice. With multiple copies of data available, data access time is reduced if the mirror read policy is properly configured. Mirror read and write policies are described in detail later in this module.

You can mirror any file system, including existing file systems. You can also use a mirror for any application, such as a database.

Using Multiple Submirrors

A mirror is made of two or more RAID-0 volumes configured as either stripes or concatenations. The mirrored RAID-0 volumes are called submirrors. A mirror consisting of two submirrors is known as a two-way mirror, while a mirror consisting of three submirrors is known as a three-way mirror.

Creating a two-way mirror is usually sufficient for data redundancy. Creating a third submirror enables you to make online backups without losing data redundancy while one submirror is offline for the backup.

When a submirror is offline, it is in a read-only mode. The Solaris Volume Manager software tracks all the changes written to the online submirror. When the submirror is brought back online, only the newly written portions are resynchronized. Other reasons for taking the submirror offline include troubleshooting, and repair.

You can attach or detach a submirror from a mirror at any time, though at least one submirror must remain attached to the mirror at all times. Usually, you begin the creation of a mirror with only a single submirror, after which you can attach additional submirrors, as shown in Figure 4-3.

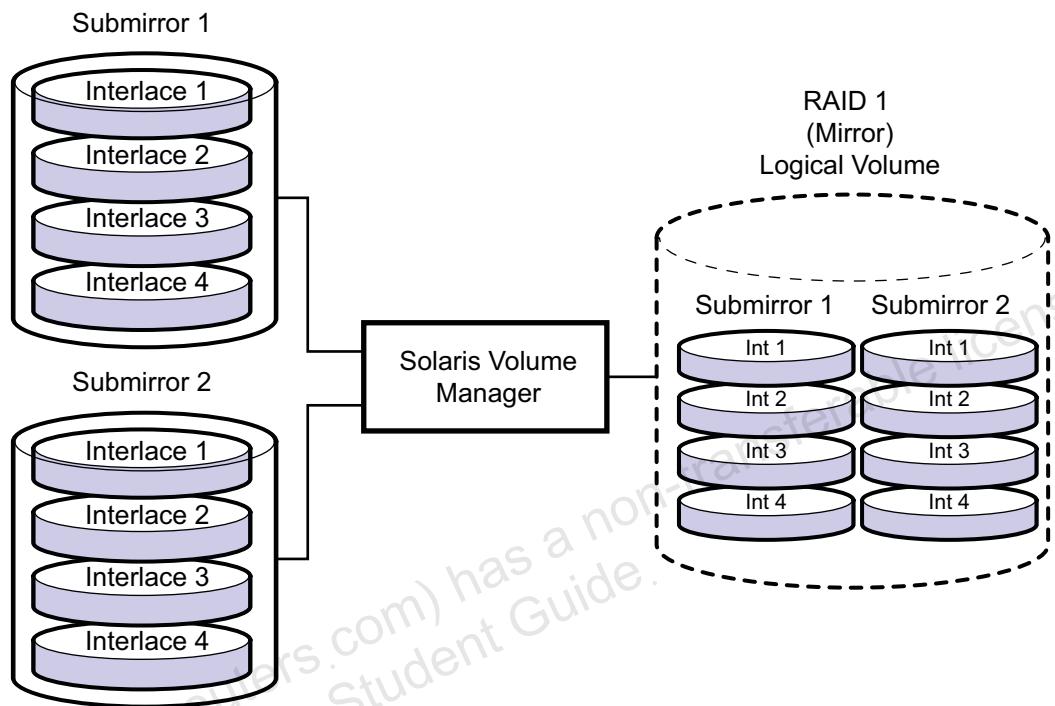


Figure 4-3 RAID-1 Mirror

The Solaris Volume Manager software makes duplicate copies of the data located on multiple physical disks. The Solaris Volume Manager software presents one virtual disk to the application. All disk writes are duplicated, and disk reads come from one of the underlying submirrors. If the submirrors are not of equal size, the total capacity of the mirror is limited by the size of the smallest submirror.

RAID 0+1

In RAID-0+1 volumes, stripes are mirrored to each other. In a pure RAID-0+1 configuration, the failure of one slice would cause the failure of the whole submirror.

Figure 4-4 shows an example of a RAID-0+1 configuration. A failure in slice A, B, or C causes a failure of the entire Submirror 1. A failure in slice D, E, or F causes a failure of the entire Submirror 2. One failure in each submirror of the RAID 0+1 mirror causes a failure of the entire mirror.

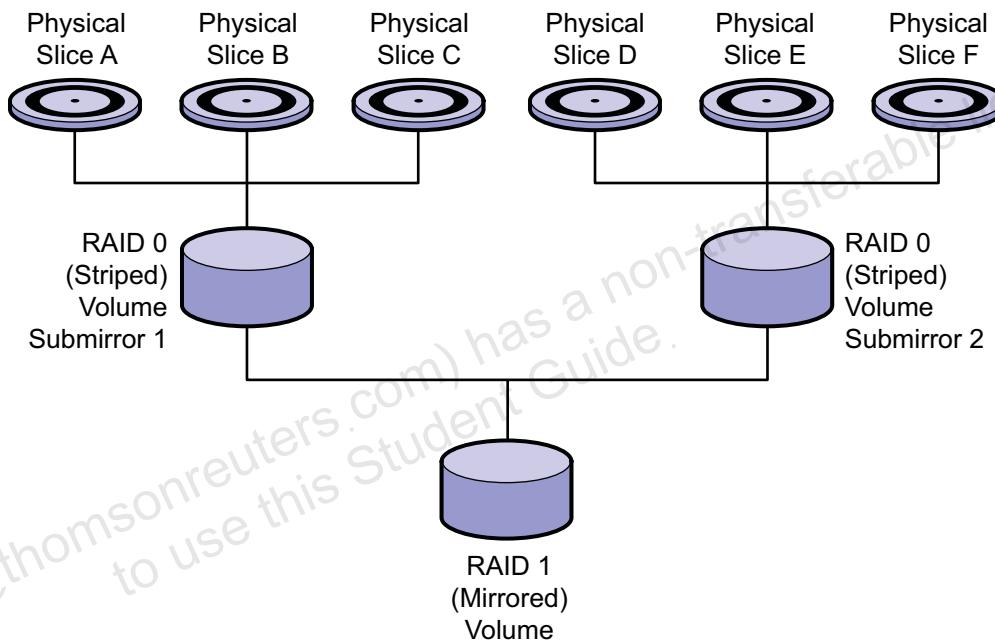


Figure 4-4 RAID-0+1 Mirror of Stripes

RAID 1+0

RAID-1+0 volumes consist of multiple mirrors striped together. RAID 1+0 provides greater data security, because a failure of a single physical disk slice causes a failure for only one half of one of the submirrors, leaving most of the configuration's redundancy intact.

Figure 4-5 shows an example of a RAID-1+0 volume configuration. This example consists of three slices. Each of these three slices mirrors itself. The RAID 0 stripe can tolerate three simultaneous physical slice failures, one in each RAID-1 mirror, before the entire RAID-0 stripe is considered to have failed. This is a more fault-tolerant configuration, as compared with the RAID-0+1 mirror. If both submirrors in any one of the mirrors fail, one third of the data is lost, and the RAID-1+0 volume is also considered failed.

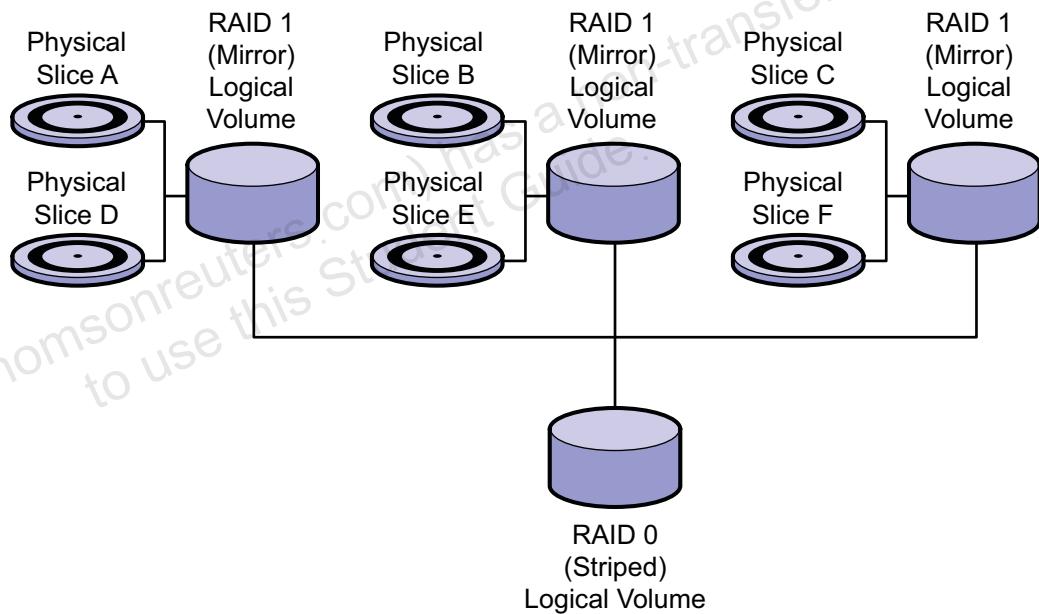


Figure 4-5 RAID 1+0 Stripe of Mirrors



Mirror Options

Mirror performance can be modified by using the following options:

- Mirror read policy
- Mirror write policy

Note – The mirror options listed here are representative of the options presented when configuring RAID-1 mirrors using the Solaris Volume Manager software.

You can define mirror options when you initially create the mirror or after you set up the mirror. You can distribute the load across the submirrors to improve read performance. Table 4-1 describes the configurable mirror read policies.

Table 4-1 Mirror Read Policies

| Read Policy | Description |
|-----------------------|--|
| Round Robin (default) | Balances the load across the submirrors |
| Geometric | Enables the system to divide reads among submirrors on the basis of a logical disk block address |
| First | Directs all reads to the first submirror |

You can improve write performance by replicating all submirrors simultaneously. If a failure occurs during this write, all submirrors will be in an unknown state. Table 4-2 describes the configurable mirror write policies.

Table 4-2 Mirror Write Policies

| Write Policy | Description |
|--------------------|---|
| Parallel (Default) | Replicates a write to a mirror, and dispatches the write to all of the submirrors simultaneously |
| Serial | Specifies that writes to one submirror must complete before initiating writes to the next submirror |

When a submirror is offline, any writes to the mirror are tracked in a dirty region log. When the submirror is brought back online, those regions must be updated or resynchronized.

Mirror Configuration Guidelines

The general configuration guidelines for configuring Solaris Volume Manager software mirrors are:

- Keep the slices of different submirrors on different disks and on different controllers for the best data protection. Organizing submirrors across separate controllers reduces the impact of a single controller failure and also improves mirror performance.
- Use the same type of disks and controllers in a single mirror. Particularly in old Small Computer System Interface (SCSI) devices different models or brands of disks or controllers can vary in performance. Different performance levels can lead to a decrease in overall performance.
- Use submirrors of the same size to reduce unused disk space.
- Mount the mirror device directly. Do not try and mount a submirror directly, unless it is offline and mounted as read-only. Do not mount a slice that is part of a submirror, or you might destroy data and crash the system.
- Mirroring improves read performance, but reduces write performance. Mirroring improves read performance only in multi-threaded or asynchronous I/O situations. There is no performance gain if there is only a single thread reading from the volume.
- Experiment with the mirror read policies to improve performance. For example, using the Solaris Volume Manager software, the default read mode is to alternate reads using a round-robin method among the disks. This mode is the default because it works best for UFS multiuser, multiprocess activity.
- In some cases, the geometric read option improves performance by minimizing head motion and access time. This option is most effective when there is only one slice per disk, when only one process at a time is using the file system, when I/O patterns are sequential, or when all accesses are read.

- Use the `swap -l` command to check for all swap devices. Mirror the slices specified as swap separately.
- Use only similarly configured submirrors within a mirror. In particular, if you create a mirror with an unlabeled submirror, you cannot attach any submirrors that contain disk labels.

RAID 5

RAID-5 volumes are striped volumes that use a distributed parity scheme for data protection. To fully understand RAID-5 volumes, you must understand each of the following:

- Standard RAID-5 volume
- Requirements for RAID-5 volumes
- Suggestions for RAID-5 volumes

Standard RAID-5 Volume

RAID level 5 is similar to striping in that data is distributed across a set of disks. The difference between a RAID level 5 and striping is that in the RAID level 5, parity data is also distributed across the same set of disks. When a disk fails, lost data from the failing disk is rebuilt on the failed volume from the other disks using the distributed data and parity information stored on the remaining (unfailed) disks in the RAID-5 volume.

A RAID-5 volume uses a storage capacity equivalent to one slice to store parity information from the remainder of the RAID-5 volume's slices. The parity information is distributed across all slices in the volume. Like a mirror, a RAID-5 volume increases data availability, but minimizes hardware cost. You cannot use a RAID-5 volume for the root (/) directory, the /usr directory, swap space, or existing file systems because the metadevice software is not loaded early enough in the Solaris OS boot process.

Figure 4-6 shows that the first three data interlaces are written to slices A, B, and C. The next item written is parity to Drive D. The pattern of writing data and parity results in both data and parity spread across all disks in the RAID-5 volume. You can read each drive independently. The parity protects against a single disk failure. In “RAID-5 Distributed Parity”, if each disk were 2 Gbytes, the total capacity of the RAID-5 volume would be 6 Gbytes. Parity information occupies the space equivalent to one drive.

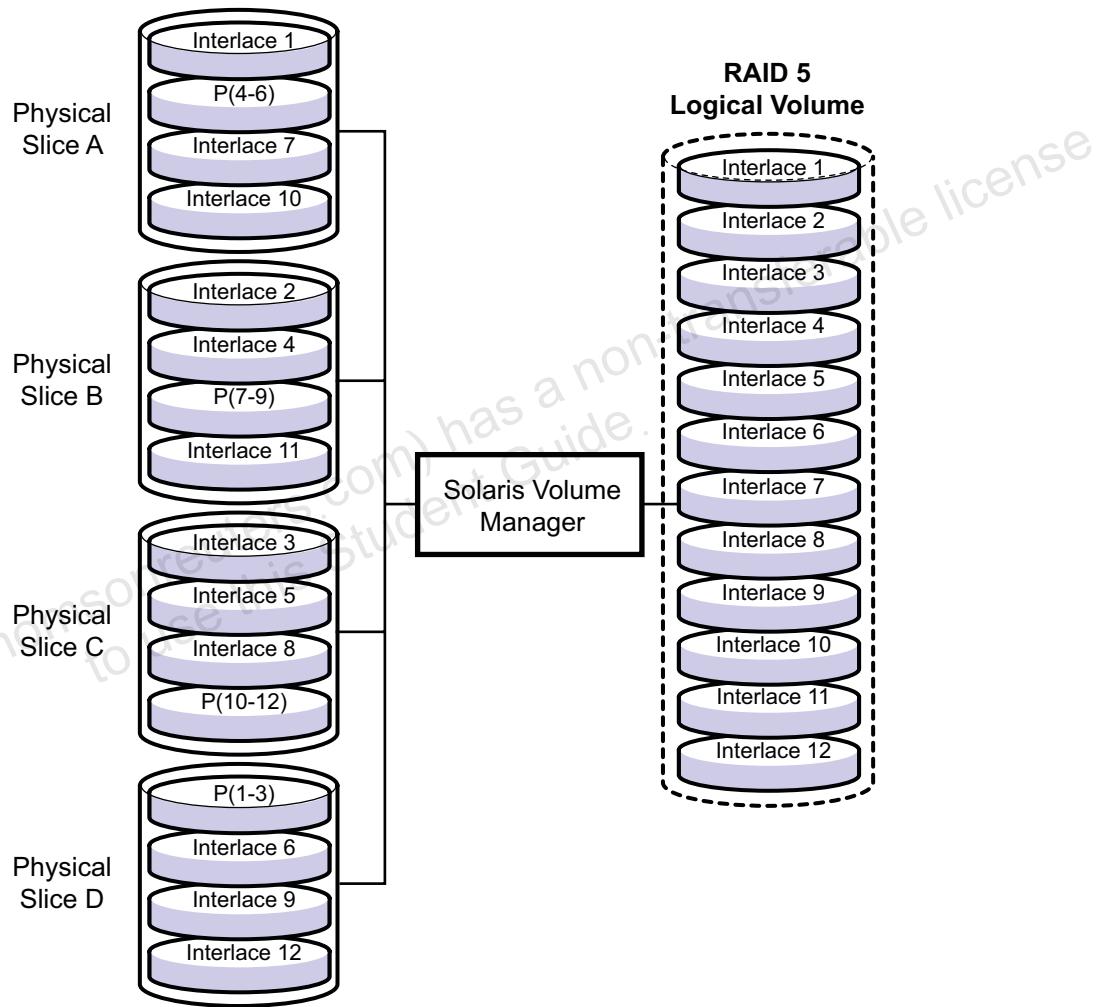


Figure 4-6 RAID-5 Distributed Parity

Requirements for RAID-5 Volumes

The general configuration guidelines for configuring RAID-5 volumes are:

- Create a RAID-5 volume with a minimum of three slices. The more slices a RAID-5 volume contains, the longer read and write operations take when a slice fails.
- Do not stripe, concatenate, or mirror RAID-5 volumes.
- Do not create a RAID-5 volume from a slice that contains an existing file system, because you will erase the data during the RAID-5 initialization process.
- When you create a RAID-5 volume, you can define the interlace value. If you do not specify a value, a default value of 16 Kbytes is assigned.
- A RAID-5 volume (with no hot spares) can only handle a single slice failure.
- To optimize performance, use slices across separate controllers when creating RAID-5 volumes.
- Use disk slices of the same size. Creating a RAID-5 volume of different-sized slices results in unused disk space on the larger slices.

Suggestions for RAID 5 Volumes

The following general suggestions can help avoid common performance problems when using RAID-5 volumes:

- Because of the complexity of parity calculations, volumes with greater than about 20 percent writes should probably not be configured as RAID-5 volumes. If data redundancy on a write-heavy volume is needed, consider mirroring.
- If the slices in the RAID-5 volume reside on different controllers and the accesses to the volume are primarily large sequential accesses, then setting the interlace value to 32 Kbytes might improve performance.

Storage Configurations

When planning your storage management configuration, keep in mind that for any given application there are trade-offs in performance, availability, and hardware costs. You might need to experiment with the different variables to determine what works best for your configuration. A few categories of information that you must address during the storage planning phase are:

- General storage guidelines
- Determining storage characteristics
- Storage performance guidelines

Note – While adding more drives to your configuration increases redundancy, it also increases your overall Mean Time Between Failure (MTBF) by having more hardware.



Storage Characteristics

When you classify storage characteristics, you provide guidelines for working with the Solaris Volume Manager software RAID-0 (concatenation and stripe) volumes, RAID-1 (mirror) volumes, and RAID-5 (striping with distributed parity) volumes.

While building your storage management plan, decide what types of storage devices to use. The storage characteristics guidelines help you compare and contrast the various storage mechanisms and also help you choose the best storage device.



Note – The storage mechanisms listed in Table 4-3 are not mutually exclusive. You can use them in combination to meet multiple goals. For example, you could create a RAID-1 volume for redundancy, and then create soft partitions on it to increase the number of possible discrete file systems.

Table 4-3 Choosing Storage Mechanisms

| Feature | RAID-0 Concatenation | RAID-0 Stripe | RAID-1 Mirror | RAID-5 Stripe With Parity |
|----------------------------|----------------------|---------------|----------------------------------|---------------------------|
| Redundant data | No | No | Yes | Yes |
| Improved read performance | No | Yes | Depends on the underlying device | Yes |
| Improved write performance | No | Yes | No | No |

You must consider many factors when optimizing redundant storage. Table 4-4 compares RAID-1 and RAID-5 volumes for the speed of write operations, random read operations, and the overall cost of the underlying hardware.

Table 4-4 Optimizing Redundant Storage

| Factors | RAID 1 (Mirror) | RAID 5 | Non-Redundant |
|----------------------------|-----------------|--------|---------------|
| Write operations | Faster | Slower | Neutral |
| Random read | Slower | Faster | Neutral |
| Hardware cost | Highest | Higher | Lowest |
| Performance during failure | Best | Poor | Data loss |

General Storage Guidelines

The general configuration guidelines for planning your storage configuration are:

- RAID-0 devices (stripes and concatenations) do not provide data redundancy.
- Concatenation works well for small, random I/O.
- Striping performs well for large, sequential I/O and for random I/O distributions.
- Mirroring improves read performance
- Because of the read-modify-write property of RAID-5 volumes, volumes with greater than about 20-percent writes should probably not be configured as RAID 5. In these write intensive situations, consider mirroring if data protection is required.
- RAID 5 writes are not as fast as mirrored writes, and mirrored writes are not as fast as unprotected writes.

Performance Guidelines

When designing your storage configuration, consider the following performance guidelines:

- Whenever possible, distribute storage devices across multiple I/O controllers, cables, and devices.
- Striping generally has the best performance, but it offers no data protection. For write-intensive applications, RAID 1 performs better than RAID 5.
- RAID-1 and RAID-5 volumes both increase data availability. Mirroring improves random read performance.
- RAID-5 volumes have a lower hardware cost than RAID-1 volumes, while RAID-0 volumes have no additional hardware cost.
- Identify the most frequently accessed data, and increase the access bandwidth for that data with mirroring or striping.
- Both stripes and RAID-5 volumes distribute data across multiple disk drives and help balance the I/O load. You can also use RAID-1 volumes to help balance the I/O load.

- Use available performance monitoring capabilities and generic tools, such as the `iostat` command, to identify the most frequently accessed data. Then increase the “access bandwidth” to the frequently accessed data, by striping RAID-1 volumes or RAID-5 volumes.
- A stripe’s performance is better than that of a RAID 5 volume, but stripes do not provide data redundancy.
- RAID 5 volume performance is lower than stripe performance for write operations, because the RAID-5 volume requires multiple I/O operations to calculate and store the parity.
- For raw random I/O reads, the stripe and the RAID-5 volume are comparable. Both the stripe and RAID-5 volume split the data across multiple disks, and the RAID-5 volume parity calculations are not a factor in reads, except after a component failure.

Module 5

Configuring Solaris Volume Manager Software

Objectives

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

- Describe Solaris Volume Manager software concepts
- Build a RAID-0 (concatenated) volume
- Build a RAID-1 (mirror) volume for the root (/) file system

Introducing Solaris Volume Manager Software Concepts

The Solaris Volume Manager software lets you manage large numbers of disks and the data on those disks. Although there are many ways to use the Solaris Volume Manager software, most tasks include:

- Increasing storage capacity
- Increasing data availability
- Making the administration of large storage devices easier

In some instances, the Solaris Volume Manager software can also improve I/O performance.

Logical Volume

The Solaris Volume Manager software uses virtual disks called logical volumes to manage physical disks and their associated data. Historically, a logical volume is functionally identical to a physical slice. However, a logical volume can span multiple disk *members*. The Solaris Volume Manager software converts I/O requests directed at a volume into I/O requests to the underlying member disks.

You can create the Solaris Volume Manager software volumes from slices (disk partitions) or from other Solaris Volume Manager software volumes.

You build and modify volumes using command-line utilities in the Solaris Volume Manager software.

To create more storage capacity as a single volume, you can use the Solaris Volume Manager software to make the system treat a collection of many small slices as one large slice or device. After creating a large volume from these slices, you can immediately begin by using it just as any other slice or device.

The Solaris Volume Manager software can increase the reliability and availability of data by using RAID-1 volumes and RAID-5 volumes. Solaris Volume Manager software hot spares provide another level of data availability for RAID-1 volumes and RAID-5 volumes.



Note – In earlier versions of the Solaris OS, the Solaris Volume Manager software was known as Solstice DiskSuite™ software, and logical volumes were known as metadevices. Most of the associated commandline tools begin with the prefix *meta*. Logical devices are located under the /dev/md directory.

Soft Partitions

As disks become larger, and disk arrays present larger logical devices to the Solaris OS, users must be able to subdivide disks or logical volumes into more than eight sections, often to create manageable file systems or partition sizes.

Soft partitions provide a mechanism for dividing large storage spaces into smaller, more manageable, sizes. For example, large storage aggregations provide redundant storage of many gigabytes, but many scenarios would not require as much space. Soft partitions allow you to subdivide that storage space into more manageable sections, each of which can have a complete file system.

For example, you could create 1000 soft partitions on top of a RAID-1 volume or RAID-5 volume so that each of your users can have a home directory on a separate file system. If a user needs more space at a later date, you can grow the soft partition.



Note – The Solaris Volume Manager software can support up to 8192 logical volumes per disk set, but is configured for 128 (d0–d127) by default. For instructions on increasing the number of logical volumes, refer to the *Solaris Volume Manager Administration Guide*, part number 806-6111-10.

Use soft partitioning to divide a slice or volume into as many divisions as needed. Assign a name for each division or soft partition, just like you would do for other storage volumes, such as stripes or mirrors. A soft partition, once named, can be directly accessed by applications, including file systems, as long as it is not included in another volume.

When you partition a disk and build a file system on the resulting slices, you cannot later extend a slice without modifying or destroying the disk format. With soft partitions, you can extend portions up to the amount of space on the underlying device without moving or destroying data on other soft partitions.

Suggestions for Soft Partitioning

Consider the following factors when implementing soft partitions in your storage environment:

- You can build soft partitions on any slice. Creating a single slice that occupies the entire disk and then creating soft partitions on that slice is the most efficient way to use soft partitions at the disk level.
- To expand and manage storage space, build stripes on top of your disk slices, and then build soft partitions on the stripes.
- You can grow soft partitions to use any available space on a volume.
- Create a RAID-1 volume or a RAID-5 volume, and then create soft partitions on the RAID 1 volume or RAID-5 volume for maximum flexibility and higher availability.

Introducing the State Database

Before creating volumes using the Solaris Volume Manager software, state database replicas must exist on the Solaris Volume Manager software system. The state database stores information on disk about the state of your Solaris Volume Manager software configuration. The state database records and tracks changes made to your configuration. The Solaris Volume Manager software automatically updates the state database when a configuration or state change occurs. For example, creating a new volume is a configuration change, while a submirror failure is a state change. This section addresses the following:

- The Solaris Volume Manager software state database
- Recommendations for state database replicas
- Suggestions for state database replicas

The Solaris Volume Manager Software State Database

The state database is a collection of *multiple, replicated* database copies. Each copy, called a state database replica, ensures that the data in the database is always valid. Having copies of the state database protects against data loss from single points-of-failure. The state database tracks the location and status of all known state database replicas. During a state database update, each state database replica is updated. The updates take place one at a time to protect against corrupting all updates if the system crashes.

The Solaris Volume Manager software state database contains configuration and status information for all volumes and hot spares. The Solaris Volume Manager software maintains replicas (copies) of the state database to provide redundancy and to prevent database corruption during a system crash.

If your system loses a state database replica, Solaris Volume Manager software must determine which state database replicas still contain non-corrupted data. The Solaris Volume Manager software determines this information by a majority consensus algorithm. This algorithm requires that a majority ($\text{half} + 1$) of the state database replicas be available and in agreement with each other before any of them are considered non-corrupt. Because of the majority consensus algorithm, you should create at least three state database replicas when you set up your disk configuration. A consensus can be reached as long as at least two of the three state database replicas are available.

During booting, the Solaris Volume Manager software ignores corrupted state database replicas. In some cases, Solaris Volume Manager software tries to rewrite state database replicas that are corrupted. Otherwise the databases are ignored until you repair them. If a state database replica becomes corrupt because its underlying slice encountered an error, you must repair or replace the slice, and then recreate the replica.

If all state database replicas are lost, you could lose all data that is stored on your Solaris Volume Manager software volumes. You should create enough state database replicas on separate drives and across controllers to prevent complete data loss. You should also save your initial configuration information, as well as your disk partition information.

To protect data, the Solaris Volume Manager software will not function unless half of all state database replicas are available. The main functions of the majority consensus algorithm are:

- The system will stay running if at least half of the state database replicas are available.
- The system will panic if fewer than half the state database replicas are available.
- The system will not start the Solaris Volume Manager software unless a majority ($\text{half} + 1$) of the total number of state database replicas are available.

Recommendations for State Database Replicas

To avoid single points-of-failure, you should distribute state database replicas across slices, drives, and controllers. A majority of replicas must survive a single component failure. The Solaris Volume Manager software requires that half the replicas be available to run, and that a majority ($\text{half} + 1$) be available to boot. If you lose a replica (for example, due to a device failure), you might run into problems when running Solaris Volume Manager software or when rebooting the system. When working with state database replicas, consider the following:

- You should create state database replicas on a dedicated slice of at least 4 Mbytes per replica.
- You can put replicas on unused slices, and then use them on RAID-0, RAID-1, or RAID-5 volumes.
- You cannot create state database replicas on any slices in use.
- A minimum of three state database replicas are recommended. The following guidelines are recommended:
 - For a system with only a single drive: put all three replicas in one slice.
 - For a system with two to four drives: put two replicas on each drive.
 - For a system with five or more drives: put one replica on each drive.
- Make sure that you have at least two extra replicas per mirror.
- You can add additional state database replicas to the system at any time. The additional state database replicas help to ensure the Solaris Volume Manager software's availability.



Caution – If you upgraded from Solstice DiskSuite software to Solaris Volume Manager software and have state database replicas at the beginning of slices (as opposed to on separate slices), do not delete existing replicas and replace them with new ones in the same location. The default Solaris Volume Manager software state database replica size is 8192 blocks, while the default size in Solstice DiskSuite software was 1034 blocks. If you delete a default-size state database replica from Solstice DiskSuite software, and add a new default-size replica with the Solaris Volume Manager software, you will overwrite the first 7158 blocks of any file system occupying the rest of the shared slice, which destroys the data.

Introducing Hot Spares and Hot Spare Pools

Hot spares and hot spare pools provide additional physical slices for automatic recovery from RAID-1 mirror or RAID-5 volume failures.

Hot Spares

A hot spare is a slice (not a volume) that is functional and available, but not in use. A hot spare is on reserve to substitute for a failed slice in a submirror or RAID-5 volume. You cannot use a hot spare to hold data or state database replicas until the hot spare is assigned as a member. A hot spare must be ready for immediate use in the event of a slice failure in the volume with which it is associated. To use hot spares, invest in additional disks beyond those that the system requires to function.

Hot Spare Pools

A hot spare pool is a collection of slices. The Solaris Volume Manager software uses hot spare pools to provide increased data availability for RAID-1 volumes and RAID-5 volumes. The Solaris Volume Manager software reserves a hot spare for automatic substitution when a slice failure occurs in either a submirror or a RAID-5 volume.



Note – Hot spares do not apply to RAID-0 volumes or to one-way mirrors. For automatic substitution to work, redundant data must be available.

Solaris Volume Manager Concepts

The Solaris Volume Manager software in the Solaris 9 and 10 Operating System replaces the Solstice DiskSuite software used in releases of the Solaris OS prior to Solaris 9 OS.

The Solaris Volume Manager software is used to implement RAID 0, RAID 1, RAID 1+0, and RAID 5.

This module covers the configuration of the following:

- RAID 0: Non-redundant disk array (concatenation and striping)
- RAID 1: Mirrored disk array

The State Database Replicas

The state database stores information on disk about the state of your Solaris Volume Manager software configuration. Multiple copies of the database, called replicas, provide redundancy and protect against data loss if a copy of the database is corrupted due to the system crashing or other failure. The state database replicas should be distributed across multiple disks so that failure of a single disk only causes the loss of a single state database replica.

If the system loses a state database replica, Solaris Volume Manager software uses a majority consensus algorithm to determine which state database replicas still contain valid data. The algorithm requires that a majority (half +1) of the state database replicas are available before any of them are considered valid. The majority consensus algorithm requires that you create at least three state database replicas before you build or commit any metadevices. To reach a consensus, at least two of the three replicas must be available.

The majority consensus algorithm:

- Makes sure that the system stays running if at least half of the state database replicas are available.
- Causes the system to panic if fewer than half of the state database replicas are available.
- Prevents the system from starting the Solaris Volume Manager software unless a majority of the total number of state database replicas are available.

If insufficient state database replicas are available, you must boot into single-user mode and delete enough of the corrupt replicas to achieve a majority consensus.

State database replicas are stored in their own disk slices.

Creating the State Database

You can create state database replicas by using:

- The `metadb -a` command
- The Solaris Volume Manager software GUI

Creating the State Database Using the Command Line

To create state database replicas using the command line, use the `metadb` command. The syntax of the command is:

```
metadb -a [-f] [-c n] [-l nnnn] disk_slice
```

where:

- a Adds a state database replica.
- f Forces the operation, even if no replicas exist. Use this flag to force the creation of the initial replicas.
- c n Specifies the number of replicas to add to the slice.
- l nnnn Specifies the size of the new replicas, in blocks.
- disk_slice* Specifies the name of the *disk_slice* that will hold the replica.

Note – The `metadb` command without options reports the status of all replicas.

The following example shows the creation of state database replicas:

```
# metadb -a -f c0t0d0s4 c0t0d0s5 c1t0d0s0 c1t0d0s1
# metadb
      flags   first blk    block count
      a       u      16        8192          /dev/dsk/c0t0d0s4
      a       u      16        8192          /dev/dsk/c0t0d0s5
      a       u      16        8192          /dev/dsk/c1t0d0s0
      a       u      16        8192          /dev/dsk/c1t0d0s1
```

This example lists the four replicas that were just created. Each replica begins at block 16 of the assigned disk slice. Each replica is 8192 blocks, or 4 Mbytes in size. The flags indicate that the replica is active and up to date. If there are capital letters in the flags field, it is an indication that the replica is corrupt.

Note – The previous example places the state database replicas on disks on different controllers. This is an appropriate configuration for a production environment.

Configuring RAID-0

RAID-0 volumes allow you to expand disk storage capacity efficiently. These volumes do not provide data redundancy but can be used to expand disk storage capacity. If a single slice fails on a RAID-0 volume, there is a loss of data. RAID-0 comes in two forms, stripes and concatenations.

- Concatenated volumes (or concatenations)
A concatenated volume writes data to the first available slice. When the first slice is full, the volume writes data to the next available slice.
- Striped volumes (or stripes)
A stripe distributes data equally across all slices in the stripe.

RAID-0 Striped Volumes

Figure 5-1 shows the arrangement of a RAID-0 volume configured as a stripe. A RAID-0 volume configured as a stripe arranges data across two or more slices. Striping alternates equally-sized segments of data across two or more slices, forming one logical storage unit. These segments are interleaved round-robin, so that the combined space is created alternately from each slice.

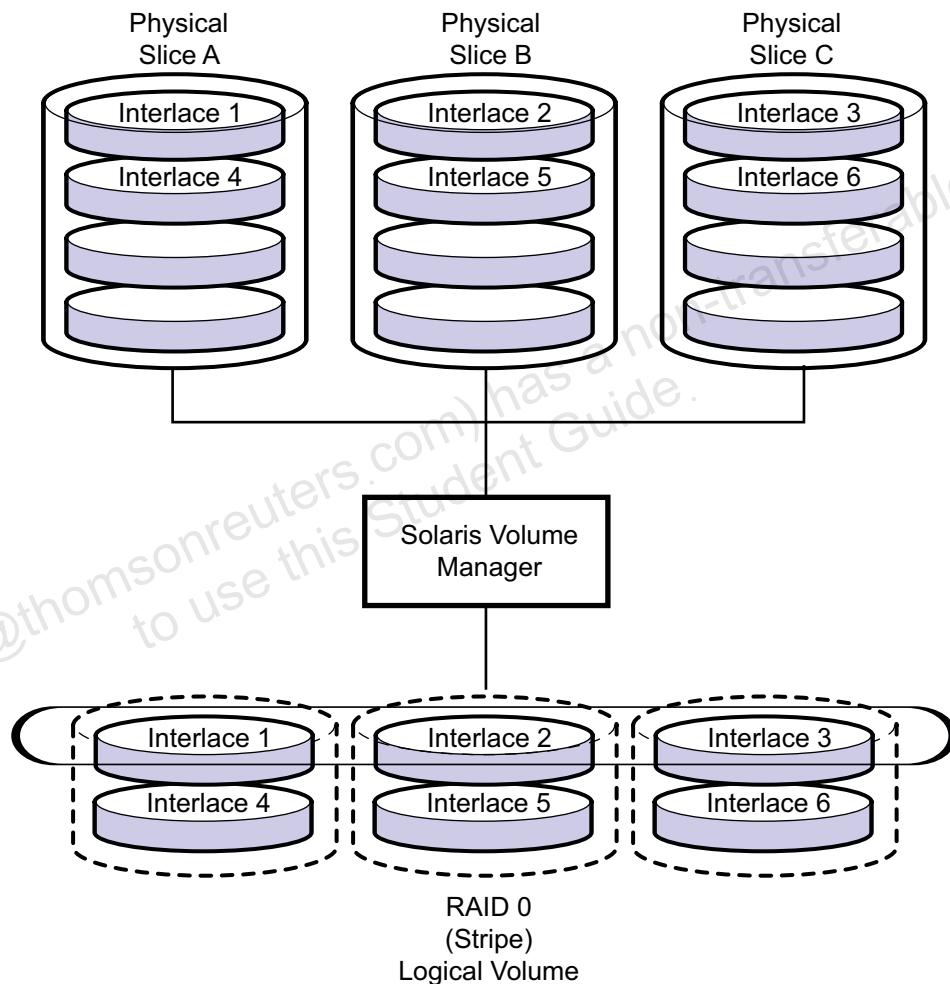


Figure 5-1 RAID-0 Stripe

Striping enables parallel data access because multiple controllers can access the data at the same time. Parallel access increases Input/Output (I/O) performance because multiple disks in the volume can service I/O requests simultaneously.

You cannot convert an existing file system directly to a striped volume. You must first back up the file system, create the striped volume, and then restore the file system to the striped volume.

Creating a RAID-0 Volume

The following examples will show how to create a RAID-0 volume using the command line and Sun Management Console.

Using the Command Line

In this example, the slice being used for the /export/home file system is almost at capacity. A new slice from another disk is concatenated to it, making a RAID-0 concatenated volume. The existing slice is shown:

```
# df -h /export/home
Filesystem          size   used  avail capacity  Mounted on
/dev/dsk/c0t0d0s7    470M   395M    28M     94%      /export/home
```

If the metadatabases are not already configured, they need to be configured before creating any metadevices.

```
# metadb -a -f -c 2 c3t2d0s7 c3t3d0s7
# metadb
      flags        first blk      block count
      a            16           8192      /dev/dsk/c3t2d0s7
      a            8208         8192      /dev/dsk/c3t2d0s7
      a            16           8192      /dev/dsk/c3t3d0s7
      a            8208         8192      /dev/dsk/c3t3d0s7
```

RAID-0 Striped Volumes

The concatenated volume must be referenced by a metadevice name. The metainit command creates the metadevices. The syntax of the metainit command is:

```
metainit -f concat/stripe numstripes width component...
```

where:

| | |
|----------------------|--|
| <i>-f</i> | Forces the metainit command to continue, even if one of the slices contains a mounted file system or is being used as swap space. This option is useful when configuring mirrors or concatenations on root (/), swap, and /usr file systems. |
| <i>concat/stripe</i> | Specifies the volume name of the concatenation or stripe being defined. |
| <i>numstripes</i> | Specifies the number of individual stripes in the metadevice. For a simple stripe, <i>numstripes</i> is always 1. For a concatenation, <i>numstripes</i> is equal to the number of slices. |
| <i>width</i> | Specifies the number of slices that make up a stripe. When the <i>width</i> is greater than 1, the slices are striped. |
| <i>component</i> | Specifies the logical name for the physical slice (partition) on a disk drive, such as /dev/dsk/c0t0d0s1. |

Metadevices are referenced by the letter d followed by a number. The new metadevice will be called d0. The -f option is required, as one of the slices being included in the concatenated volume is mounted. As this is a concatenation, the number of stripes is equal to the number of slices being added, in this case 2. The number of slices in each stripe is one, so the number 1 appears before each slice:

```
# metainit -f d0 2 1 c0t0d0s7 1 c3t2d0s0
d0: Concat/Stripe is setup
```

The metastat command is used to check the configuration:

```
# metastat
d0: Concat/Stripe
  Size: 3118752 blocks (1.5 GB)
  Stripe 0:
    Device      Start Block  Dbase   Reloc
    c0t0d0s7        0       No     Yes
  Stripe 1:
    Device      Start Block  Dbase   Reloc
    c3t2d0s0      2160      No     Yes

Device Relocation Information:
Device  Reloc  Device ID
c0t0d0  Yes    id1,dad@AST38420A=7AZ0VMFG
c3t2d0  Yes    id1,sd@SFUJITSU_MAB3045S_SUN4.2G00F50615_____
```

The d0 metadevice is shown, with the two stripes which make up the concatenation. The new device is represented with block and character special device files:

```
# ls -lL /dev/md/dsk
total 0
brw-r----- 1 root      sys          85,  0 Oct 25 12:35 d0
# ls -lL /dev/md/rdsk
total 0
crw-r----- 1 root      sys          85,  0 Oct 25 12:35 d0
```

RAID-0 Striped Volumes

The new metadevice (d0) has been created but is not being used yet. The /export/home file system is still mounted as a regular disk slice:

```
# df -h /export/home
Filesystem           size   used  avail capacity  Mounted on
/dev/dsk/c0t0d0s7    470M   395M   28M    94%      /export/home
```

It needs to be remounted using the new metadevice device files. Locate the entry in /etc/vfstab file which mounts the file system at boot time:

```
/dev/dsk/c0t0d0s7 /dev/rdsk/c0t0d0s7 /export/home ufs 2 yes -
```

Change the device files to the metadevice files:

```
/dev/md/dsk/d0 /dev/md/rdsk/d0 /export/home ufs 2 yes -
```

Then un-mount and re-mount the file system using the new device files:

```
# umount /export/home
# mount /export/home
# df -h /export/home
Filesystem           size   used  avail capacity  Mounted on
/dev/md/dsk/d0       470M   395M   28M    94%      /export/home
```

The file system is now mounted using the metadevice device file. Notice that the file system does not appear to be any bigger, and the capacity is still at 94 percent. The existing file system needs to be grown into the new space. This is done with the growfs command. Use the option -M to specify a mount point:

```
# growfs -M /export/home /dev/md/rdsk/d0
/dev/md/rdsk/d0: 3118752 sectors in 3094 cylinders of 16 tracks, 63
sectors
      1522.8MB in 194 cyl groups (16 c/g, 7.88MB/g, 3776 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
  32, 16224, 32416, 48608, 64800, 80992, 97184, 113376, 129568, 145760,
  2968096, 2984288, 3000480, 3016672, 3032864, 3049056, 3065248, 3081440,
  3096608, 3112800,
```

The file system now occupies all the space in the d0 metadevice:

```
# df -h /export/home
Filesystem           size   used  avail capacity  Mounted on
/dev/md/dsk/d0       1.4G   395M   988M    29%      /export/home
```

Configuring RAID-1

RAID-1 volumes are also known as mirrors and provide data redundancy. In a two-way mirror, the data is written to two disk slices of the same size. If one disk fails, the other will have an up-to-date copy of the data.

A RAID-1 volume maintains identical copies of the data in several RAID-0 volumes. Mirroring requires more disks. You need at least twice as much disk space as the amount of data to be mirrored.

After configuring a mirror, you can use it as if it were a physical slice. With multiple copies of data available, and correctly configured read and write policies, data access time is reduced.

You can mirror any file system, including existing file systems.

Using Multiple Submirrors

A mirror is made of two or more RAID-0 volumes. The mirrored RAID-0 volumes are called submirrors. A mirror consisting of two submirrors is known as a two-way mirror, while a mirror consisting of three submirrors is known as a three-way mirror.

Creating a two-way mirror is usually sufficient for data redundancy. A third submirror lets you maintain redundancy with one of the other two submirrors offline.

When a submirror is offline, it is in a read-only mode. The Solaris Volume Manager software tracks all the changes written to the online submirror. When the submirror is brought back online, only the newly written portions are resynchronized. Typical reasons for taking the submirror offline include backups, troubleshooting and repair.

You can attach or detach a submirror from a mirror at any time, though at least one submirror must remain attached to the mirror at all times. Usually, you begin the creation of a mirror with only a single submirror, after which you can attach additional submirrors.



Mirror Options

Mirror performance can be modified by using the following options:

- Mirror read policy
- Mirror write policy

Note – The mirror options listed here are representative of the options presented when configuring RAID-1 mirrors using the Solaris Volume Manager software.

You can define mirror options when you initially create the mirror or after you set up the mirror. You can distribute the load across the submirrors to improve read performance. Table 5-1 describes the configurable mirror read policies.

Table 5-1 Mirror Read Policies

| Read Policy | Description |
|-----------------------|--|
| Round Robin (default) | Balances the load across the submirrors |
| Geometric | Enables the system to divide reads among submirrors on the basis of a logical disk block address |
| First | Directs all reads to the first submirror |

You can improve write performance by replicating all submirrors simultaneously. If a failure occurs during this write, the submirror that had the failure is put into maintenance state (errored state). Table 5-2 describes the configurable mirror write policies.

Table 5-2 Mirror Write Policies

| Write Policy | Description |
|--------------------|---|
| Parallel (Default) | Replicates a write to a mirror, and dispatches the write to all of the submirrors simultaneously |
| Serial | Specifies that writes to one submirror must complete before initiating writes to the next submirror |

When a submirror is offline, any writes to the mirror are tracked in a dirty region log. When the submirror is brought back online, those regions must be updated or resynchronized.

Building a Mirror of the Root (/) File System

The procedure for building a mirror of the root (/) file system can be accomplished using the command line exclusively but it is not possible to use the Solaris Management Console (SMC) exclusively. As seen during RAID-0 configuration, SMC is not able to force the creation of a metadevice from a mounted file system.



Note – Remove the volume d0 created in the previous example to avoid confusion during this procedure.

This section describes how to create a RAID-1 volume for the root (/) file system, which cannot be unmounted. To create a mirror, do the following:

1. Create a RAID-0 volume for the file system you want to mirror.
2. Create a second RAID-0 volume to contain the second submirror of the RAID-1 volume.
3. Create a one-way mirror using the RAID-0 volume that contains the file system to be mirrored.
4. Use the metaroot command to update the system's configuration, as this is a root (/) mirror.
5. Reboot your system, as this is a root (/) mirror.
6. Attach the second submirror to the file system mirror.
7. Record the alternate boot path that is used in the event of a failure of the primary submirror, as this is a mirror of the root (/) file system.

The Scenario

The scenario assumes the root (/) file system is on disk slice c0t0d0s0.

1. A RAID-0 volume called d11 is created from slice c0t0d0s0.
2. A second RAID-0 volume is created as metadevice d12 from a spare disk slice at c3t3d0s1.
3. A RAID-1 volume is created and named d10 using the RAID-0 volumes named d11 and d12, as shown in Figure 5-2.

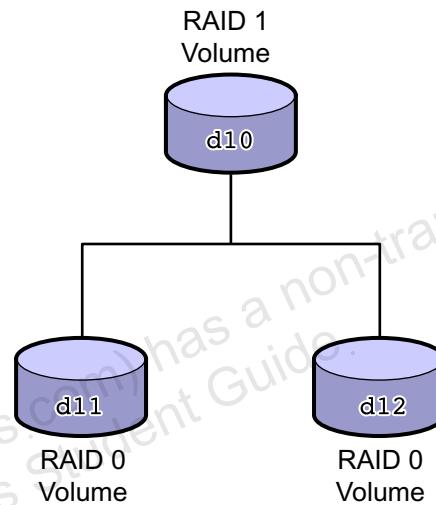


Figure 5-2 Mirror of Root (/) Partition

Creating the RAID-0 Volumes

The first step when building a mirror of the root (/) file system is to create RAID-0 volumes, which you later combine to form the mirror. Each RAID-0 volume becomes a submirror to the mirror. Use the `metainit` command to force the creation of the RAID-0 volume. The force (-f) option must be used because this is the root (/) file system, which cannot be unmounted.

The following example shows how to use the `metainit` command to create a RAID-0 volume:

```
# /usr/sbin/metainit -f d11 1 1 c0t0d0s0
d11: Concat/Stripe is setup
```



Caution – If converting an existing file system to a RAID-0 volume, both the `numstripes` and `width` arguments must be 1, or the data is lost.

Creating the RAID-1 Volume

You can create the RAID-1 volume using:

- The `metainit` command
- The Enhanced Storage Tool within the Solaris Management Console

The `metainit` Command

The syntax for creating a RAID-1 volume by using the `metainit` command is:

```
metainit mirror -m submirror [read_options] [write_options] [pass_num]
```

where:

| | |
|----------------------|---|
| <i>mirror -m</i> | Specifies the volume name of the mirror. |
| <i>submirror</i> | The <code>-m</code> indicates that the configuration is a mirror. Submirror is a volume (stripe or concatenation) that makes up the initial one-way mirror. |
| <i>read_options</i> | The following read options for mirrors are available: <ul style="list-style-type: none">• <code>-g</code> – Enables the geometric read option, which results in faster performance on sequential reads.• <code>-r</code> – Directs all reads to the first submirror. Use the <code>-r</code> option only when the devices that comprise the first submirror are substantially faster than those of the second mirror. You cannot use the <code>-r</code> option with the <code>-g</code> option. |
| <i>write_options</i> | The following write option is available: <code>S</code> – Performs serial writes to mirrors. The default setting for this option is parallel write. |
| <i>pass_num</i> | A number (0–9) at the end of an entry defining a mirror that determines the order in which that mirror is resynchronized during a reboot. The default is 1. Smaller pass numbers are resynchronized first. Equal pass numbers are run concurrently. If 0 is used, the resynchronization is skipped. Use 0 only for mirrors mounted as read-only, or as swap space. |



Note – If neither the **-g** nor **-r** options are specified, reads are made in a round-robin order from all submirrors in the mirror. This process enables load balancing across the submirrors.

The following command-line example creates a mirrored volume named d10, and attaches a one-way mirror using volume d11. Volume d11 is a submirror of the mirror named d10.

```
# /usr/sbin/metainit d10 -m d11  
d10: Mirror is setup
```

Executing the metaroot Command

When creating mirrors of mounted file systems, you must update the /etc/vfstab file to change the mount point from a slice, such as /dev/dsk/c#t#d#s#, to a volume, such as /dev/md/dsk/d#. When mirroring any mounted file system other than root (/), you can use the vi editor to update the /etc/vfstab file.

When mirroring the root (/) file system, use the metaroot command to modify the /etc/vfstab and /etc/system files, as follows:

```
metaroot device
```

where *device* specifies either the metadevice or the conventional disk device (slice) used for the root (/) file system.

The following example shows that the /etc/vfstab file has been updated by the metaroot command to point to the RAID-1 mirrored metadevice.

```
# metaroot d10
# grep md /etc/vfstab
/dev/md/dsk/d10    /dev/md/rdsk/d10    /      ufs      1      no      -
```

In addition to modifying the /etc/vfstab file to update the root (/) file system pointer, the metaroot command updates the /etc/system file to support the logical volumes. For example:

```
# tail /etc/system
rootdev:/pseudo/	md@0:0,10,blk
```

You must reboot the system before attaching the secondary submirror. When the system boots, it mounts the root file system using the metadevice device file. Enter the init command to reboot the system:

```
# init 6
```

After the reboot is complete, the root file system is mounted through the d10 metadevice:

```
# df -h /
Filesystem          size   used  avail capacity  Mounted on
/dev/md/dsk/d10     141M   111M   15M    88%      /
```

The metastat command shows the state of the metadevices. Notice here that only one submirror is in the d10 metadevice:

```
# metastat
d10: Mirror
    Submirror 0: d11
        State: Okay
        Pass: 1
        Read option: roundrobin (default)
        Write option: parallel (default)
        Size: 307440 blocks (150 MB)

d11: Submirror of d10
    State: Okay
    Size: 307440 blocks (150 MB)
    Stripe 0:
        Device      Start Block  Dbase      State Reloc Hot Spare
        c0t0d0s0          0       No        Okay   Yes
(output omitted)
```

Attach the secondary submirror by using the metattach command:

```
# metattach d10 d12
d10: submirror d12 is attached
```



Caution – Create a one-way mirror with the metainit command, and then attach the additional submirrors with the metattach command. If the metattach command is not used, no resynchronization operations occur. As a result, data could become corrupted as the Solaris Volume Manager software assumes that both sides of the mirror are identical and can be used interchangeably.

The metastat command shows the mirror synchronization taking place.

```
# metastat d10
d10: Mirror
    Submirror 0: d11
        State: Okay
    Submirror 1: d12
        State: Resyncing
    Resync in progress: 83 % done
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 307440 blocks (150 MB)
```

Building a Mirror of the Root (/) File System

```
d11: Submirror of d10
  State: Okay
  Size: 307440 blocks (150 MB)
  Stripe 0:
    Device      Start Block  Dbase      State Reloc Hot Spare
    c0t0d0s0          0       No        Okay   Yes

d12: Submirror of d10
  State: Resyncing
  Size: 2097360 blocks (1.0 GB)
  Stripe 0:
    Device      Start Block  Dbase      State Reloc Hot Spare
    c3t3d0s1          0       No        Okay   Yes
```

Note – All of the usual considerations of mddb quorum apply to x86-based systems, just as they do for SPARC-based systems.



Updating the boot-device PROM Variable

If you mirror your root (/) file system, record the alternate boot path contained in the boot-device PROM variable. In the following example, you determine the path to the alternate boot device by using the `ls -l` command on the slice that is being attached as the secondary submirror to the root (/) mirror.

```
# ls -l /dev/dsk/c3t3d0s1
lrwxrwxrwx  1 root      root          57 Oct 25 11:22 /dev/dsk/c3t3d0s1 -
> ../../devices/pci@1f,0/pci@1/pci@1/SUNW,isptwo@4/sd@3,0:b
```

Record the path that follows the `/devices` directory:

`/pci@1f,0/pci@1/pci@1/SUNW,isptwo@4/sd@3,0:b`



Caution – When using some disk controllers, the path to the device varies between the entries in the `/devices` directory and the entries in the OpenBoot™ programmable read-only memory (PROM). In these instances, follow the entries in the OpenBoot PROM.

If, for example, on one Ultra™ 5 workstation, the PCI-SCSI controller returns:

`/pci@1f,0/pci@1/scsi@4,1/sd@2,0:b`

from the /devices directory, yet the show-disks command from the OpenBoot PROM returned:

```
/pci@1f,0/pci@1/scsi@4,1/disk
```

then, the alternate boot path must be:

```
/pci@1f,0/pci@1/scsi@4,1/disk@2,0:b
```

If you do not adapt to the change when attempting to boot from the alternate boot device, you get an error stating:

```
can't open boot device
```

To get the system to boot automatically from the alternate boot device in the event of a primary root submirror failure, complete the following steps:

1. Use the OpenBoot nvalias command to define a `backup_root` device alias for the secondary root mirror. For example:

```
ok nvalias backup_root /pci@1f,0/pci@1/pci@1/SUNW,isptwo@4/sd@3,0:b
```

2. Redefine the boot-device variable to reference both the primary and secondary submirrors, in the order in which you want to access them. For example:

```
ok printenv boot-device
```

```
boot-device= disk net
```

```
ok setenv boot-device disk backup_root net
```

```
boot-device= disk backup_root net
```

In the event of primary root disk failure, the system automatically boots from the secondary submirror. To test the secondary submirror, boot the system manually, as follows:

```
ok boot backup_root
```

Configuring an x86-Based System for Mirrored Failover

Root mirroring on x86-based systems is more complex than root mirroring on SPARC-based systems. Specifically, there are issues with being able to boot from the secondary side of the mirror when the primary side fails. On x86-based systems, the system Basic Input/Output System (BIOS), fdisk partitioning, and the GNU GRand Unified Bootloader (GRUB) are the complicating factors.

BIOS

The BIOS found on x86-based systems is analogous to the PROM interpreter on SPARC. The BIOS is responsible for finding the right device to boot from, then loading and executing the master boot record from that device. BIOS is configurable to some degree. In general you can usually select the order of devices that you want the BIOS to probe devices (for example, floppy, IDE disk, SCSI disk, network) but you may be limited in configuring at a more granular level.

Before the kernel is started, the system is controlled by BIOS. You can usually configure the BIOS to select the order of devices to probe for the boot record. Additionally, most modern BIOS implementations allow you to configure your devices so that the failover to the secondary submirror is automatic. If your system's BIOS does not have this feature and the primary submirror fails, you need to access the BIOS during system boot to reconfigure the system to boot from the secondary root slice.

fdisk Partitioning

Sun's x86-based systems use fdisk partitions on system disks. For an x86-based system with the Solaris OS installed, there are two common fdisk partitioning schemes. One approach uses two fdisk partitions.

There is a Solaris OS fdisk partition and another, small fdisk partition of about 10 Mbyte called the x86 boot partition. This partition has an ID value of 190. The Solaris OS system installation software creates a configuration with these two fdisk partitions as the default. The x86 boot partition is needed in some cases, such as when you want to use live-upgrade on a single disk configuration, but it is problematic when using root mirroring. The Solaris OS system installation software only allows one x86 boot partition for the entire system and it places important data on that fdisk partition.

You can determine if your system has a separate x86 boot partition, check the /etc/vfstab file. The x86 boot partition exists if the file contains an entry similar to the following:

```
/dev/dsk/c0t0d0p0:boot - /boot pcfs - no -
```

Because this fdisk partition is outside of the Solaris OS fdisk partition, it cannot be mirrored by SVM. Furthermore, because there is only a single copy of this fdisk partition, it represents a single point of failure.

To use the SVM to mirror the root file system, the file system must use the single Solaris fdisk partition. Therefore, if the x86 boot partition already exists in the system, delete this partition with the `fdisk` command and then reinstall the Solaris software. When you reinstall, the boot partition is no longer recreated.

The GRand Unified Bootloader (GRUB)

Starting with the Solaris 10 1/06 release, the open source GNU GRand Unified Bootloader (GRUB) has been adopted in the Solaris OS for x86 based systems. GRUB is responsible for loading a boot archive into the system's memory. A boot archive is a collection of critical files that is needed during system start-up before the root (/) file system is mounted. The boot archive is used to boot the Solaris OS.

The most notable change is the replacement of the Solaris Device Configuration Assistant with the GRUB menu. The GRUB menu facilitates booting the different operating systems that are installed on your system. The GRUB menu is displayed when you boot an x86 based system. From the GRUB menu, you can select an OS instance to install by using the arrow keys. If you do not make a selection, the default OS instance is booted.

The device naming conventions that GRUB uses are slightly different from previous Solaris OS versions. Understanding the GRUB device naming conventions can assist you in correctly specifying drive and partition information when you configure GRUB on your system. Table describes the GRUB device naming conventions.

Table 5-3 Naming Conventions for GRUB Devices

| Device Name | Description |
|----------------------|---|
| (fd0), (fd1) | First diskette, second diskette |
| (nd) | Network device |
| (hd0,0), (hd0,1) | First and second fdisk partition of first BIOS disk |
| (hd1,0), (hd1,1) | First and second fdisk partition of second BIOS disk |
| (hd0,0,a), (hd0,0,b) | Solaris/BSD slice 0 and 1 on first fdisk partition on the first BIOS disk |

The functional GRUB components include the stage1 and stage2 programs, and the menu.1st file. The stage1 program is installed by default on the first sector of the Solaris fdisk partition. The stage2 program is installed beginning at sector 50 of the first cylinder of the Solaris fdisk partition and extends for 233 sectors.

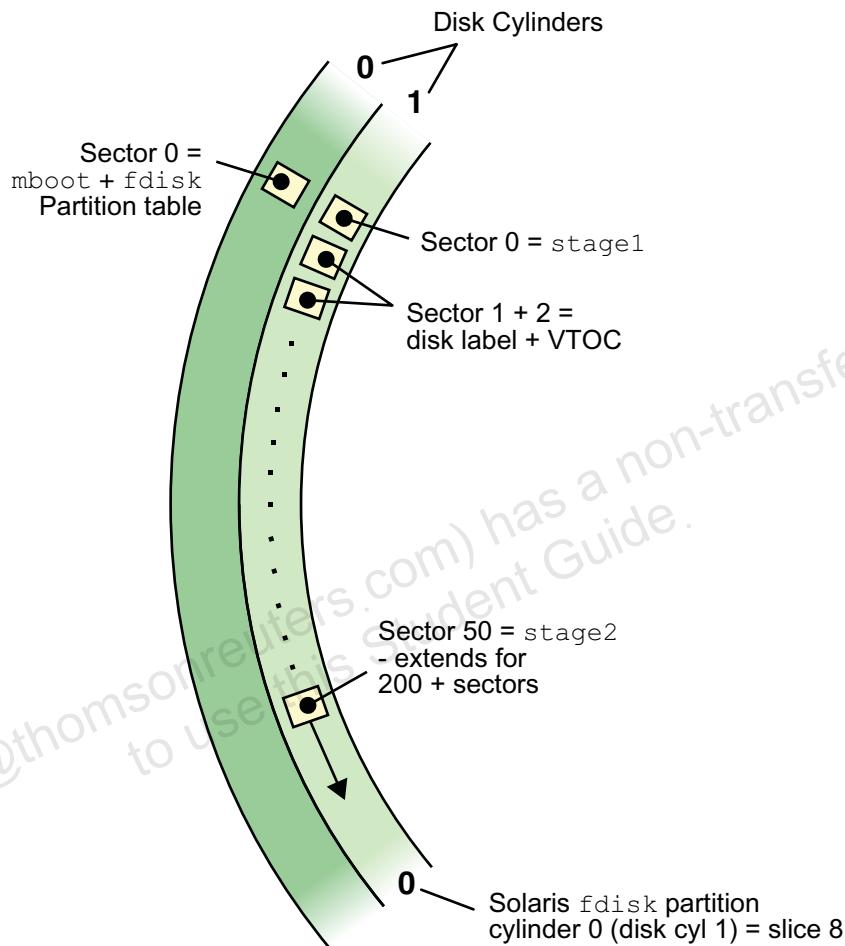


Figure 5-3 x86/x64 Boot Program Locations

The stage1 and stage2 programs are located in cylinder 0 (typically disk cylinder 1) of the Solaris fdisk partition, and remain protected by the space allocated to slice 8 within the Solaris fdisk partition. To make a disk bootable, you use the `installgrub` command to install the stage1 and stage2 programs in these locations. For example:

```
# /sbin/installgrub /boot/grub/stage1 /boot/grub/stage2 /dev/rdsck/c2d0p0
```

The disk label located at sector 0 of disk cylinder 0 contains the mboot program by default. The fdisk utility automatically installs the mboot program in this location when you establish an fdisk partition table. The mboot program locates the active fdisk partition at the beginning of the boot process.

To make a disk bootable, you may need to install the mboot program manually. To do this, you use the fdisk command as in this example, and specify the appropriate disk device:

```
# fdisk -b /usr/lib/fs/ufs/mboot -n /dev/rdsk/c2d0p0
```

The menu.1st is located in the /boot/grub directory, and read by the GRUB stage2 program. Figure 5-3 illustrates the layout of the mboot, stage1 and stage2 programs on an x86/x64 system's disk.

The following description may be useful in understanding how these objects on disk participate in the boot process on an x86/x64 system.

1. The BIOS loads the MBR.

The program loaded is normally mboot, but it can be stage1 GRUB instead. The mboot program merely finds the active partition, then loads and executes the first sector from the stage1 GRUB. The stage1 GRUB has the physical location of stage2 GRUB hard-coded within itself at offset 0x44 from the start of the partition boot sector. This address and the stage1 program itself are put there by the installgrub program.

2. Knowing the block address at which the stage2 GRUB starts on the disk, stage1 then loads and executes stage2. The stage2 GRUB contains code that allows it to navigate the UFS structure on the root file system.
3. The stage2 GRUB locates the GRUB configuration file /boot/grub/menu.1st.
4. From here, a menu is presented to the user based on the contents of /boot/grub/menu.1st.

Typically, for the Solaris OS, the /boot/grub/menu.1st file contains a kernel line and a module line. The module will point to the boot archive. The boot archive is a basic root file system that contains the kernel, the kernel modules, and a number of configuration files needed for bringing up the kernel such as /etc/system.

5. The boot archive is loaded entirely in memory as a ramdisk.
6. The kernel is loaded from the boot archive.

- Unauthorized reproduction or distribution prohibited. Copyright© 2010, Oracle and/or its affiliates.
7. The kernel then executes, finds out what the real root file system is by looking at the bootpath property in the /boot/solaris/bootenv.rc file in the boot archive, mounts the device specified as root, and continues with the boot process.
 8. The memory that was taken up by GRUB and the ramdisk are subsequently freed for future use.

Laying out all the components to make the boot process (Figure 5-3 on page 5-30) work properly is accomplished by running a program called installgrub. The installgrub command performs the following operations:

1. Copy the stage1 GRUB to the partition boot sector (the first sector of the Solaris fdisk partition). The installgrub program modifies this copy of stage1 to point to the physical location of stage2 GRUB on the disk.
2. Copy the stage2 GRUB starting at an offset of 50 sectors from the beginning of the Solaris fdisk partition.
3. If the -m option is specified, copy the stage1 GRUB to the MBR.

Creating a RAID-1 Volume From the root File System

This section describes the general steps required for creating a RAID-1 volume from the root file system by using GRUB.

1. Verify that the ordering for the BIOS boot device can be configured to allow the system to boot off of the second disk in the mirror.
2. Verify that the fdisk partitions are configured to support root mirroring.
3. Next, make the secondary submirror bootable with a master boot program. The -b option of fdisk(1M) does this.

For example:

```
# fdisk -b /usr/lib/fs/ufs/mboot -n /dev/rdsck/c2d0p0
```

4. Make the secondary disk bootable.

For example:

```
# /sbin/installgrub /boot/grub/stage1 /boot/grub/stage2 /dev/rdsck/c2d0p0
```

5. Identify the slice that contains the existing root (/) file system to be mirrored.
6. Create a new RAID-0 volume on the slice from the previous step.

This can be done using the Enhanced Storage tool within the Solaris Management Console or by using the metainit command. See “Creating the RAID-0 Volumes” on page 5-21.

7. Create a second RAID-0 volume (`c1t1d0s0` in this example) on an unused slice to act as the second submirror.
8. Create a one-way mirror.

This can be done using the Enhanced Storage tool within the Solaris Management Console or by using the metainit command. See “Creating the RAID-1 Volume” on page 5-22.

9. Remount your newly mirrored file system, then reboot the system.

```
# metaroot volume-name
# reboot
```

10. Attach the second submirror.

```
# metattach volume-name submirror-name
```

11. Define the alternative boot path in the `/boot/grub/menu.lst` file.

To enable the system to boot off of the disk that holds the secondary submirror, configure the system to see the disk as the alternate boot device.

In this `menu.lst` example, the alternative path is on the first slice of the first `fdisk` partition on the second BIOS disk.

```
# vi /boot/grub/menu.lst
...
title alternate boot
root (hd1,0,a)
kernel /boot/multiboot
module /boot/x86.miniroot-safe
```

Unmirroring the root (/) File System

Follow this procedure to unmirror the root (/) file system. This procedure assumes that the root (/) file system is mirrored on a Solaris Volume Manager software volume named d10, and that the mirror consists of two submirrors. The primary submirror is d11, and the secondary submirror is d12. To unmirror the root (/) file system, complete the following steps:

1. Run the metastat command on the mirror to verify that submirror 0 is in the Okay state.

```
# metastat d10
d10: Mirror
    Submirror 0: d11
        State: Okay
    Submirror 1: d12
        State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 307440 blocks (150 MB)

d11: Submirror of d10
    State: Okay
    Size: 307440 blocks (150 MB)
    Stripe 0:
        Device      Start Block  Dbase      State Reloc Hot Spare
        c0t0d0s0          0     No        Okay   Yes

d12: Submirror of d10
    State: Okay
    Size: 2097360 blocks (1.0 GB)
    Stripe 0:
        Device      Start Block  Dbase      State Reloc Hot Spare
        c3t3d0s1          0     No        Okay   Yes

Device Relocation Information:
Device  Reloc  Device ID
c0t0d0  Yes    id1,dad@AST38420A=7AZ0VMFG
c3t3d0  Yes    id1,sd@SFUJITSU_MAB3045S_SUN4.2G00F52267
```

2. Run the metadetach command on the mirror to make a one-way mirror.

```
# metadetach d10 d12  
d10: submirror d12 is detached
```

3. Because this is a root (/) file system mirror, run the metaroot command to update the /etc/vfstab and etc/system files.

```
# metaroot /dev/dsk/c0t0d0s0  
# grep c0t0d0s0 /etc/vfstab  
/dev/dsk/c0t0d0s0 /dev/rdsck/c0t0d0s0 / ufs 1 no -
```

4. Reboot the system.

```
# init 6
```

5. Run the metaclear command to clear the mirror and submirrors. The -r option recursively deletes specified metadevices and hot spare pools, associated with the targeted metadevices specified in the metaclear command.

```
# metaclear -r d10  
d10: Mirror is cleared  
d11: Concat/Stripe is cleared  
# metaclear d12  
d12: Concat/Stripe is cleared
```

6. If you changed your boot-device variable to an alternate boot path, return it to its original setting.

Exercise: Mirroring the root (/) File System on SPARC-Based Systems

In this exercise, you complete the following:

- Mirror the root (/) file system
- Update the default boot device
- Unmirror the root (/) file system

Preparation

Due to the differences in Sun's SPARC and x86 architecture, this exercise can only be performed on SPARC-based systems. If you are using an x86-based system, perform the tasks in "Exercise: Mirroring the root (/) File System on x86/x64-Based Systems" on page 5-49.

This exercise mirrors the root (/) file system of your system's boot disk.

This exercise requires a second disk that is not in use. Steps in this exercise direct you to partition the second disk so that it has two partitions equal to or greater than the size of the root (/) partition on the boot disk, and at least two partitions to be used for state database replicas.

This exercise is performed on each individual system, so there is no need to work with a partner. Steps in these procedures are executed using the command line.

This exercise requires an understanding of how to use the format utility to partition disks.

Tasks

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Use the `df -h` command to list file systems in use, and the `format` utility to display the partition table for your system's boot disk.
Record the following information:

Exercise: Mirroring the root (/) File System on SPARC-Based Systems

- Disk slice used for the root (/) file system, and its size in megabytes. This will become the primary submirror:

 - Does the slice used for the root (/) file system start on cylinder 0 of the boot disk?

 - Disk slice for state database replica 1:

 - Disk slice for state database replica 2:

3. Use the format utility to partition your *spare* disk so that it includes the partitions listed:
- Set the size of slice 0 to be equal to or greater than the disk slice used for the root (/) file system. This slice is a candidate to become the secondary submirror.
 - Set the size of slice 1 to be equal to or greater than the disk slice used for the root (/) file system. This slice is a candidate to become the secondary submirror.
 - Set the size of slice 6 to be at least 16 Mbytes. This slice will be used for state database replica 3.
 - Set the size of slice 7 to be at least 16 Mbytes. This slice will be used for state database replica 4.
4. Determine the names of Solaris Volume Manager objects to use for this exercise:
- Volume to map to the root (/) file system primary submirror:

 - Volume to map to the root (/) file system secondary submirror:

 - Volume to map to the root (/) file system mirror:

5. Create a sufficient number of state database replicas to support the majority consensus algorithm used in the Solaris Volume Manager software.

Exercise: Mirroring the root (/) File System on SPARC-Based Systems

What is the minimum number of state database replicas necessary to support the majority consensus algorithm?

6. Create a RAID-0 volume to use as the root (/) file system's primary submirror.
7. Create a RAID 0 volume on the secondary drive to use as the root (/) file system's secondary submirror.

You should refer to step 2 to determine which of the following conditions is true.

 - a. If the root slice on your boot disk *starts on* cylinder 0, use slice 0 on the second disk as the secondary submirror.
 - b. If the root slice on your boot disk *does not start on* cylinder 0, use slice 1 on the second disk as the secondary submirror.
8. Create a RAID-1 volume as a one-way mirror using the root (/) file system primary submirror as the source of the mirror's data.
9. Update the /etc/vfstab file to use the RAID-1 volume as the mount point for the root (/) file system and observe the changes to the /etc/vfstab and the /etc/system files.
10. Reboot the system, and then log in as root.
11. Attach the RAID-0 volume used as the root (/) file system's secondary submirror to the RAID-1 volume and allow the mirror synchronization to complete before continuing.

What is the primary reason for using the command line to attach a secondary submirror to a mirror?



Note – To view the status of the resynchronization process, use the /usr/sbin/metastat | grep Resync command.

12. Determine the physical device path to the alternate root (/) device you selected in step 6 (as reported by the Solaris 10 OS).

This varies by system. Use the ls -l command.
13. Use the init 0 command to shut the system down to the OpenBoot PROM level, and then use the show-disks command to determine the path to the alternate root (/) device (as reported by the OpenBoot PROM).

14. Define a backup root (/) device alias.
15. Add the backup_root device alias to the boot-device variable.
You should retain the alias for the primary boot disk.
16. Test the ability to boot the secondary root (/) submirror and log in as root when the boot process completes.
17. Verify the status of the root (/) submirrors.
18. Detach one submirror to make the root (/) mirror a one-way mirror.
19. Update the /etc/vfstab file to redefine the root (/) mount point using the original disk slice, and the /etc/system file to remove the forcedload statements.
20. Shut down the system to the OBP level.
21. If you changed your boot-device variable to an alternate boot path, complete the following steps:
 - a. Reset it to its default setting.
 - b. Boot the system to the multi-user milestone.
22. Clear the mirror and submirrors.
23. Remove all state database replicas.

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 Solutions: Mirroring the root (/) File System on SPARC-Based Systems

In this exercise, you complete the following:

- Mirror the root (/) file system
- Update the default boot device
- Unmirror the root (/) file system

Preparation

Due to the differences in Sun's SPARC and x86 architecture, this exercise can only be performed on SPARC-based systems. If you are using an x86-based system, perform the tasks in "Exercise: Mirroring the root (/) File System on x86/x64-Based Systems" on page 5-49.

This exercise mirrors the root (/) file system of your system's boot disk.

This exercise requires a second disk that is not in use. Steps in this exercise direct you to partition the second disk so that it has two partitions equal to or greater than the size of the root (/) partition on the boot disk, and at least two partitions to be used for state database replicas.

This exercise is performed on each individual system, so there is no need to work with a partner. Steps in these procedures are executed using the command line.

This exercise requires an understanding of how to use the format utility to partition disks.

Tasks and Solutions

This section provides the tasks and their solutions.

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Use the `df -h` command to list file systems in use, and the `format` utility to display the partition table for your system's boot disk. Record the following information:

- Disk slice used for the root (/) file system, and its size in megabytes. This will become the primary submirror: 0
As pre-defined for your lab system. (Slice 0 and 10.00 Gbytes, in this example.)
- Does the slice used for the root (/) file system start on cylinder 0 of the boot disk?
As pre-defined for your lab system. (No, in this example - it starts on cylinder 52319.) This information is required to determine what slice on the second disk to use as the secondary submirror, for the purpose of this exercise.
- Disk slice for state database replica 1:
As pre-defined for your lab system. (Slice 4, 15.94 Mbytes in this example.)
- Disk slice for state database replica 2:
As pre-defined for your lab system. (Slice 5, 15.94 Mbytes in this example.)

```
# df -h
Filesystem      size   used  avail capacity  Mounted on
/dev/dsk/c0t0d0s0    9.8G  4.5G   5.2G   47%       /
/devices          0K     0K     0K     0%       /devices
ctfs             0K     0K     0K     0%       /system/contract
proc             0K     0K     0K     0%       /proc
mnttab           0K     0K     0K     0%       /etc/mnttab
swap             2.5G  1.6M   2.5G   1%       /etc/svc/volatile
objfs            0K     0K     0K     0%       /system/object
sharefs           0K     0K     0K     0%       /etc/dfs/sharetab
/platform/sun4u-us3/lib/libc_psr/libc_psr_hwcap1.so.1
                  9.8G  4.5G   5.2G   47%       /platform/sun4u-
us3/lib/libc_psr.so.1
/platform/sun4u-us3/lib/sparcv9/libc_psr/libc_psr_hwcap1.so.1
                  9.8G  4.5G   5.2G   47%       /platform/sun4u-
us3/lib/sparcv9/libc_psr.so.1
fd                0K     0K     0K     0%       /dev/fd
swap              2.5G   80K   2.5G   1%       /tmp
swap              2.5G   48K   2.5G   1%       /var/run
/dev/dsk/c0t0d0s3    3.9G  4.0M   3.9G   1%       /myzone
/dev/dsk/c0t0d0s7    9.8G  10M   9.7G   1%       /export/home
# format
Searching for disks...done
```

AVAILABLE DISK SELECTIONS:

0. c0t0d0 <ST3120026A cyl 57459 alt 2 hd 16 sec 255>
 /pci@1e,600000/ide@d/dad@0,0
1. c0t1d0 <DEFAULT cyl 38307 alt 2 hd 16 sec 255>
 /pci@1e,600000/ide@d/dad@1,0

Exercise Solutions: Mirroring the root (/) File System on SPARC-Based Systems

```
Specify disk (enter its number): 0
selecting c0t0d0
[disk formatted, no defect list found]
(output omitted)
format> partition
(output omitted)
partition> print
Current partition table (original):
Total disk cylinders available: 57459 + 2 (reserved cylinders)
```

| Part | Tag | Flag | Cylinders | Size | Blocks |
|------|------------|------|---------------|----------|-----------------------|
| 0 | root | wm | 52319 - 57458 | 10.00GB | (5140/0/0) 20971200 |
| 1 | swap | wu | 0 - 1027 | 2.00GB | (1028/0/0) 4194240 |
| 2 | backup | wm | 0 - 57458 | 111.79GB | (57459/0/0) 234432720 |
| 3 | unassigned | wm | 45123 - 47178 | 4.00GB | (2056/0/0) 8388480 |
| 4 | unassigned | wm | 45115 - 45122 | 15.94MB | (8/0/0) 32640 |
| 5 | unassigned | wm | 45107 - 45114 | 15.94MB | (8/0/0) 32640 |
| 6 | unassigned | wm | 0 | 0 | (0/0/0) 0 |
| 7 | home | wm | 47179 - 52318 | 10.00GB | (5140/0/0) 20971200 |

```
partition> q
(output omitted)
format> q
#
```

3. Use the format utility to partition your *spare* disk so that it includes the partitions listed:
- Set the size of slice 0 to be equal to or greater than the disk slice used for the root (/) file system. This slice is a candidate to become the secondary submirror.
 - Set the size of slice 1 to be equal to or greater than the disk slice used for the root (/) file system. This slice is a candidate to become the secondary submirror.
 - Set the size of slice 6 to be at least 16 Mbytes. This slice will be used for state database replica 3.
 - Set the size of slice 7 to be at least 16 Mbytes. This slice will be used for state database replica 4.

```
# format
Searching for disks...done
```

AVAILABLE DISK SELECTIONS:

0. c0t0d0 <ST3120026A cyl 57459 alt 2 hd 16 sec 255>
/pci@1e,600000/ide@d/dad@0,0
1. c0t1d0 <DEFAULT cyl 38307 alt 2 hd 16 sec 255>
/pci@1e,600000/ide@d/dad@1,0

```
Specify disk (enter its number): 1
selecting c0t1d0
[disk formatted, no defect list found]
```

Exercise Solutions: Mirroring the root (/) File System on SPARC-Based Systems

```
(output omitted)
format> partition
(output omitted)
partition> print
Current partition table (original):
Total disk cylinders available: 38307 + 2 (reserved cylinders)
```

| Part | Tag | Flag | Cylinders | Size | Blocks |
|------|------------|------|---------------|---------|-----------------------|
| 0 | root | wm | 0 - 5654 | 11.00GB | (5655/0/0) 23072400 |
| 1 | swap | wu | 5655 - 11309 | 11.00GB | (5655/0/0) 23072400 |
| 2 | backup | wu | 0 - 38306 | 74.53GB | (38307/0/0) 156292560 |
| 3 | unassigned | wm | 0 | 0 | (0/0/0) 0 |
| 4 | unassigned | wm | 0 | 0 | (0/0/0) 0 |
| 5 | unassigned | wm | 0 | 0 | (0/0/0) 0 |
| 6 | unassigned | wm | 11310 - 11318 | 17.93MB | (9/0/0) 36720 |
| 7 | unassigned | wm | 11319 - 11327 | 17.93MB | (9/0/0) 36720 |

```
partition> q
(output omitted)
format> q
```

4. Determine the names of Solaris Volume Manager objects to use for this exercise:
 - Volume to map to the root (/) file system primary submirror:
As defined for your lab system. (The examples use d11.)
 - Volume to map to the root (/) file system secondary submirror:
As defined for your lab system. (The examples use d12.)
 - Volume to map to the root (/) file system mirror:
As defined for your lab system. (The examples use d10.)
5. Create a sufficient number of state database replicas to support the majority consensus algorithm used in the Solaris Volume Manager software.

Example:

```
# /usr/sbin/metadb -a -f c0t0d0s4
# /usr/sbin/metadb -a c0t0d0s5
# /usr/sbin/metadb -a c0t1d0s6
# /usr/sbin/metadb -a c0t1d0s7
#
```

What is the minimum number of state database replicas necessary to support the majority consensus algorithm?

As a best practice, you should use three state database replicas as the minimum to support the majority consensus algorithm.

6. Create a RAID-0 volume to use as the root (/) file system's primary submirror.

```
# /usr/sbin/metainit -f d11 1 1 c0t0d0s0
d11: Concat/Stripe is setup
#
```

(The variable points to the root (/) slice.)

7. Create a RAID 0 volume on the secondary drive to use as the root (/) file system's secondary submirror.

You should refer to step 2 to determine which of the following conditions is true.

- If the root slice on your boot disk *starts on* cylinder 0, use slice 0 on the second disk as the secondary submirror.

```
# /usr/sbin/metainit d12 1 1 c0t1d0s0
d12: Concat/Stripe is setup
#
```

- If the root slice on your boot disk *does not start on* cylinder 0, use slice 1 on the second disk as the secondary submirror.

```
# /usr/sbin/metainit d12 1 1 c0t1d0s1
d12: Concat/Stripe is setup
#
```

8. Create a RAID-1 volume as a one-way mirror using the root (/) file system primary submirror as the source of the mirror's data.

```
# /usr/sbin/metainit d10 -m d11
d10: Mirror is setup
#
```

9. Update the /etc/vfstab file to use the RAID-1 volume as the mount point for the root (/) file system and observe the changes to the /etc/vfstab and the /etc/system files.

```
# cat /etc/vfstab
(output omitted)
fd      -        /dev/fd fd      -      no      -
/proc   -        /proc   proc   -      no      -
/dev/dsk/c0t0d0s1      -      -      swap   -      no      -
/dev/dsk/c0t0d0s0      /dev/rdsk/c0t0d0s0   /      ufs    1      no      -
(output omitted)
# cat /etc/system
(output omitted - the file contains only comments)
# /usr/sbin/metaroot d10
# cat /etc/vfstab
(output omitted)
```

Exercise Solutions: Mirroring the root (/) File System on SPARC-Based Systems

```
fd      -      /dev/fd  fd      -      no      -
/proc   -      /proc    proc    -      no      -
/dev/dsk/c0t0d0s1   -      -      swap    -      no      -
/dev/md/dsk/d10  /dev/md/rdsk/d10   /      ufs     1      no      -
(output omitted)
# cat /etc/system
(output omitted)
* Begin MDD root info (do not edit)
rootdev:/pseudo/	md@0:0,10,blk
* End MDD root info (do not edit)
#
```

10. Reboot the system, and then log in as root.

```
# init 6
```

11. Attach the RAID-0 volume used as the root (/) file system's secondary submirror to the RAID-1 volume and allow the mirror synchronization to complete before continuing.

```
# /usr/sbin/metattach d10 d12
d10: submirror d12 is attached
#
```

What is the primary reason for using the command line to attach a secondary submirror to a mirror?

The primary reason for using the command line to attach a secondary submirror to a mirror is to force a resynchronization of the data between the primary and secondary submirror.



Note – To view the status of the resynchronization process, use the `/usr/sbin/metastat | grep Resync` command.

12. Determine the physical device path to the alternate root (/) device you selected in step 6 (as reported by the Solaris 10 OS).

This varies by system. Use the `ls -l` command.

```
# ls -l /dev/dsk/c0t1d0s1
lrwxrwxrwx  1 root      root          43 Oct  6 20:42 /dev/dsk/c0t1d0s1 -
> ../../devices/pci@1e,600000/ide@d/dad@1,0:b
```

13. Use the `init 0` command to shut the system down to the OpenBoot PROM level, and then use the `show-disks` command to determine the path to the alternate root (/) device (as reported by the OpenBoot PROM).

This varies by system. For example:

```
# init 0
(output omitted)
ok
```

Exercise Solutions: Mirroring the root (/) File System on SPARC-Based Systems

```

ok show-disks
a) /ramdisk-root
b) /pci@1e,600000/ide@d/cdrom
c) /pci@1e,600000/ide@d/disk
q) NO SELECTION
Enter Selection, q to quit: c
/pci@1e,600000/ide@d/disk has been selected.
Type ^Y (Control-Y) to insert it in the command line.
e.g. ok nvalias mydev ^Y
      for creating devalias mydev for /pci@1e,600000/ide@d/disk
ok

```

14. Define a backup root (/) device alias.

This varies by system. Use the nvalias command. For example:

```
ok nvalias backup_root /pci@1e,600000/ide@d/disk@1,0:b
```

15. Add the backup_root device alias to the boot-device variable.

You should retain the alias for the primary boot disk.

This varies by system. Use a combination of the printenv and setenv commands.

```

ok printenv boot-device
boot-device = disk net
ok setenv boot-device disk backup_root
boot-device = disk backup_root

```

16. Test the ability to boot the secondary root (/) submirror and log in as root when the boot process completes.

```
ok boot backup_root
```

17. Verify the status of the root (/) submirrors.

```

# /usr/sbin/metastat d10
d10: Mirror
  Submirror 0: d11
    State: Okay
  Submirror 1: d12
    State: Okay
  Pass: 1
  Read option: roundrobin (default)
  Write option: parallel (default)
  Size: 20971200 blocks (10.0 GB)

d11: Submirror of d10
  State: Okay
  Size: 20971200 blocks (10.0 GB)
  Stripe 0:
    Device      Start Block  Dbase      State Reloc Hot Spare
    c0t0d0s0          0       No        Okay     Yes

```

```
d12: Submirror of d10
  State: Okay
  Size: 23072400 blocks (11 GB)
  Stripe 0:
    Device      Start Block  Dbase      State Reloc Hot Spare
    c0t1d0s1        0       No        Okay   Yes
```

Device Relocation Information:

| | | |
|--------|-------|--|
| Device | Reloc | Device ID |
| c0t0d0 | Yes | id1,dad@AST3120026A=4JT0VAZE |
| c0t1d0 | Yes | id1,dad@AHDS728080PLAT20=_____PFD242SDTWAVDB |
| # | | |

18. Detach one submirror to make the root (/) mirror a one-way mirror.

```
# /usr/sbin/metadetach d10 d12
d10: submirror d12 is detached
```

19. Update the /etc/vfstab file to redefine the root (/) mount point using the original disk slice, and the /etc/system file to remove the forceunload statements.

```
# /usr/sbin/mataroot /dev/dsk/c0t0d0s0
```

20. Shut down the system to the OBP level.

```
# init 0
```

21. If you changed your boot-device variable to an alternate boot path, complete the following steps:

- a. Reset it to its default setting.
- b. Boot the system to the multi-user milestone.

```
ok set-default boot-device
ok boot
```

22. Clear the mirror and submirrors.

```
# /usr/sbin/metaclear -r d10
d10: Mirror is cleared
d11: Concat/Stripe is cleared
# /usr/sbin/metaclear d12
d12: Concat/Stripe is cleared
#
```

23. Remove all state database replicas.

```
# /usr/sbin/metadb -d c0t0d0s4
# /usr/sbin/metadb -d c0t0d0s5
# /usr/sbin/metadb -d c0t1d0s6
# /usr/sbin/metadb -d -f c0t1d0s7
```

Exercise: Mirroring the root (/) File System on x86/x64-Based Systems

In this exercise, you complete the following:

- Mirror the root (/) file system
- Make the mirrored disk bootable
- Unmirror the root (/) file system

Preparation

This exercise only runs on x86-based systems. If you are using a SPARC-based system, perform the tasks in “Exercise: Mirroring the root (/) File System on SPARC-Based Systems” on page 5-36.

This exercise assumes students know how to use the `format` utility to create `fdisk` partitions, and disk slices, if required.

The spare disk must have these structures defined:

- One Solaris `fdisk` partition that uses the entire spare disk.
- One slice that is equal to or larger than the root (/) slice of the system disk.
- Two small slices to hold state database replicas.

Table 5-4 defines sizes for using slice 0 or 1 for the root mirror, and slices 3 and 4 to hold state database replicas.

Table 5-4 Slice Information

| Slice | Size | Use |
|-------|---|------------------------|
| 0 | 10240 Mbytes | Possible root mirror |
| 1 | 10240 Mbytes | Possible root mirror |
| 3 | 16 Mbytes | State database replica |
| 4 | 16 Mbytes | State database replica |
| 5 | 10240 Mbytes (or remainder of disk) | Unassigned |

Table 5-4 Slice Information (Continued)

| Slice | Size | Use |
|-------|----------|------------|
| 6 | 0 Mbytes | Unassigned |
| 7 | 0 Mbytes | Unassigned |

This exercise is performed on each individual system, so there is no need to work with a partner for this exercise.

Tasks

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Use the `df` command to list file systems in use. From the `df` command output, verify that only the boot disk has file systems in use.
3. Use the `format` utility to gather the following information:
 - Record the name of the spare disk that you will use to mirror the root file system.
 - What is the size in megabytes of the slice used for the root (/) file system? This will become the primary submirror:
 - Which two slices on the boot disk use less than 16 Mbytes of space?
4. If necessary, use the `format` utility to establish the required Solaris `fdisk` partition and slices on disk. To do this, complete the following steps;
 - a. In the `format` utility, select the spare disk, and enter the `fdisk` menu.
 - b. If the following message displays, accept the default `fdisk` partition. Then, enter the `fdisk` menu.

No `fdisk` table exists. The default partition for the disk is:

a 100% "SOLARIS System" partition

Type "y" to accept the default partition, otherwise type "n" to edit the partition table.

y
format> **fdisk**

- c. If a Solaris fdisk partition exists, the fdisk menu lists the partition. Verify that the Solaris fdisk partition is configured as described below, and use the fdisk menu to make any corrections required. Save any changes you make.
- Verify that the Solaris fdisk partition uses 100% of the disk.
 - Verify that the Solaris fdisk partition *is not marked* as the active partition in the Status column.

The fdisk menu lists a correctly configured Solaris fdisk partition as described in this example:

```
Total disk size is 9729 cylinders  
Cylinder size is 16065 (512 byte) blocks
```

| Partition | Status | Type | Cylinders | | | % |
|-----------|--------|---------|-----------|------|--------|-----|
| | | | Start | End | Length | |
| 1 | | Solaris | 1 | 9728 | 9728 | 100 |

SELECT ONE OF THE FOLLOWING:

1. Create a partition
2. Specify the active partition
3. Delete a partition
4. Change between Solaris and Solaris2 Partition IDs
5. Exit (update disk configuration and exit)
6. Cancel (exit without updating disk configuration)

Enter Selection:

- d. If disk slices on the spare disk are not correctly defined, use the partition menu in the format utility to create slices within the Solaris fdisk partition. Use the information in Table 5-5 to determine the correct slice sizes.
- e. If you have defined new slices on the spare disk, write the partition table to disk, and quit the format utility.
5. Use the installgrub command to install the stage1 and stage2 programs onto the Solaris fdisk partition on the spare disk.
6. Determine the names of Solaris Volume Manager objects to use for this exercise:
- Volume to map to the root (/) file system primary submirror:

Exercise: Mirroring the root (/) File System on x86/x64-Based Systems

- Volume to map to the root (/) file system secondary submirror:

 - Volume to map to the root (/) file system mirror:

7. Create a sufficient number of state database replicas to support the majority consensus algorithm used in the Solaris Volume Manager software.
- What is the minimum number of state database replicas necessary to support the majority consensus algorithm?
-
8. Create a RAID-0 volume to use as the root (/) file system's primary submirror.
9. Create a RAID-0 volume to use as the root (/) file system's secondary submirror.
10. Create a RAID-1 volume as a one-way mirror using the root (/) file system primary submirror as the source of the mirror's data.
11. Use the metaroot command to update the /etc/vfstab file to use the RAID-1 volume as the mount point for the root (/) file system. Observe the changes to the /etc/vfstab and the /etc/system files.
12. Reboot the system, and then log in as root.
13. Attach the RAID-0 volume used as the root (/) file system's secondary submirror to the RAID-1 volume, and allow the mirror synchronization to complete before continuing.
- What is the primary reason for using the command line to attach a secondary submirror to a mirror?
-

Note – To view the status of the resynchronization process, use the /usr/sbin/metastat | grep Resync command.

-
14. Use the metastat command to display the status of the root (/) submirrors. Verify that the state of each submirror is listed as Okay.
15. Detach the secondary submirror to make the root (/) mirror a one-way mirror.



Exercise: Mirroring the root (/) File System on x86/x64-Based Systems

16. Use the metaroot command to redefine the root (/) device in the /etc/vfstab file so it uses the original disk slice, and to remove the forceunload statements from the /etc/system file.
17. Reboot the system.
18. Use the metaclear command to remove the mirror and submirrors, and use metastat to verify that they no longer exist.
19. Remove all state database replicas.

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 Solutions: Mirroring the root (/) File System on x86/x64-Based Systems

In this exercise, you complete the following:

- Mirror the root (/) file system
- Make the mirrored disk bootable
- Unmirror the root (/) file system

Preparation

This exercise only runs on x86-based systems. If you are using a SPARC-based system, perform the tasks in “Exercise: Mirroring the root (/) File System on SPARC-Based Systems” on page 5-36.

This exercise assumes students know how to use the `format` utility to create `fdisk` partitions, and disk slices, if required.

The spare disk must have these structures defined:

- One Solaris `fdisk` partition that uses the entire spare disk.
- One slice that is equal to or larger than the root (/) slice of the system disk.
- Two small slices to hold state database replicas.

Table 5-5 defines sizes for using slice 0 or 1 for the root mirror, and slices 3 and 4 to hold state database replicas.

Table 5-5 Slice Information

| Slice | Size | Use |
|-------|---|------------------------|
| 0 | 10240 Mbytes | Possible root mirror |
| 1 | 10240 Mbytes | Possible root mirror |
| 3 | 16 Mbytes | State database replica |
| 4 | 16 Mbytes | State database replica |
| 5 | 10240 Mbytes (or remainder of disk) | Unassigned |

Table 5-5 Slice Information (Continued)

| Slice | Size | Use |
|-------|----------|------------|
| 6 | 0 Mbytes | Unassigned |
| 7 | 0 Mbytes | Unassigned |

This exercise is performed on each individual system, so there is no need to work with a partner for this exercise.

Tasks and Solutions

This section provides the exercise steps and their solutions.

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Use the `df` command to list file systems in use. From the `df` command output, verify that only the boot disk has file systems in use.

In this example, only `c1d0` is in use:

```
# df -h
Filesystem      size  used  avail capacity  Mounted on
/dev/dsk/c0d1s0   9.9G  4.3G   5.5G    44%       /
/devices          0K   0K    0K     0%   /devices
ctfs             0K   0K    0K     0%   /system/contract
proc              0K   0K    0K     0%   /proc
mnttab            0K   0K    0K     0%   /etc/mnttab
swap              4.2G  968K   4.2G    1%   /etc/svc/volatile
objfs             0K   0K    0K     0%   /system/object
sharefs            0K   0K    0K     0%   /etc/dfs/sharetab
/usr/lib/libc/libc_hwcap1.so.1
                  9.9G  4.3G   5.5G    44%   /lib/libc.so.1
fd                 0K   0K    0K     0%   /dev/fd
swap              4.2G   80K   4.2G    1%   /tmp
swap              4.2G   28K   4.2G    1%   /var/run
/dev/dsk/c0d1s3   3.9G  4.0M   3.9G    1%   /myzone
/dev/dsk/c0d1s7   3.9G  4.0M   3.9G    1%   /export/home
```

3. Use the `format` utility to gather the following information:
 - Record the name of the spare disk that you will use to mirror the root file system.

According to the `format` command output, two disks exist in the system, `c0d1` and `c1d0` in this example. The `df -h` command output indicates the system currently has no file systems mounted from `c1d0`, so `c1d0` is the spare disk.

- What is the size in megabytes of the slice used for the root (/) file system? This will become the primary submirror:
As pre-defined for your lab system. (Slice 0 and 10.00 Gbytes, in this example.)
- Which two slices on the boot disk use less than 16 Mbytes of space?
Slices 4 and 5. These are the slices reserved for state database replicas on the boot disk.

```
# format
Searching for disks...done
```

AVAILABLE DISK SELECTIONS:

0. c1d0 <DEFAULT cyl 60548 alt 2 hd 128 sec 63>
/pci@0,0/pci-ide@7/ide@0/cmdk@0,0
1. c2d0 <DEFAULT cyl 30398 alt 2 hd 255 sec 63>
/pci@0,0/pci-ide@7/ide@1/cmdk@0,0

Specify disk (enter its number): **0**

selecting c1d0

(output omitted)

format> **part**

(output omitted)

partition> **print**

Current partition table (original):

Total disk cylinders available: 9726 + 2 (reserved cylinders)

| Part | Tag | Flag | Cylinders | Size | Blocks |
|------|------------|------|-------------|---------|----------------------|
| 0 | root | wm | 8420 - 9725 | 10.00GB | (1306/0/0) 20980890 |
| 1 | swap | wu | 3 - 263 | 2.00GB | (261/0/0) 4192965 |
| 2 | backup | wm | 0 - 9725 | 74.50GB | (9726/0/0) 156248190 |
| 3 | unassigned | wm | 6591 - 7113 | 4.01GB | (523/0/0) 8401995 |
| 4 | unassigned | wm | 6589 - 6590 | 15.69MB | (2/0/0) 32130 |
| 5 | unassigned | wm | 6587 - 6588 | 15.69MB | (2/0/0) 32130 |
| 6 | unassigned | wm | 0 | 0 | (0/0/0) 0 |
| 7 | home | wm | 7114 - 8419 | 10.00GB | (1306/0/0) 20980890 |
| 8 | boot | wu | 0 - 0 | 7.84MB | (1/0/0) 16065 |
| 9 | alternates | wu | 1 - 2 | 15.69MB | (2/0/0) 32130 |

```
partition> q
(output omitted)
format>
```

4. If necessary, use the format utility to establish the required Solaris fdisk partition and slices on disk. Complete the following steps;
 - a. In the format utility, select the spare disk, and enter the fdisk menu.

```
format> disk
```

Exercise Solutions: Mirroring the root (/) File System on x86/x64-Based Systems

AVAILABLE DISK SELECTIONS:

0. c0d1 <DEFAULT cyl 9726 alt 2 hd 255 sec 63>
 /pci@0,0/pci-ide@1f,1/ide@0/cmdk@1,0
1. c1d0 <DEFAULT cyl 9726 alt 2 hd 255 sec 63>
 /pci@0,0/pci-ide@1f,2/ide@0/cmdk@0,0

Specify disk (enter its number) [0]: 1

selecting c1d0

Controller working list found
[disk formatted, defect list found]
format> **fdisk**

- b. If the following message displays, accept the default fdisk partition. Then, enter the fdisk menu.

No fdisk table exists. The default partition for the disk is:

a 100% "SOLARIS System" partition

Type "y" to accept the default partition, otherwise type "n" to edit the partition table.

y

format> **fdisk**

- c. If a Solaris fdisk partition exists, the fdisk menu lists the partition. Verify that the Solaris fdisk partition is configured as described below, and use the fdisk menu to make any corrections required. Save any changes you make.
- Verify that the Solaris fdisk partition uses 100% of the disk.
 - Verify that the Solaris fdisk partition *is not marked* as the active partition in the Status column.

The fdisk menu lists a correctly configured Solaris fdisk partition as described in this example:

```
Total disk size is 9729 cylinders
Cylinder size is 16065 (512 byte) blocks
```

| Partition | Status | Type | Cylinders | | | |
|-----------|--------|---------|-----------|------|--------|-----|
| | | | Start | End | Length | % |
| 1 | | Solaris | 1 | 9728 | 9728 | 100 |

SELECT ONE OF THE FOLLOWING:

1. Create a partition
2. Specify the active partition
3. Delete a partition
4. Change between Solaris and Solaris2 Partition IDs
5. Exit (update disk configuration and exit)
6. Cancel (exit without updating disk configuration)

Enter Selection:

- d. If disk slices on the spare disk are not correctly defined, use the partition menu in the format utility to create slices within the Solaris fdisk partition. Use the information in Table 5-5 to determine the correct slice sizes. When complete, the correct list of slices looks like this example from an Ultra 20:

| Part | Tag | Flag | Cylinders | Size | Blocks |
|------|------------|------|-------------|---------|----------------------|
| 0 | unassigned | wm | 3 - 1308 | 10.00GB | (1306/0/0) 20980890 |
| 1 | unassigned | wm | 1309 - 2614 | 10.00GB | (1306/0/0) 20980890 |
| 2 | backup | wu | 0 - 9725 | 74.50GB | (9726/0/0) 156248190 |
| 3 | unassigned | wm | 2615 - 2617 | 23.53MB | (3/0/0) 48195 |
| 4 | unassigned | wm | 2618 - 2620 | 23.53MB | (3/0/0) 48195 |
| 5 | unassigned | wu | 2621 - 3926 | 10.00GB | (1306/0/0) 20980890 |
| 6 | unassigned | wu | 0 | 0 | (0/0/0) 0 |
| 7 | unassigned | wu | 0 | 0 | (0/0/0) 0 |
| 8 | boot | wu | 0 - 0 | 7.84MB | (1/0/0) 16065 |
| 9 | alternates | wm | 1 - 2 | 15.69MB | (2/0/0) 32130 |

- e. If you have defined new slices on the spare disk, write the partition table to disk, and quit the format utility.
5. Use the installgrub command to install the stage1 and stage2 programs onto the Solaris fdisk partition on the spare disk.

```
# installgrub /boot/grub/stage1 /boot/grub/stage2 /dev/rdsk/c1d0s0
Solaris boot partition inactive.
stage1 written to partition 0 sector 0 (abs 16065)
stage2 written to partition 0, 272 sectors starting at 50 (abs 16115)
#
```

6. Determine the names of Solaris Volume Manager objects to use for this exercise:
- Volume to map to the root (/) file system primary submirror:
As defined for your lab system. (The examples use d11.)
 - Volume to map to the root (/) file system secondary submirror:
As defined for your lab system. (The examples use d12.)
 - Volume to map to the root (/) file system mirror:
As defined for your lab system. (The examples use d10.)
7. Create a sufficient number of state database replicas to support the majority consensus algorithm used in the Solaris Volume Manager software.

For example:

```
# /usr/sbin/metadb -a -f c0d1s4
# /usr/sbin/metadb -a c0d1s5
# /usr/sbin/metadb -a c1d0s3
```

```
# /usr/sbin/metadb -a c1d0s4
```

What is the minimum number of state database replicas necessary to support the majority consensus algorithm?

As a best practice, you should use three state database replicas as the minimum to support the majority consensus algorithm.

8. Create a RAID-0 volume to use as the root (/) file system's primary submirror.

```
# /usr/sbin/metainit -f d11 1 1 c0d1s0
```

d11: Concat/Stripe is setup

(The variable points to the root (/) slice.)

9. Create a RAID-0 volume to use as the root (/) file system's secondary submirror.

```
# /usr/sbin/metainit d12 1 1 c1d0s0
```

d12: Concat/Stripe is setup

10. Create a RAID-1 volume as a one-way mirror using the root (/) file system primary submirror as the source of the mirror's data.

```
# /usr/sbin/metainit d10 -m d11
```

d10: Mirror is setup

11. Use the metaroot command to update the /etc/vfstab file to use the RAID-1 volume as the mount point for the root (/) file system. Observe the changes to the /etc/vfstab and the /etc/system files.

```
# cat /etc/vfstab
```

| device | device | mount | FS | fsck | mount |
|------------------|------------------|------------|----|------|--------|
| mount | | | | | |
| #to mount | to fsck | point | | | |
| options | | | | | |
| # | | | | | |
| fd | - | /dev/fd fd | - | no | - |
| /proc | - | /proc proc | - | no | - |
| /dev/dsk/c0d1s1 | - | - swap | - | no | - |
| /dev/dsk/c0d1s0 | /dev/rdsk/c0d1s0 | | / | ufs | 1 no - |
| (output omitted) | | | | | |

```
# cat /etc/vfstab
```

| device | device | mount | FS | fsck | mount |
|-----------------|---------|------------|----|------|-------|
| mount | | | | | |
| #to mount | to fsck | point | | | |
| options | | | | | |
| # | | | | | |
| fd | - | /dev/fd fd | - | no | - |
| /proc | - | /proc proc | - | no | - |
| /dev/dsk/c0d1s1 | - | - swap | - | no | - |

```
/dev/md/dsk/d10 /dev/md/rdsk/d10      /      ufs      1      no      -
(output omitted)
# cat /etc/system
(output omitted)
* Begin MDD root info (do not edit)
rootdev:/pseudo/	md@0:0,10,blk
* End MDD root info (do not edit)
#
#
```

12. Reboot the system, and then log in as root.

```
# init 6
```

13. Attach the RAID-0 volume used as the root (/) file system's secondary submirror to the RAID-1 volume, and allow the mirror synchronization to complete before continuing.

```
# /usr/sbin/metattach d10 d12
d10: submirror d12 is attached
#
```

What is the primary reason for using the command line to attach a secondary submirror to a mirror?

The primary reason for using the command line to attach a secondary submirror to a mirror is to force a resynchronization of the data between the primary and secondary submirror.

 **Note** – To view the status of the resynchronization process, use the /usr/sbin/metastat | grep Resync command.

14. Use the metastat command to display the status of the root (/) submirrors. Verify that the state of each submirror is listed as Okay.

```
# /usr/sbin/metastat
d10: Mirror
    Submirror 0: d11
        State: Okay
    Submirror 1: d12
        State: Okay
    Pass: 1
    Read option: roundrobin (default)
    Write option: parallel (default)
    Size: 20980890 blocks (10 GB)

d11: Submirror of d10
    State: Okay
    Size: 20980890 blocks (10 GB)
    Stripe 0:
        Device   Start Block  Dbase      State Reloc Hot Spare
        c0d1s0          0     No        Okay     Yes
d12: Submirror of d10
```

Exercise Solutions: Mirroring the root (/) File System on x86/x64-Based Systems

```
State: Okay
Size: 20980890 blocks (10 GB)
Stripe 0:
Device Start Block Dbase      State Reloc Hot Spare
c1d0s0        0     No      Okay   Yes
```

Device Relocation Information:

```
Device Reloc Device ID
c1d0  Yes    id1,cmdk@AST380215AS=_____5QZ41H1T
c0d1  Yes    id1,cmdk@AHDS722580VLAT20=_____VN62GECHC0DX1C
#
```

15. Detach the secondary submirror to make the root (/) mirror a one-way mirror.

```
# /usr/sbin/metadetach d10 d12
```

16. Use the metaroot command to redefine the root (/) device in the /etc/vfstab file so it uses the original disk slice, and to remove the forceunload statements from the /etc/system file.

```
# /usr/sbin/metaroot /dev/dsk/c0d1s0
```

17. Reboot the system.

```
# init 6
```

18. Use the metaclear command to remove the mirror and submirrors, and use metastat to verify that they no longer exist.

```
# /usr/sbin/metaclear -r d10
```

d10: Mirror is cleared

d11: Concat/Stripe is cleared

```
# /usr/sbin/metaclear d12
```

d12: Concat/Stripe is cleared

```
# metastat
```

```
#
```

19. Remove all state database replicas.

```
# /usr/sbin/metadb -d c0d1s4
```

```
# /usr/sbin/metadb -d c0d1s5
```

```
# /usr/sbin/metadb -d c1d0s3
```

```
# /usr/sbin/metadb -d -f c1d0s4
```

Module 6

Configuring Role-Based Access Control (RBAC)

Objectives

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

- Describe RBAC fundamentals
- Describe component interaction within RBAC
- Manage RBAC

Introducing RBAC Fundamentals

In conventional UNIX systems, the root user (also referred to as the superuser) is the most powerful user, with the ability to read and write to any file, run all programs, and send kill signals to any process. Anyone who can become superuser can modify a site's firewall, alter the audit trail, and read confidential records.

In systems implementing RBAC, individual users can be assigned to roles, such as system administrator, network administrator, or operator. Roles are associated with rights profiles. The rights profiles list the rights to run specific commands and applications with escalated privileges.

Roles can also be assigned authorizations. An authorization grants access to restricted functions in RBAC compliant applications. RBAC compliant applications are linked to libsecdb so they can be checked for privileges.

Key RBAC Files

As well as roles, individual users may also be granted rights profiles and authorizations to specific applications. The authorizations, roles, rights profiles, and privileged commands are defined in four files.

- The /etc/user_attr file
- The /etc/security/prof_attr file
- The /etc/security/policy.conf file
- The /etc/security/exec_attr file

The user_attr File

The /etc/user_attr file contains user and role information that supplements the /etc/passwd and /etc/shadow files. The /etc/user_attr file lists the rights profiles and authorizations associated with users and roles.

When creating a new user account with no rights profiles, authorizations or roles, nothing is added to the file:

```
# useradd -m -d /export/home/chris chris
64 blocks
# grep chris /etc/user_attr
#
```

As each of the RBAC features are explained, the automatic modifications to this file are shown.

Roles

A role is a special identity, similar to a user account, for running privileged applications or commands that can be assumed by assigned users only.

While no predefined roles are shipped with the Solaris 10 OS, predefined rights profiles, or collections of privileges, can be associated with roles. To define a role, you assign the rights profiles to the role, as shown in Figure 6-1.

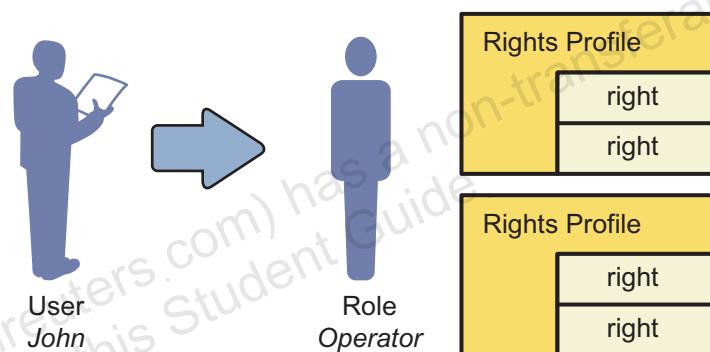


Figure 6-1 A Role With Two Rights Profiles

It is not possible to login as a role. A role can only be used by switching the user to the role with the `su` command. The `roles` command lists the roles a user has been assigned:

```
# roles root
No roles
# roles chris
No roles
```



Note – You can also set up the `root` user as a role through a manual process. This approach prevents users from logging in directly as the `root` user. Therefore, they must log in as themselves first, and then use the `su` command to assume the role.

Assigning Rights Profiles To Users

A rights profile, is a collection of rights that can be assigned to a user, as shown in Figure 6-2. The rights are commands or scripts which are run with special security attributes.

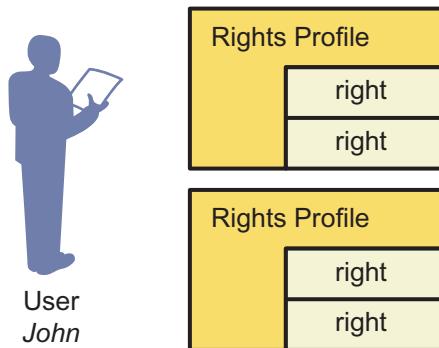


Figure 6-2 Rights Profile

Many examples of rights profiles are shipped with the Solaris 10 OS. The rights profile names and descriptions are defined in the /etc/security/prof_attr file. New rights profiles can be created by editing this file or using the Solaris Management Console (SMC). This example shows a few lines from that file.

```
# cat /etc/security/prof_attr
(output omitted)
All:::Execute any command as the user or role:help=RtAll.html
Log Management:::Manage log files:help=RtLogMngmnt.html
Media Backup:::Backup files and file systems:help=RtMediaBkup.html
Media Restore:::Restore files and file systems from
backups:help=RtMediaRestore.html
(output omitted)
```

Each line starts with the rights profile name. The middle fields are not used and the last two fields are a comment and a pointer to a help file. Help files are written in Hypertext Markup Language (HTML) and they can be customized if required. These HTML help files exist in the /usr/lib/help/auths/locale/C directory.

The rights profiles assigned to a user can be listed with the **profiles** command or through the Solaris Management Console. This example shows the default profiles assigned to every new user account:

```
# profiles chris
Basic Solaris User
All
```

Every account has the All rights profile. It allows any command to be executed but with special security attributes. Other rights profiles given to all new user accounts are defined in the /etc/security/policy.conf file. The Basic Solaris User rights profile is listed in this file:

```
# grep 'PROFS' /etc/security/policy.conf
PROFS_GRANTED=Basic Solaris User
```

Rights profiles can be assigned to a user account with the **usermod** command or the Solaris Management Console (SMC). This example shows the Printer Management rights profile being assigned to the chris user account:

```
# usermod -P "Printer Management" chris
# profiles chris
Printer Management
Basic Solaris User
All
```

This automatically updates the /etc/user_attr file as shown below:

```
# grep chris /etc/user_attr
chris::::type=normal;profiles=Printer Management
```

The new line for the user chris shows the new profile assignment. The file uses colons (:) to separate the fields on each line. The first field is the user name as it appears in the /etc/passwd and /etc/shadow files. The middle fields are reserved for future use, and the last field is a list of semicolon-separated (;) key-value pairs that describe the security attributes to be applied when the user runs commands.

The contents of a rights profile can be examined from the command line with the `-l` option of the `profiles` command or in the Solaris Management Console (SMC).

```
# profiles -l chris
```

Printer Management:

```
/etc/init.d/lp      euid=0, uid=0
/usr/bin/cancel     euid=lp, uid=lp
/usr/bin/lpset      egid=14
/usr/bin/lpstat     euid=0
/usr/lib/lp/local/accept   uid=lp
/usr/lib/lp/local/lpadmin    uid=lp, gid=8
/usr/lib/lp/lpsched   uid=0
/usr/sbin/accept    euid=lp, uid=lp
/usr/sbin/lpadmin   egid=14, uid=lp, gid=8
/usr/sbin/lpfilter  euid=lp, uid=lp
/usr/sbin/lpforms   euid=lp
/usr/sbin/lpmove    euid=lp
/usr/sbin/lpshut    euid=lp
/usr/sbin/lpusers   euid=lp
/usr/ucb/lpq       euid=0
/usr/ucb/lprm      euid=0
```

All:

```
*
```

The individual commands in the rights profile can be seen, along with the special security attributes with which they are executed.

This example shows the user `chris` being able to enable and disable a printer.

The /etc/security/exec_attr File

The /etc/security/exec_attr file holds the execution attributes. An execution attribute is associated with a rights profile name.

An execution attribute can be a command with no options or a script that contains a command with options. The only way to add options to a command is by using a script. You can use the (*) wildcard. Commands should have the full path.

Special security attributes refer to attributes, such as UID, EUID, GID, and EGID, that can be added to a process when the command is run. Only the users and roles assigned access to this rights profile can run the command with special security attributes.

The commands and special security attributes for the Printer Management rights profile are listed below:

```
# grep 'Printer Management' /etc/security/exec_attr
Printer Management:suser:cmd:::/etc/init.d/lp:euid=0;uid=0
Printer Management:suser:cmd:::/usr/bin/cancel:euid=lp;uid=lp
Printer Management:suser:cmd:::/usr/bin/lpset:egid=14
Printer Management:suser:cmd:::/usr/bin/lpstat:euid=0
Printer Management:suser:cmd:::/usr/lib/lp/local/accept:uid=lp
Printer Management:suser:cmd:::/usr/lib/lp/local/lpadmin:uid=lp;gid=8
Printer Management:suser:cmd:::/usr/lib/lp/lpsched:uid=0
Printer Management:suser:cmd:::/usr/sbin/accept:euid=lp;uid=lp
Printer Management:suser:cmd:::/usr/sbin/lpadmin:egid=14;uid=lp;gid=8
Printer Management:suser:cmd:::/usr/sbin/lpfILTER:euid=lp;uid=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/lphut:euid=lp
Printer Management:suser:cmd:::/usr/sbin/lpusers:euid=lp
Printer Management:suser:cmd:::/usr/ucb/lpq:euid=0
Printer Management:suser:cmd:::/usr/ucb/lprm:euid=0
```

Assigning Rights Profiles To Roles

The previous section described how to add rights profiles to user accounts. If a large number of user accounts require the same configuration and management of rights profiles, it can be easier to assign the rights profiles to a role and give the users access to the role.

Figure 6-3 shows the assignment of rights profiles to a role called `level1` and giving the `john` user account access to the role:

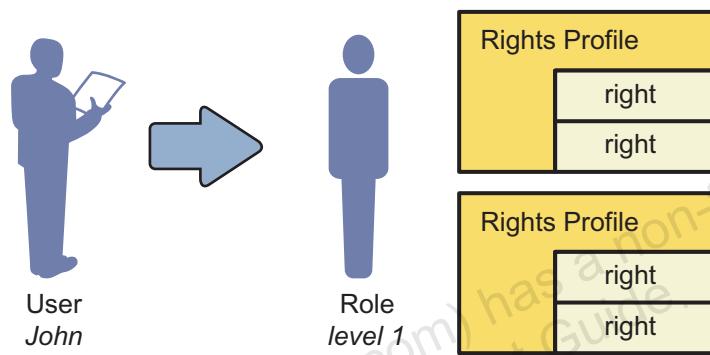


Figure 6-3 The Assignment of Profiles to Roles

Creating a Role

The `roleadd` command creates a role entry in the `/etc/passwd`, `/etc/shadow`, and `/etc/user_attr` files. Some common options include:

- `-c comment` A text string that provides a short description of the role.
- `-d dir` Specifies the home directory of the new role.
- `-m` Creates the new role's home directory if it does not already exist.
- `-P profile` Assigns rights profiles to the role. Use commas (,) to separate multiple rights profiles.

```
# roleadd -m -d /export/home/level1 -c "Level One Support" \
-P "Printer Management,Media Backup,Media Restore" level1
64 blocks
# passwd level1
New Password: level1
Re-enter new Password: level1
passwd: password successfully changed for level1
```

In this example, the `roleadd` command creates a new role called `level1`, builds the home directory, and assigns the role with rights profiles of Printer Management, Media Backup, and Media Restore. The role can not be used until a password is applied to it.



Note – The installation of the Solaris 10 OS has the Printer Management, Media Backup, and Media Restore rights profiles already defined in the `/etc/security/exec_attr` file and the `/etc/security/prof_attr` file.

The changes to the `/etc/passwd`, `/etc/shadow`, and `/etc/user_attr` files are shown below:

```
# grep level1 /etc/passwd
level1:x:102:1:Level One Support:/export/home/level1:/bin/pfsh

# grep level1 /etc/shadow
level1:CU8aQ64vTrZ.:12713::::::

# grep level1 /etc/user_attr
level1::::type=role;profiles=Printer Management,Media Backup,Media
Restore
```

The type of this account is role (`type=role`) and includes the rights profiles Printer Management, Media Backup, and Media Restore.

Modifying a Role

To modify the login information of a role on a system, use the `rolemod` command. The `rolemod` command changes the definition of the specified role and makes the appropriate login-related changes to the system file and file system. The fields in the `rolemod` command are:

| | |
|-----------------------------|---|
| <code>-e expire</code> | Specifies the expiration date for a role. |
| <code>-l new_logname</code> | Specifies the new login name for the role. |
| <code>-P profile</code> | Specifies one or more comma-separated rights profiles, as defined in the <code>/etc/security/prof_attr</code> file. |
| <code>-s shell</code> | Specifies the full path name of the program that is used as the role's shell when logging in. These shells are special versions of the Bourne shell (<code>sh</code>), C shell (<code>csh</code>), and Korn shell (<code>ksh</code>). |

This example modifies the role's rights profiles.

```
# rolemod -P profile1,profile2 -s /usr/bin/pfksh level1
```

In this example, the `rolemod` command assigns the `profile1` and `profile2` profiles and the `/usr/bin/pfksh` profile shell to the role named `level1`.

Purpose of the Profile Shells

A profile shell is a special type of shell that enables access to the privileged rights that are assigned to the rights profile. The standard UNIX shells can not be used, as they are not aware of the RBAC files, and do not consult them.

When the user executes a command, the profile shell searches the role's rights profiles and associated rights. If the same command appears in more than one profile, the profile shell uses the first matching entry. The profile shell executes the command with the attributes specified in the RBAC configuration files.

The profile shells are `pfsh`, `pfcsh`, and `pfksh`. These profile shells correspond to Bourne shell (`sh`), C shell (`csh`), and Korn shell (`ksh`), respectively.

Assigning Roles to Users

A user can have access to many roles. The useradd command or Solaris Management Console (SMC) can be used to define which roles a new user has access to. The example shows the useradd command being used with the -R option to define roles:

```
# useradd -m -d /export/home/paul -R level1 paul
64 blocks
# passwd paul
New Password: paul
Re-enter new Password: paul
passwd: password successfully changed for paul
```

The roles command lists the roles a user account has access to:

```
# roles paul
level1
```

The association between the paul user account and the level1 role is defined automatically in the /etc/user_attr file:

```
# grep paul /etc/user_attr
paul:::::type=normal;roles=level1
```

To add roles to an existing user account, use the usermod command or the Solaris Management Console (SMC). This example shows access to the level1 role being given to chris with the usermod command:

```
# usermod -R level1 chris
```

To remove all role access from a user account, use the usermod command or the Solaris Management Console (SMC). This example uses usermod to remove all role access from the chris account:

```
# usermod -R "" chris
```

Using Roles

As it is not possible to log in to a role account, log in as a regular user first. The **roles** command shows the roles available to your account.

```
$ id  
uid=103(paul) gid=1(other)  
$ roles  
level1
```

Switch the user to the role account with the **su** command.

```
$ su level1  
Password: level1  
$ id  
uid=102(level1) gid=1(other)
```

The **level1** role has the two default rights profiles and was configured with three extra rights profiles.

```
$ profiles  
Printer Management  
Media Backup  
Media Restore  
Basic Solaris User  
All
```

The Printer Management rights profile has a right which allows the **cancel** command to be run as the **lp** user.

```
$ lpstat -t  
scheduler is running  
system default destination: laser  
system for _default: host1 (as printer laser)  
device for laser: /dev/null  
_default accepting requests since Fri Oct 22 13:59:24 2004  
laser accepting requests since Fri Oct 22 13:59:24 2004  
printer laser disabled since Fri Oct 22 13:59:34 2004. available.  
    Changing Toner  
laser-8                      root          479   Oct 22 14:12  
$ cancel laser-8  
laser-8: cancelled
```

Authorizations

An authorization grants access to restricted functions in RBAC compliant applications. Some applications and commands in the Solaris 10 OS are written to check the authorizations of the user calling them. You cannot create new authorizations, however, you can create and assign authorizations to new applications.

The predefined authorizations are listed in the authorization attributes configuration file named /etc/security/auth_attr.

```
# cat /etc/security/auth_attr
(output omitted)
solaris.jobs.:::Job Scheduler:::help=JobHeader.html
solaris.jobs.admin:::Manage All Jobs:::help=AuthJobsAdmin.html
solaris.jobs.grant:::Delegate Cron & At Administration:::help=JobsGrant.html
solaris.jobs.user:::Manage Owned Jobs:::help=AuthJobsUser.html
(output omitted)
```

It identifies, by a unique string, what is being authorized. For example, the crontab command requires the solaris.jobs.admin authorization for a user to edit another user's crontab file.

A hierarchy of authorizations can be established. Table 6-1 shows how a hierarchy can be established.

Table 6-1 Role and Authorization Relationships

| Authorization | Action |
|---|--|
| solaris.admin.usermgr.read | Provides read but no write access to user configuration files. |
| solaris.admin.usermgr.read solaris.admin.usermgr.write | Provides read and write access to user configuration files. Cannot change passwords. |
| solaris.admin.usermgr.read solaris.admin.usermgr.write solaris.admin.usermgr.pswd | Provides read, write, and password access to user configuration files. |



Caution – An authorization that ends with the suffix grant permits a user to delegate any assigned authorizations that begin with the same prefix to other users.

For example, a role with the authorizations:

```
solaris.admin.usermgr.grant  
solaris.admin.usermgr.read
```

Can delegate the solaris.admin.usermgr.read authorization to another user.

A role with the authorizations:

```
solaris.admin.usermgr.grant  
solaris.admin.usermgr.*
```

Can delegate any of the authorizations with the solaris.admin.usermgr prefix to other users.

Default Authorizations

All users have the Basic Solaris User profile by default.

```
# profiles chris  
Printer Management  
Basic Solaris User  
All
```

The Basic Solaris User profile grants users access to all listed authorizations. The profiles=All field grants unrestricted access to all Solaris OS commands that have not been restricted by a definition in a previously listed authorization.

```
# grep 'Basic Solaris User' /etc/security/prof_attr
```

```
Basic Solaris User:::Automatically assigned rights:  
auths=solaris.profmgr.read,solaris.jobs.users,solaris.mail.  
mailq,  
solaris.admin.usermgr.read,solaris.admin.logsvc.read,  
solaris.admin.fsmgr.read,solaris.admin.serialmgr.read,  
solaris.admin.diskmgr.read,solaris.admin.procmgr.user,  
solaris.compsys.read,solaris.admin.printer.read,  
solaris.admin.prodreg.read,solaris.admin.dcmgr.read,  
solaris.snmp.read,solaris.project.read,solaris.admin.patchm  
gr.read,  
solaris.network.hosts.read,solaris.admin.volmgr.read;profil  
es=All; help=RtDefault.html
```

Other default authorizations for every user can be defined in the /etc/security/policy.conf file:

```
# grep 'AUTHS' /etc/security/policy.conf  
AUTHS_GRANTED=solaris.device.cdrw
```

This authorization is in the default /etc/security/policy.conf file as installed with the Solaris 10 OS.

Assigning Authorizations

Authorizations can be assigned to user accounts. Authorizations can also be assigned to roles or embedded in a rights profile which can be assigned to a user or role.

Figure 6-4 shows the authorization assignment permutations.

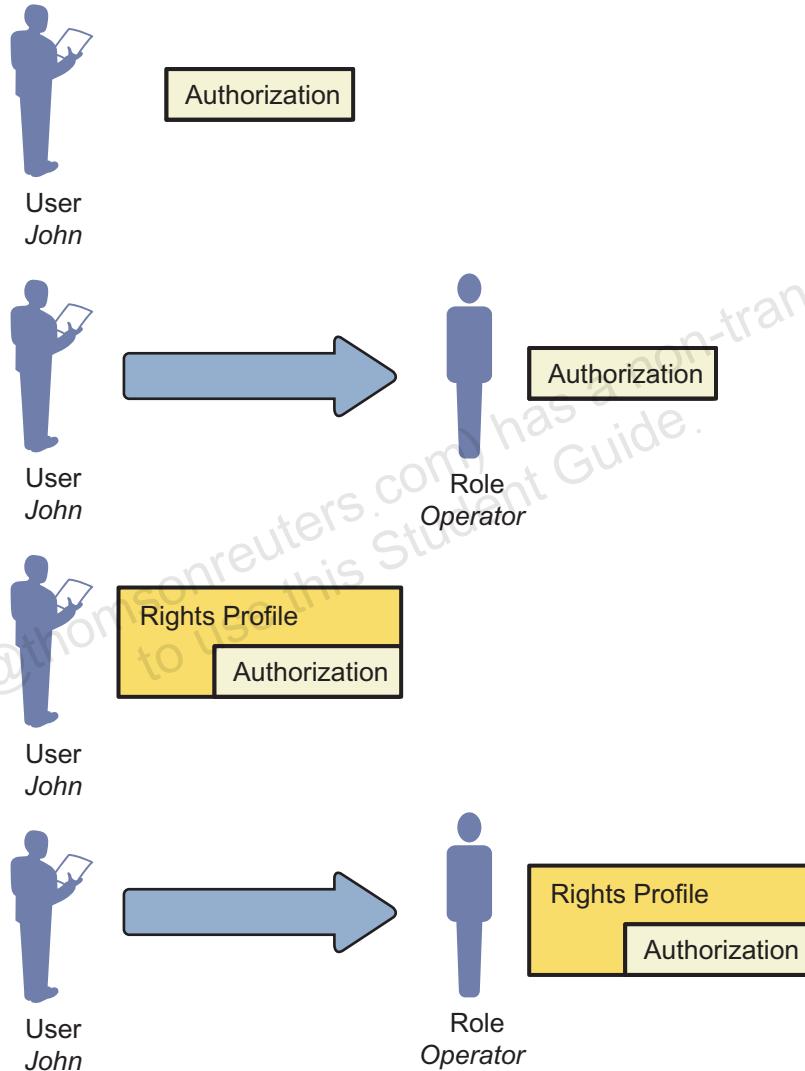


Figure 6-4 Authorization Assignment Permutations

Assigning Authorizations to User Accounts

The following example shows that a regular user is not permitted to look at another user's crontab file:

```
# su - chris
Sun Microsystems Inc.      SunOS 5.10          s10_68  Sep. 20, 2004
$ crontab -l root
crontab: you must be super-user to access another user's crontab file
$ exit
```

The authorization to manage other user's crontab file can be granted to the user from the command line. This example shows the usermod command being used with the -A option to add an authorization:

```
# usermod -A solaris.jobs.admin chris
```

The /etc/user_attr user attributes file has been automatically modified with this new information.

```
# grep chris /etc/user_attr
chris::::type=normal;auths=solaris.jobs.admin;profiles=Printer Management
```

The chris account, is a normal user account (type=normal), he has had the solaris.jobs.admin authorization and the Printer Management rights profile added previously. Use the auths command to see the authorizations assigned to a user:

```
# auths chris
solaris.admin.printer.read,solaris.admin.printer.modify,solaris.admin.printer.delete,solaris.device.cdrw,solaris.profmgr.read,solaris.jobs.users,solaris.mail.mailq,solaris.admin.usermgr.read,solaris.admin.logsvc.read,solaris.admin.fsmgr.read,solaris.admin.serialmgr.read,solaris.admin.diskmgr.read,solaris.admin.procmgr.user,solaris.compsys.read,solaris.admin.prodrdg.read,solaris.admin.dcmgr.read,solaris.snmp.read,solaris.project.read,solaris.admin.patchmgr.read,solaris.network.hosts.read,solaris.admin.volmgr.read
```

With this authorization, he can view or modify other user's crontab files:

```
# su - chris
Sun Microsystems Inc.      SunOS 5.10          s10_68  Sep. 20, 2004
$ crontab -l root
#ident  "@(#)root      1.21      04/03/23 SMI"
#
# The root crontab should be used to perform accounting data collection.
#
#
(output omitted)
$ exit
```

Assigning Authorizations to Roles

If a large number of user accounts require the same configuration and management of authorizations, it can be easier to assign the authorizations to a role and give the users access to the role.

The role can be created with the `roleadd` command or the Solaris Management Console (SMC). This example uses the `-P` and `-A` options of the `roleadd` command to create a role called `level2` with the rights profile `Mail Management` and the authorization `solaris.admin.user.*`.

```
# roleadd -m -d /export/home/level2 -P "Mail Management" \
-A "solaris.admin.usermgr.*" level2
64 blocks
# passwd level2
New Password: level2
Re-enter new Password: level2
passwd: password successfully changed for level2
```

```
# profiles level2
Mail Management
Basic Solaris User
All
# auths level2
solaris.admin.usermgr.*
(output omitted)
```

Assigning Authorizations to Rights Profiles

A rights profile usually includes a list of commands and special security attributes, the rights, as defined in the /etc/security/exec_attr file.

```
# grep '^Mail' /etc/security/exec_attr
Mail Management:suser:cmd:::/etc/init.d/sendmail:uid=0;gid=sys
Mail Management:suser:cmd:::/usr/lib/sendmail:uid=0
Mail Management:suser:cmd:::/usr/sbin/editmap:euid=0
Mail Management:suser:cmd:::/usr/sbin/makemap:euid=0
Mail Management:suser:cmd:::/usr/sbin/newaliases:euid=0
```

It is also possible to include predefined authorizations from the /etc/security/auth_attr file in the rights profile by adding the authorizations to the /etc/security/prof_attr file.

For example, the predefined Cron Management rights profile includes commands and authorizations. The /etc/security/prof_attr file defines the authorizations.

```
# grep '^Cron' /etc/security/prof_attr
Cron Management:::Manage at and cron
jobs:auths=solaris.jobs.*;help=RtCronMngmnt.html
```

The /etc/security/exec_attr defines the commands and special security attributes.

```
# grep '^Cron' /etc/security/exec_attr
Cron Management:suser:cmd:::/usr/bin/crontab:euid=0
```

The rights profile can then be given to a user:

```
# usermod -P "Cron Management" paul
```

Or a role:

```
# rolemod -P "Cron Management" level12
```

Make root User Into a Role

You can change root from a login user to a role. When you complete this procedure, you can no longer directly log in to the system as root, except in single-user mode. You must be assigned the root role and su to root.

By changing the root user into a role, you prevent anonymous root log-ins. Because a user must log in and then assume the root role, the user's login ID is provided to the auditing service and is in the sulog file.

Note – For safety, at least one local user should be assigned the root role.

You cannot perform this procedure when you are directly logged in as root. You must log in as yourself, then su to root.

- As a regular user, log in to the target system.
- Assume the Primary Administrator role, or become superuser.
- Create a local user who can assume the root role.

```
# useradd -c comments -u uid -d homedir username
```

-c comment The comment that describes the user

-u uid The user identification number

-d homedir The home directory of the user. This directory should be on the local system.

username The name of the new local user.

- Give the user a password.
- Make sure that you are not logged in as root.
- Change the root user into a role: **usermod -K type=role root**
- Verify that root is a role. The root entry in the **user_attr** file should be similar to following:

```
# grep root /etc/user_attr
root:::::type=role;auths=solaris.* ,solaris.grant;profile
s=...
```

- Assign the root role to your local account.

Caution – If you do not assign the root role to a user, no one can become superuser, except in single-user mode. You must type a root password to enter single-user mode.

- Configure the name service to return in case of failure.
 - Open a terminal window and assume the root role.
 - Edit the nsswitch.conf file.
- (Optional) Assign the root role to selected user accounts in the name service.

In a desktop environment, you cannot directly log in as root when root is a role. A diagnostic message indicates that root is a role on your system. If you do not have a local account that can assume the root role, create one. As root, log in to the system in single-user mode, create a local user account, and assign the root role to the new account. Then, log in as the new user and assume the root role.

No one can become superuser if you change the root user into a role and fail to make one of the following assignments:

- Assign the root role to a valid user.
- Assign a rights profile that is equivalent to root's rights profile to a valid user. The Primary Administrator profile is an equivalent rights profile for root capabilities.
- Create a role that has the capabilities of root and assign the role to a valid user. A role that is assigned the Primary Administrator profile is equivalent to the root role.

RBAC Configuration File Summary

The four files used by RBAC are interrelated. Figure 6-5 shows how these files are related.

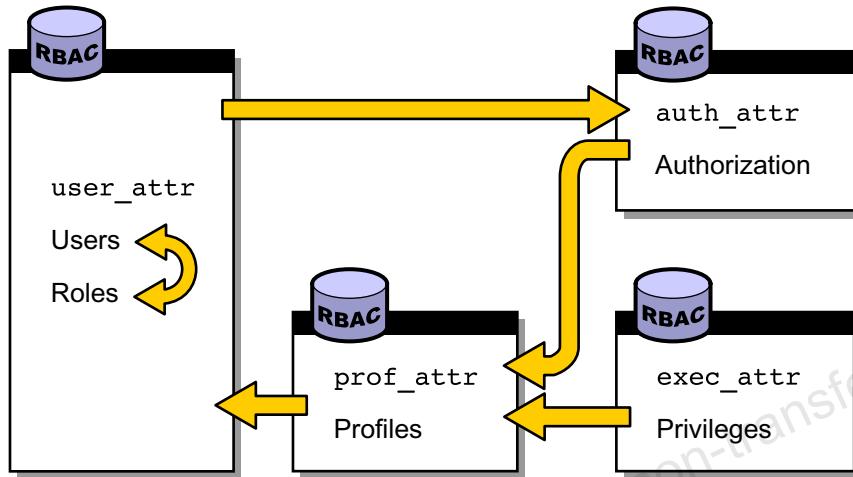


Figure 6-5 RBAC Files

The /etc/user_attr File

Figure 6-6 shows how the roles and users are associated within the file.

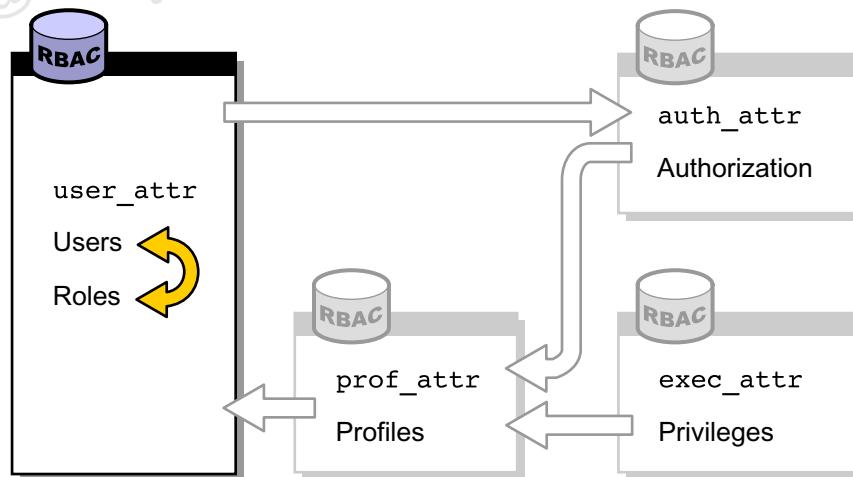


Figure 6-6 The /etc/user_attr File

Figure 6-7 shows a portion of a /etc/user_attr file. The user johndoe is a normal user account. The user is given the role of sysadmin. The sysadmin role is a role account. When assuming the sysadmin role, johndoe has access to specific rights profiles, defined as Device Management, Filesystem Management, and Printer Management profiles.

```
root::::type=normal;auth=solaris.*;solaris.grant
sysadmin::::type=role;profiles=Device Management,Filesystem
Management,Printer Management
johndoe::::type=normal;auths=solaris.system.date;roles=sysadmin
```

Figure 6-7 User and Role Association

The /etc/security/prof_attr File

The /etc/security/prof_attr file holds the rights profiles, as shown in Figure 6-8.

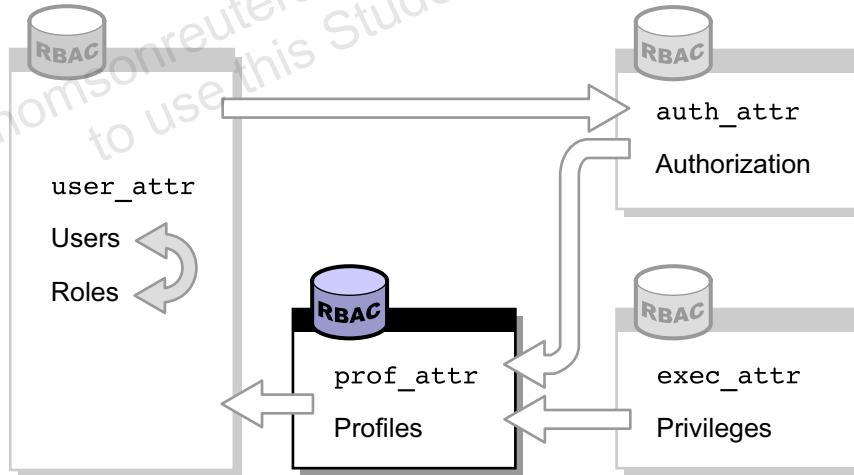


Figure 6-8 The prof_attr File

In the following example, the Printer Management rights profile is a supplementary rights profile that is assigned to the Operator rights profile and the System Administrator rights profile.

```
# grep 'Printer Management' /etc/security/prof_attr
Operator::::Can perform simple administrative tasks:profiles=Printer
Management,Media Backup,All;help=RtOperator.html
```

RBAC Configuration File Summary

Printer Management:::Manage printers, daemons,
spooling:help=RtPrntAdmin.html;auths=solaris.admin.printer.read,solaris.a
dmin.printer.modify,solaris.admin.printer.delete
System Administrator:::Can perform most non-security administrative
tasks:profiles=Audit Review,Printer Management,Cron Management,Device
Management,File System Management,Mail Management,Maintenance and
Repair,Media Backup,Media Restore,Name Service Management,Network
Management,Object Access Management,Process Management,Software
Installation,User Management,All;help=RtSysAdmin.html

Figure 6-9 shows one relationship between the
`/etc/security/prof_attr` and the `/etc/user_attr` files. The Printer
Management rights profile, which is defined in the
`/etc/security/prof_attr` file, is assigned to the sysadmin role in the
`/etc/user_attr` file.

From the `/etc/security/prof_attr` database:

```
Printer Management:::Manage printers, daemons,\  
spooling:help=RtPrntAdmin.html;auths=solaris.admin.printer.read,\  
solaris.admin.printer.modify,solaris.admin.printer.delete
```

From the `/etc/user_attr` database:

```
root::::type=normal;auth=solaris.* ,solaris.grant  
sysadmin::::type=role;profile=Device Management,Printer Management  
...
```

Figure 6-9 User and Profile Association

Figure 6-10 shows the relationship between the /etc/security/prof_attr and the /etc/security/auth_attr files. The Printer Management profile is defined in the /etc/security/prof_attr file as having all authorizations, beginning with the solaris.admin.printer. string, assigned to it. These authorizations are defined in the /etc/security/auth_attr file.

From the /etc/security/prof_attr database:

```
Printer Management:::Manage printers, daemons, spooling: \
help=RtPrntAdmin.html;authss=solaris.admin.printer.read, \
solaris.admin.printer.modify,solaris.admin.printer.delete
```

From the /etc/security/auth_attr database:

```
solaris.admin.printer.modify:::Update Printer Information: \
help=AuthPrinterModify.html
solaris.admin.printer.delete:::Delete Printer Information: \
help=AuthPrinterDelete.html
solaris.admin.printer:::Printer Information::help=AuthPrinterHeader.html
solaris.admin.printer.read:::View Printer Information: \
help=AuthPrinterRead.html
```

Figure 6-10 Profile and Authorization Association

The /etc/security/exec_attr File

Figure 6-11 shows the /etc/security/exec_attr file.

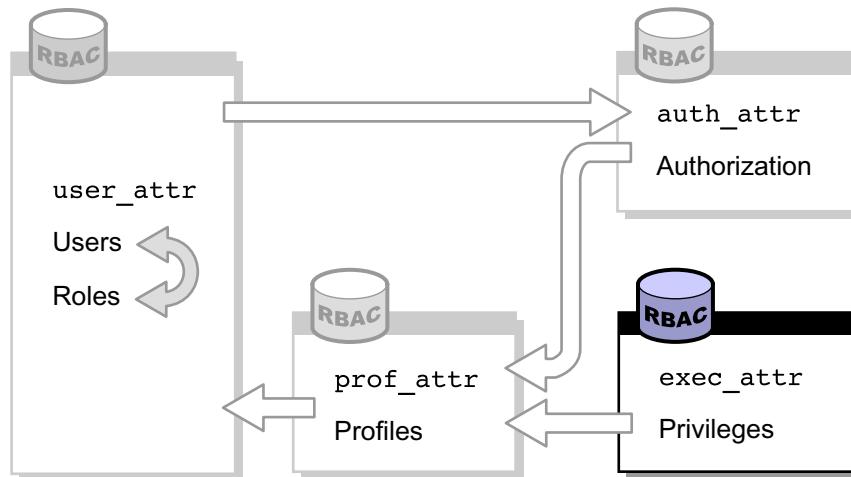


Figure 6-11 The exec_attr File

Figure 6-12 shows the relationship between the /etc/security/exec_attr and /etc/security/prof_attr files.

From the /etc/security/prof_attr database:

```
Printer Management:::Manage printers, daemons,  
spooling:help=RtPrntAdmin.html;auths=solaris.admin.printer.read,solaris.a  
dmin.printer.modify,solaris.admin.printer.delete
```

From the /etc/security/exec_attr database:

```
Printer Management:suser:cmd:::/usr/sbin/accept:euid=lp  
Printer Management:suser:cmd:::/usr/ucb/lpq:euid=0  
Printer Management:suser:cmd:::/etc/init.d/lp:euid=0  
Printer Management:suser:cmd:::/usr/bin/lpstat:euid=0  
Printer Management:suser:cmd:::/usr/lib/lp/lpsched:uid=0  
Printer Management:suser:cmd:::/usr/sbin/lpfILTER:euid=lp  
Printer Management:suser:cmd:::/usr/bin/lpset:egid=14  
Printer Management:suser:cmd:::/usr/sbin/lpadmin:egid=14  
Printer Management:suser:cmd:::/usr/sbin/lpsystem:uid=0  
Printer Management:suser:cmd:::/usr/sbin/lpmove:euid=lp  
Printer Management:suser:cmd:::/usr/sbin/lphut:euid=lp  
Printer Management:suser:cmd:::/usr/bin/cancel:euid=0  
Printer Management:suser:cmd:::/usr/bin/disable:euid=lp  
Printer Management:suser:cmd:::/usr/sbin/lpforms:euid=lp  
Printer Management:suser:cmd:::/usr/sbin/reject:euid=lp  
Printer Management:suser:cmd:::/usr/ucb/lprm:euid=0  
Printer Management:suser:cmd:::/usr/bin/enable:euid=lp  
Printer Management:suser:cmd:::/usr/sbin/lpusers:euid=lp
```

Figure 6-12 Profile and Execution Association

The Printer Management rights profile lists commands with the appropriate security attributes assigned in the /etc/security/exec_attr file.

The /etc/security/auth_attr File

Figure 6-13 shows the /etc/security/auth_attr file.

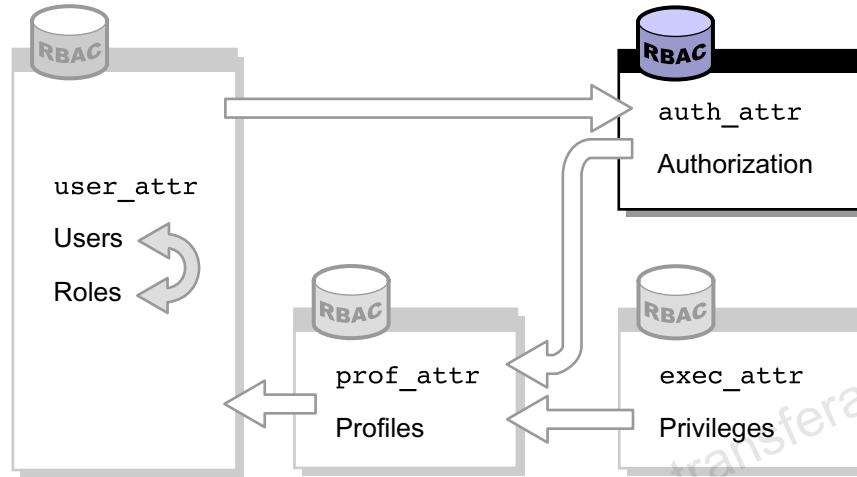


Figure 6-13 The auth_attr File

The following is an example of an /etc/security/auth_attr file, 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
```



Note – The solaris.device. entry is defined as a heading, because it ends in a dot (.). Headings are used by the GUI to organize families of authorizations.

RBAC Configuration File Summary

Figure 6-14 shows the relationship between the /etc/security/auth_attr and the /etc/user_attr files. The solaris.system.date authorization, which is defined in the /etc/security/auth_attr file, is assigned to the user johndoe in the /etc/user_attr file.

From the /etc/security/auth_attr database:

```
solaris.*:::Primary Administrator:::help=PriAdmin.html  
...  
solaris.system.date:::Set Date & Time:::help=SysDate.html  
...
```

From the /etc/user_attr database:

```
johndoe:::type=normal;auths=solaris.system.date;roles=sysadmin
```

Figure 6-14 User, Role, and Authorization Association

Figure 6-15 shows how the fields of the four files are related.

```
sysadmin:::type=role;profiles=Device Management,Filesystem  
Management,Printer Management,All  
  
johndoe:::type=normal;auths=solaris.system.date;roles=sysadmin
```

From the /etc/security/prof_attr database:

```
Printer Management:::Manage printers, daemons,  
spooling:::help=RtPrntAdmin.html;auths=solaris.admin.printer.read,solaris.a  
dmin.printer.modify,solaris.admin.printer.delete
```

From the /etc/security/exec_attr database:

```
Printer Management:suser:cmd:::/usr/sbin/accept:euid=lp  
Printer Management:suser:cmd:::/usr/ucb/lpq:euid=0  
Printer Management:suser:cmd:::/etc/init.d/lp:euid=0  
Printer Management:suser:cmd:::/usr/bin/lpstat:euid=0  
Printer Management:suser:cmd:::/usr/lib/lp/lpsched:uid=0
```

Figure 6-15 Relationship Among the Four RBAC Files

Exercise: Configuring RBAC

In this exercise, you configure RBAC by using the command line in the first task and by using the Solaris Management Console in the second task.

Preparation

Some steps in this exercise direct you to execute commands that do not work in order to demonstrate how RBAC must be used by logged in users.

Verify that the /export/home directory exists, and is writable by all users. If this directory does not exist, create it, and change its permission mode to 777.

Review how to use the auths, profiles, and roles RBAC commands to determine user privileges.

Tasks

The following section describes the tasks you must perform.

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Create a role named sdown. Give it a user ID of 5000 and a group ID of 10.
3. Create the profile named Shut by adding a line to the prof_attr file.
4. Add the Shut profile to the sdown role.
5. Verify that the sdown role is included in the /etc/user_attr file.
6. Create a user named user9 and assign it access to the sdown role. Give this user a user ID of 4009 and a group ID of 10.
7. Check the roles attributes for user9.
8. Assign the shutdown command to the Shut profile.

Exercise: Configuring RBAC

9. Use the `su` command to change user identity to `user9`.
10. As `user9`, without assuming the new role, attempt to shut down the system.

What is the result of this shutdown attempt, and why?

11. Execute the `profiles` command to determine which RBAC profiles are associated with `user9`.
12. Execute the `roles` command to determine which RBAC roles are associated with `user9`.
13. Assume the role `sdown`.
14. Shut down the system by using the `init` command.

What is the result of this shutdown attempt, and why?

15. List the commands that the `sdown` profile can execute.
16. Shut down the system using the `shutdown` command.

Note – Avoid shutting down the system by responding `n` to the prompt that the `shutdown` command presents.

What is the result of this shutdown attempt, and why?

17. Exit the `sdown` role.
18. Exit the shell for `user9`.



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 Solutions: Configuring RBAC

In this exercise, you configure RBAC by using the command line in the first task and by using the Solaris Management Console in the second task.

Preparation

Some steps in this exercise direct you to execute commands that do not work in order to demonstrate how RBAC must be used by logged in users.

Verify that the /export/home directory exists, and is writable by all users. If this directory does not exist, create it, and change its permission mode to 777.

Review how to use the auths, profiles, and roles RBAC commands to determine user privileges.

Tasks and Solutions

The following section describes the tasks you must perform, and provides the solutions to these tasks.

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Create a role named sdown. Give it a user ID of 5000 and a group ID of 10.

```
# roleadd -u 5000 -g 10 -m -d /export/home/sdown sdown
64 blocks
# passwd sdown
(output omitted)
```

3. Create the profile named Shut by adding a line to the prof_attr file.

```
# vi /etc/security/prof_attr
(output omitted)
```

Shut:::Able to shutdown the system:

4. Add the Shut profile to the sdown role.

```
# rolemod -P Shut sdown
```

- Verify that the sdown role is included in the /etc/user_attr file.

```
# more /etc/user_attr
```

(output omitted)

```
sdown::::type=role;profiles=Shut
```

- Create a user named user9 and assign it access to the sdown role. Give this user a user ID of 4009 and a group ID of 10.

```
# useradd -u 4009 -g 10 -m -d /export/home/user9 -s /bin/ksh \
-R sdown user9
```

64 blocks

```
# passwd user9
```

(output omitted)

- Check the roles attributes for user9.

```
# grep user9 /etc/user_attr
```

```
user9::::type=normal;roles=sdown
```

- Assign the shutdown command to the Shut profile.

```
# vi /etc/security/exec_attr
```

```
Shut:suser:cmd::::/usr/sbin/shutdown:uid=0
```

- Use the su command to change user identity to user9.

```
# su - user9
```

- As user9, without assuming the new role, attempt to shut down the system.

```
$ /usr/sbin/shutdown -i 6 -g 0
```

/usr/sbin/shutdown: Only root can run /usr/sbin/shutdown

What is the result of this shutdown attempt, and why?

This shutdown attempt fails because user9 has not assumed the sdown role yet, and as a regular user, does not have the rights profile to execute the shutdown command.

- Execute the profiles command to determine which RBAC profiles are associated with user9.

```
$ profiles
```

Basic Solaris User

All

- Execute the roles command to determine which RBAC roles are associated with user9.

```
$ roles
```

sdown

- Assume the role sdown.

Exercise Solutions: Configuring RBAC

```
$ su sdown
```

Password:

```
$
```

14. Shut down the system by using the `init` command.

```
$ /usr/sbin/init 0
```

Insufficient privileges.

x86/x64 systems display this message:

bootadm: you must be root to run this program

Depending on the shell in use, the message may also be "Must be super-user".

What is the result of this shutdown attempt, and why?

This shut down attempt fails because, even after assuming the sdown role, user9 does not have the execution attribute to execute the init command.

15. List the commands that the `sdown` profile can execute.

```
$ profiles -l
```

Shut:

`/usr/sbin/shutdown` uid=0

All:

 *

16. Shut down the system using the `shutdown` command.

```
$ /usr/sbin/shutdown -i 6 -g 0
```

Shutdown started. Sun Apr 29 17:30:10 MDT 2007

Do you want to continue? (y or n): **n**

What is the result of this shutdown attempt, and why?

This command succeeds because the sdown role has execute permission when issuing the shutdown command.

17. Exit the `sdown` role.

```
$ exit
```

```
$
```

18. Exit the shell for user9.

```
$ exit
```

Module 7

Configuring System Messaging

Objectives

The syslog system messaging facility manages the system logs. You can manually generate log messages by using the logger command. The Solaris Management Console allows the graphical viewing of logs including SMC activity. Regardless of the type of information you want to record, a messaging feature exists to record it.

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

- Describe the fundamentals of the syslog function
- Configure the /etc/syslog.conf file
- Configure syslog messaging
- Use the Solaris Management Console log viewer

Introducing the syslog Function

The syslog function, the syslogd daemon, and input from the /etc/syslog.conf file work together to facilitate system messaging for the Solaris 10 OS.

The syslog Concept

The syslog function sends messages generated by the kernel and system utilities and applications to the syslogd daemon, as shown in the Figure 7-1. With the syslog function you can control message logging, depending on the configuration of the /etc/syslog.conf file. The daemon can:

- Write messages to a system log
- Forward messages to a centralized log host
- Forward messages to a list of users
- Write messages to the system console

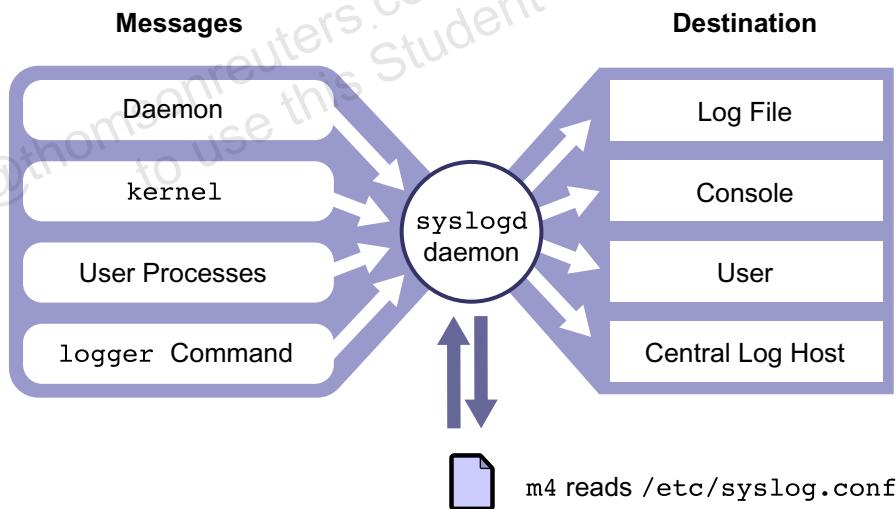


Figure 7-1 The syslog Structure

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 has two components, 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, when you place the following entry in the /etc/syslog.conf file, error messages for all facilities are sent to the /var/adm/messages file:

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

where:

| | |
|-------------------|--|
| *.err | Is the selector field. The asterisk (*) is the <i>facility</i> , and the dot (.) is the delimiter. The err field is the <i>level</i> of the message. |
| /var/adm/messages | Is the action field. |

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



The Solaris OS accesses the /usr/include/sys/syslog.h file to determine the correct *facility.level* sequencing order.

Selector Field

The selector field is a semicolon-separated list of priority specifications in the following format:

`facility.level;facility.level`

In the selector field syntax, `facility` is a system facility. Table 7-1 shows values that the selector field (`facility`) can contain.

Table 7-1 Selector Field (`facility`) Options

| Field | Description |
|----------|--|
| kern | Messages generated by the kernel. |
| user | Messages generated by user processes. This file does not list the default priority for messages from programs or facilities. |
| mail | The mail system. |
| daemon | System daemons, such as the <code>in.ftp</code> and the <code>telnetd</code> daemons. |
| auth | The authorization system, including the <code>login</code> , <code>su</code> , and <code>ttymon</code> commands. |
| syslog | Messages generated internally by the <code>syslogd</code> daemon. |
| lpr | The line printer spooling system, such as the <code>lpr</code> and <code>lpc</code> commands. |
| news | Files reserved for the USENET network news system. |
| uucp | The UNIX-to-UNIX copy (UUCP) system does not use the <code>syslog</code> function. |
| cron | The <code>cron</code> and <code>at</code> facilities, including <code>crontab</code> , <code>at</code> , and <code>cron</code> . |
| local0-7 | Fields reserved for local use. |
| mark | The time when the message was last saved. The messages are produced internally by the <code>syslogd</code> daemon. |
| * | All facilities, except the <code>mark</code> <i>facility</i> . |



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

In the selector field syntax, *level* is the severity or importance of the message. Each *level* includes all the levels above (of a higher severity). Table 7-2 shows the levels in descending order of severity.

Table 7-2 Selector Field (*level*) Options

| Level | Priority | Description |
|---------|----------|--|
| emerg | 0 | Panic conditions that are normally broadcast to all users |
| alert | 1 | Conditions that should be corrected immediately, such as a corrupted system database |
| crit | 2 | Warnings about critical conditions, such as hard device errors |
| err | 3 | Errors other than hard device errors |
| warning | 4 | Warning messages |
| notice | 5 | Non-error conditions that might require special handling |
| info | 6 | Informational messages |
| debug | 7 | Messages that are normally used only when debugging a program |
| none | 8 | Messages are not sent from the indicated <i>facility</i> 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 to forward the message. This field can have any one of the following entries:

| | |
|---------------------------|--|
| <code>/pathname</code> | Full path name to the targeted file. |
| <code>@host</code> | The @ sign denotes that messages must be forwarded to a remote host. Messages are forwarded to the <code>syslogd</code> daemon on the remote host. |
| <code>user1, user2</code> | The <code>user1</code> and <code>user2</code> entries receive messages if they are logged in. |
| <code>*</code> | All logged in users receive messages. |

Note – You must manually create the `/ pathname` full path and file name if it does not already exist.



Entries in the /etc/syslog.conf File

The standard /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.
#
# The 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/syslog, @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                      *
)
```

The `syslogd` Daemon and the `m4` Macro Processor

Figure 7-2 shows how the `syslogd` daemon, the `m4` macro processor, and the `/etc/syslog.conf` file interact in conceptual phases to determine the correct message routing.

Process

These conceptual phases are described as:

1. The `syslogd` daemon runs the `m4` macro processor.
2. The `m4` processor reads the `/etc/syslog.conf` file, processes any `m4` statements in the input, and passes the output to the `syslogd` daemon.
3. The `syslogd` daemon uses the configuration information output by the `m4` processor to route messages to the appropriate places.

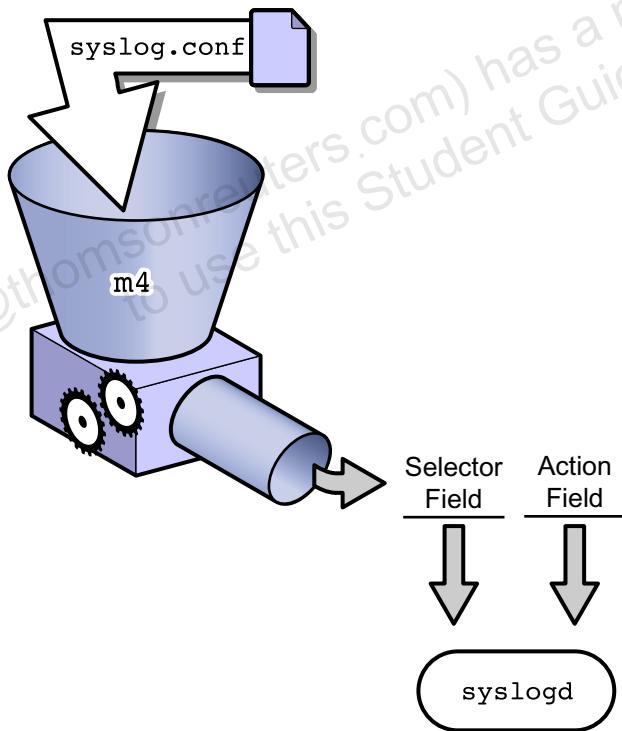


Figure 7-2 The `m4` Macro Processor

The syslogd daemon does not read the /etc/syslog.conf file directly. The syslogd daemon obtains its information as follows:

1. The syslogd daemon starts the m4 processor, which parses the /etc/syslog.conf file for m4 commands that it can interpret.
2. If the m4 processor does not recognize any m4 commands on a line, it passes the output back to the syslogd daemon as a two-column output.
3. The syslogd daemon then uses the two-column output to route messages to the appropriate destination.

If the m4 processor encounters an ifdef statement within the /etc/syslog.conf file, the ifdef statement is evaluated for a True or False condition. The message routing then occurs relative to the output of the test.

Operation Phase 1

In the following examples, the syslogd daemon is running on the host1 system. This section contains two examples of the host1 system's /etc/hosts file.

These /etc/hosts file examples are excerpts of the /etc/hosts/ file.

Example A /etc/hosts:

```
192.9.200.1 host1 loghost  
192.9.200.2 host2
```

Example B /etc/hosts:

```
192.9.200.1 host1  
192.9.200.2 host2 loghost
```

When the syslogd daemon starts at system boot, the syslogd daemon evaluates the /etc/hosts file, and checks the Internet Protocol (IP) address associated with the *hostname* as compared to the IP address associated with loghost.

In Example A, host1 and loghost are both associated with IP address 192.9.200.1. Therefore, the syslogd daemon runs the first command line: /usr/ccs/bin/m4 -D LOGHOST, causing the m4 LOGHOST variable to be defined 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. In this example, the syslogd daemon runs the second command line, /usr/ccs/bin/m4 (no -D LOGHOST), causing the m4 LOGHOST variable to be undefined during the parsing of the /etc/syslog.conf file.

Operation Phase 2

In the phase 2, the m4 macro processor parses the /etc/syslog.conf file. For each line that is parsed, the m4 processor searches the line for m4 statements, such as an ifdef statement. If no ifdef statement is encountered on the line, the m4 processor passes the line to the syslogd daemon.

If the m4 processor finds a line with an ifdef statement, the line is evaluated as follows:

- The ifdef ('LOGHOST', truefield, falsefield) command checks to see if the variable LOGHOST is defined.
- If the variable LOGHOST is defined, the entries from the truefield field are used; otherwise, entries from the falsefield field are used.

For example:

mail.debug

```
ifdef('LOGHOST', /var/log/syslog, @loghost)
```

If the variable LOGHOST variable is defined in phase 1, then the m4 processor returns:

mail.debug

```
/var/log/syslog
```

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. The m4 processor then passes the output to the syslogd daemon.

Operation Phase 3

For each line parsed in the `/etc/syslog.conf` file from phase 2, the `m4` processor produces output in a two-column field: a selector field and an action field. The output is sent to the `syslogd` daemon, which uses the information to route messages to their appropriate destinations. After the information is configured, the `syslogd` daemon continues to run with this configuration.

Configuring the /etc/syslog.conf File

The target locations for the syslog message files are defined within the /etc/syslog.conf file. You must restart the syslogd daemon whenever you make any changes to this file.

Message Routing

The following excerpt from the /etc/syslog.conf file shows how various events are logged by the system.

| | |
|---|-------------------|
| 1 *.err;kern.notice;auth.notice | /dev/sysmsg |
| 2 *.err;kern.debug;daemon.notice;mail.crit | /var/adm/messages |
| 3 *.alert;kern.err;daemon.err | operator |
| 4 *.alert | root |
| 5 *.emerg | * |

Note – Within the /etc/syslog.conf file, use a selector *level* of err to indicate that all events of priority error (and higher) are logged to the target defined in the action field.

In Line 1, every error event (*.err) and all kernel and authorization *facility* events of *level* notice, which are not error conditions but might require special handling, will write a message to the /dev/sysmsg file.

In Line 2, every error event (*.err), all kernel *facility* events of *level* debug, all daemon *facility* events of *level* notice, and all critical *level* mail events will record a message in the /var/adm/messages file. Therefore, errors are logged to both files.

Line 3 indicates that all alert *level* events, including the kernel error *level* and daemon error *level* events, are sent to the user operator if this user is logged in.

Line 4 indicates that all alert *level* events are sent to the root user if the root user is logged in.

Line 5 indicates that any event that the system interprets as an emergency will be logged to the terminal of every logged-in user.



To alter the event logging mechanism, edit the /etc/syslog.conf file, and restart the syslogd daemon.

Stopping and Starting the syslogd Daemon

The syslogd daemon can be started automatically during boot or manually from the command line.

Starting the syslogd Daemon During Boot Operation

The /lib svc/method/system-log file starts the syslogd process during each system boot.

The /etc/syslog.conf configuration file is read each time the syslogd daemon starts.

Manually Stopping and Starting the syslogd Daemon

If the configuration file has been modified, you can manually stop or start the syslogd daemon, or send it a refresh command, which causes the daemon to reread the /etc/syslog.conf file.

To stop the syslogd daemon, perform the command:

```
# svcadm disable svc:/system/system-log:default
```

To start the syslogd daemon, perform the command:

```
# svcadm enable svc:/system/system-log:default
```

To send a refresh to the syslogd daemon, perform the command:

```
# svcadm refresh svc:/system/system-log:default
```

Configuring syslog Messaging

The inetd daemon uses the `syslog` command to record incoming network connection requests made by using Transmission Control Protocol (TCP).

Enabling TCP Tracing

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

```
# inetadm -p
NAME=VALUE
bind_addr=""
bind_fail_max=-1
bind_fail_interval=-1
max_con_rate=-1
max_copies=-1
con_rate_offline=-1
failrate_cnt=40
failrate_interval=60
inherit_env=TRUE
tcp_trace=FALSE
tcp_wrappers=FALSE
```

Tracing for all services is enabled using the following command:

```
# inetadm -M tcp_trace=TRUE
# inetadm -p
NAME=VALUE
bind_addr=""
bind_fail_max=-1
bind_fail_interval=-1
max_con_rate=-1
max_copies=-1
con_rate_offline=-1
failrate_cnt=40
failrate_interval=60
inherit_env=TRUE
```

```
tcp_trace=TRUE
tcp_wrappers=FALSE
```

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

You can enable the trace option for each `inetd`-managed service to send messages to the `syslogd` daemon. Use the `inetadm` command to modify the settings of the service to enable TCP tracing. When you enable the trace option, it uses the `daemon.notice` to log the client's IP address and TCP port number, and the name of the service. To enable tracing TCP connections automatically, each service may have its trace capability enabled separately.

For example, to allow tracing of telnet sessions, the following command is issued:

```
# inetadm -m telnet tcp_trace=TRUE
# inetadm -l telnet
SCOPE      NAME=VALUE
           name="telnet"
           endpoint_type="stream"
           proto="tcp6"
           isrpc=FALSE
           wait=FALSE
           exec="/usr/sbin/in.telnetd"
           user="root"
default    bind_addr=""
default    bind_fail_max=-1
default    bind_fail_interval=-1
default    max_con_rate=-1
default    max_copies=-1
default    con_rate_offline=-1
default    failrate_cnt=40
default    failrate_interval=60
default    inherit_env=TRUE
           tcp_trace=TRUE
default    tcp_wrappers=FALSEgrep inetd /etc/init.d/inetsvc
```

 **Note** – The change is immediately recognized. There is no requirement to restart any daemon process.

Configuring syslog Messaging

The /etc/syslog.conf file configures the syslogd daemon so that it selectively distributes the messages sent to it from the inetd daemon.

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

All daemon messages of level notice or higher are sent to the /var/adm/messages file due to the daemon.notice entry in the /etc/syslog.conf file.

Note – The /var/adm/messages file must exist. If it does not exist, create it, and then stop and start the syslogd daemon, or messages will not be written to the file.



Monitoring a syslog File in Real Time

You can monitor the designated syslog file, in the /var/adm directory, in real time using the command tail -f. The tail -f command holds the file open so that you can view messages being written to the file by the syslogd daemon.

Viewing Messages In Real Time

To view messages sent to the /var/adm/messages file, perform the command:

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

Figure 7-3 shows the log entry generated by a telnet request to system host1 from IP address 192.9.200.1 on Port 45800. .

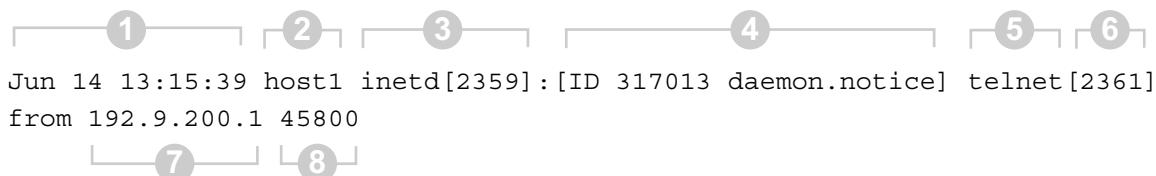


Figure 7-3 The syslogd Daemon Logged Entry

Table 7-3 lists each field in this figure and its corresponding result

Table 7-3 The syslogd Logged Entry Description

| Number | Field | Result |
|--------|---|---------------------------|
| 1 | Date/time | Jun 14 13:15:39 |
| 2 | Local host name | host1 |
| 3 | Process name/PID number | inetd[2359] |
| 4 | MsgID number/ selector <i>facility.level</i> | [ID 317013 daemon.notice] |
| 5 | Incoming request | telnet |
| 6 | PPID number | [2361] |
| 7 | IP address | 192.9.200.1 |
| 8 | Port number | 45800 |

To exit the /var/adm/messages file, press Control-C.

Note – Should any unusual activity occur, use scripts to automatically parse the log files, and then send the information to support personnel.



Adding One-Line Entries to a System Log File

The logger command enables you to send messages to the syslogd daemon. A system administrator can write administrative shell scripts that report the status of backups, or other functions by using the logger command.

The syntax of the logger command is:

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

where:

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

You can specify the message priority as a *facility.level* pair. For example, -p local3.info assigns the message priority of the info level in the local3 facility. The default priority is user.notice.

Therefore, the following example logs the message System rebooted to the syslogd daemon, using the default priority level notice and the facility user:

```
# logger System rebooted
```

If the user.notice selector field is configured in the /etc/syslog.conf file, the message is logged to the file designated for the user.notice selector field. If the user.notice selector field is not configured 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 routes the message to the /var/adm/messages file as indicated in the /etc/syslog.conf file.

A message priority can also be specified numerically. For example, logger -i -p 2 "crit" creates an entry in the message log that identifies the user.crit-facility.level pair as follows:

```
Nov 3 09:49:34 hostname root[2838]: [ID 702911 user.crit] crit
```

Exercise: Using the `syslog` Function and Auditing Utilities

In this lab, you use the `syslog` function to log messages locally and remotely.

Preparation

This exercise requires installed manual (man) pages and two systems that list each other in the `/etc/hosts` file. Verify that the `CONSOLE` variable is commented out in the `/etc/default/login` file on both systems. Except as noted otherwise, perform all steps on both systems. Refer to the lecture notes as necessary to perform the steps listed.

Tasks

Complete the following tasks.

Task 1 – Enabling and Logging `inetd` Trace Messages

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Change the directory to `/etc`, and create a backup copy of the `/etc/syslog.conf` file.
3. Display the man page for the `inetd` process, and verify the `facility` and `level` used by the `inetd` process when you run the process with the `-tcptrace` option.

Which `facility` and `level` pair is the `inetd` daemon using?

4. Examine the `/etc/syslog.conf` file, and determine if the `syslogd` daemon would recognize `inetd` tracing messages.

Are `inetd` tracing messages recognized by the `syslogd` daemon (yes or no)?

To what destination will the `syslogd` daemon send the messages?

5. Open a new terminal window, and use the `tail` command to view new entries as they are recorded in the `/var/adm/messages` file.
 6. In an available window, use the `telnet` command to connect to your own system. Exit the `telnet` session after you successfully log in.
 7. Observe the window in which you are running the `tail` command. Do any new `telnet`-related messages appear in the `/var/adm/messages` file (yes or no)?

 8. Modify the `inetd` service, and change the default value of the `tcp_trace` option to TRUE:
 9. Verify that the `inetd` daemon is running with the tracing option enabled.
 10. Repeat step 5 and step 6. Do any new `telnet`-related messages appear in the `/var/adm/messages` file? If yes, list them.
-
-

Task 2 – Using the `logger` Command to Demonstrate How Levels Operate

Complete the following steps:

1. Edit the `/etc/syslog.conf` file so that it includes the following line:
`local0.notice <TAB> /var/log/local0.log`
2. Create a file called `/var/log/local0.log`.
3. Cause the `syslogd` daemon to reread the `/etc/syslog.conf` file by sending it a `refresh` command.
4. In the window in which the `tail` command is running, stop the `tail` process. Restart the `tail` command so that it displays the end of the `/var/log/local0.log` file.

Exercise: Using the `syslog` Function and Auditing Utilities

5. In an available window, use the `logger` utility to send a message using the `local0 facility` and the `notice level`.

What, if any, new messages does the `tail` command display?

6. In an available window, use the `logger` command to send a message by using the `local0 facility` and the `crit level`.

What, if any, new messages does the `tail` command display?

7. Run the `logger` command from step 5 three times. Examine the output from the `tail` command in the other window. How many new messages appear in the `/var/log/local0.log` file?

8. Run the `logger` command with the `crit` level message instead of the `notice` level message.

Which new messages appear in the `/var/log/local0.log` file?

9. Stop the `tail` command in the window where it is running.

Task 3 – Logging Messages to Another System

Complete the following steps:



Note – These steps do not require you to change host names. In the following steps, substitute the appropriate host name for `system1` and `system2`.

1. On `system1`, edit the `/etc/syslog.conf` file, and change the line for `local0.notice` so that it reads as follows:

`local0.notice<TAB>@system2`

2. On `system1`, cause the `syslogd` daemon to reread the `/etc/syslog.conf` file using `svcadm`.
3. On system 2, create a file called `/var/log/local0.log` if it does not already exist.



Note – If you did not already edit the `/etc/syslog.conf` file on the system designated `system2` from the previous task, do so now. The `/etc/syslog.conf` file on `system2` must direct `local0.notice` messages to `/var/log/local0.log`. You must also refresh the `system/system-log` service on `system2`.

4. On `system2`, open a new terminal window, and use the `tail` command to view new entries as they arrive in the `/var/log/local0.log` file.
5. On `system1`, use the `logger` command to generate a message by using the `local0.notice` *facility* and *level* pair.
6. On `system2`, which message is displayed in the window running the `tail` command?

-
-
7. After verifying that `system1` has successfully passed messages to `system2`, stop the `tail` command on `system2`.

Task 4 – Logging Messages by Using the `loghost` Alias and `ifdef` Statements

Complete the following steps:

Exercise: Using the `syslog` Function and Auditing Utilities

1. On both systems, edit the `/etc/syslog.conf` file, and uncomment the line that identifies `auth.notice` messages.

`auth.notice`

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

Which two destinations are possible for these messages?

2. On both systems, examine the `/etc/inet/hosts` file, and identify the name of the host associated with the `loghost` alias.
3. On both systems, cause the `syslogd` daemon to reread the `/etc/syslog.conf` file by sending it a refresh command.
4. On both systems, run the following `m4` commands, and record the line for `auth.notice` messages.

```
# /usr/ccs/bin/m4 -D LOGHOST /etc/syslog.conf
```

```
# /usr/ccs/bin/m4 /etc/syslog.conf
```

5. On both systems, open a terminal window, and use the `tail` command to view new entries as they arrive in the `/var/log/authlog` file.
6. On `system2`, use the `rlogin` command to log in to your own system, and then exit the connection.

On `system2`, which message is displayed in the window running the `tail` command?

On `system1`, does a new message display in the window running the `tail` command (yes or no)?

7. On `system2`, change to the `/etc/inet` directory, and make a backup copy of the `/etc/inet/hosts` file. Edit the `/etc/inet/hosts` file to remove the `loghost` alias from the entry for `system2`, and add it to the entry for `system1`.
8. On `system2`, force the `syslogd` daemon to reread the `/etc/syslog.conf` file using `svcadm`.

9. On `system2`, use the `rlogin` command to log in to your own system, and then exit the connection.

On `system2`, does a new message display in the window running the `tail` command (yes or no)?

On `system1`, which message is displayed in the window running the `tail` command?

Task 5 – Completing the Exercise

Complete the following steps:

1. On both systems, stop the `tail` command in any window where it is running.
2. On `system2`, replace the `/etc/inet/hosts` file with the backup copy you made earlier.
3. On both systems, replace the `/etc/syslog.conf` file with the backup copy you made earlier.
4. On both systems, ensure that the `syslogd` daemon rereads the `/etc/syslog.conf` file by sending it a `refresh` command.

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 Solutions: Using the `syslog` Function and Auditing Utilities

In this lab, you use the `syslog` function to log messages locally and remotely.

Preparation

This exercise requires installed manual (man) pages and two systems that list each other in the `/etc/hosts` file. Verify that the `CONSOLE` variable is commented out in the `/etc/default/login` file on both systems. Except as noted otherwise, perform all steps on both systems. Refer to the lecture notes as necessary to perform the steps listed.

Tasks and Solutions

The following section lists the tasks you must perform and the solutions to these tasks.

Task 1 – Enabling and Logging `inetd` Trace Messages

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
1. Change the directory to `/etc`, and create a backup copy of the `/etc/syslog.conf` file.

```
# cd /etc  
# cp syslog.conf syslog.conf.bak
```

2. Display the man page for the `inetd` process, and verify the *facility* and *level* used by the `inetd` process when you run the process with the `-tcptrace` option.

```
# man inetd
```

Which *facility* and *level* pair is the `inetd` daemon using?
`daemon.notice`

3. Examine the /etc/syslog.conf file, and determine if the syslogd daemon would recognize inetd tracing messages.

Are inetd tracing messages recognized by the syslogd daemon (yes or no)?

Yes

To what destination will the syslogd daemon send the messages?

The /var/adm/messages file.

4. Open a new terminal window, and use the tail command to view new entries as they are recorded in the /var/adm/messages file.

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

5. In an available window, use the telnet command to connect to your own system. Exit the telnet session after you successfully log in.

```
# telnet host
```

Trying nnn.nnn.nnn.nnn...

Connected to host.

Escape character is '^]'.

login: root

Password:

(output omitted)

```
# exit
```

6. Observe the window in which you are running the tail command. Do any new telnet-related messages appear in the /var/adm/messages file (yes or no)?

Before starting the inetd service with telnet tracing, no.

7. Modify the inetd service, and change the default value of the tcp_trace option to TRUE:

```
# inetadm -M tcp_trace=TRUE
```

8. Verify that the `inetd` daemon is running with the tracing option enabled.

```
# inetadm -p
NAME=VALUE
bind_addr=""
bind_fail_max=-1
bind_fail_interval=-1
max_con_rate=-1
max_copies=-1
con_rate_offline=-1
failrate_cnt=40
failrate_interval=60
inherit_env=TRUE
tcp_trace=TRUE
tcp_wrappers=FALSE
```

9. Repeat step 5 and step 6. Do any new telnet-related messages appear in the `/var/adm/messages` file? If yes, list them.

A message similar to the following message appears:

```
Nov  6 14:19:21 sys-02 inetd[224]: [ID 317013 daemon.notice] telnet[1181]
from 192.168.201.21 32795
```

Task 2 – Using the `logger` Command to Demonstrate How Levels Operate

Complete the following steps:

1. Edit the `/etc/syslog.conf` file so that it includes the following line:
`local0.notice <TAB> /var/log/local0.log`
2. Create a file called `/var/log/local0.log`.

```
# touch /var/log/local0.log
```

3. Cause the `syslogd` daemon to reread the `/etc/syslog.conf` file by sending it a `refresh` command.

```
# svcadm refresh svc:/system/system-log:default
```

4. In the window in which the `tail` command is running, stop the `tail` process. Restart the `tail` command so that it displays the end of the `/var/log/local0.log` file.

```
# tail -f /var/log/local0.log
```

5. In an available window, use the logger utility to send a message using the local0 facility and the notice level.

```
# logger -p local0.notice Notice-level message
```

What, if any, new messages does the tail command display?

A message similar to the following appears:

```
Nov 04 15:21:49 host root: [ID 702911 local0.notice] Notice-level message
```

6. In an available window, use the logger command to send a message by using the local0 facility and the crit level.

```
# logger -p local0.crit Crit-level message
```

What, if any, new messages does the tail command display?

```
Nov 04 15:24:43 host1 root: [ID 702911 local0.crit] Crit-level message
```

A message similar to this displays because crit is a higher level than notice, and the syslogd daemon is configured to recognize the notice level and higher for the local0 facility.

7. Run the logger command from step 5 three times. Examine the output from the tail command in the other window. How many new messages appear in the /var/log/local0.log file?

One. The syslogd daemon will not report multiple instances of the same message until a different message is logged, or the syslogd “mark” interval is reached.

8. Run the logger command with the crit level message instead of the notice level message.

Which new messages appear in the /var/log/local0.log file?

A message indicating that the previous message was repeated a number of times, and the new message, for example:

```
Oct  8 18:33:33 host last message repeated 2 times
```

```
Oct  8 18:34:19 host root: [ID 702911 local0.crit] Crit-level message
```

9. Stop the tail command in the window where it is running.

Task 3 – Logging Messages to Another System

Complete the following steps:



Note – These steps do not require you to change host names. In the following steps, substitute the appropriate host name for `system1` and `system2`.

1. On `system1`, edit the `/etc/syslog.conf` file, and change the line for `local0.notice` so that it reads as follows:

```
local0.notice<TAB>@system2
```

2. On `system1`, cause the `syslogd` daemon to reread the `/etc/syslog.conf` file using `svcadm`.

```
# svcadm refresh system/system-log
```

3. On system 2, create a file called `/var/log/local0.log` if it does not already exist.

```
# touch /var/log/local0.log
```



Note – If you did not already edit the `/etc/syslog.conf` file on the system designated `system2` from the previous task, do so now. The `/etc/syslog.conf` file on `system2` must direct `local0.notice` messages to `/var/log/local0.log`. You must also refresh the `system/system-log` service on `system2`.

4. On `system2`, open a new terminal window, and use the `tail` command to view new entries as they arrive in the `/var/log/local0.log` file.

```
# tail -f /var/log/local0.log
```

5. On `system1`, use the `logger` command to generate a message by using the `local0.notice` *facility* and *level* pair.

```
# logger -p local0.notice Message from system1
```

6. On `system2`, which message is displayed in the window running the `tail` command?

A message similar to the following:

```
Nov 06 13:07:49 system1 root: [ID 702911 local0.notice] Message from system1
```

7. After verifying that `system1` has successfully passed messages to `system2`, stop the `tail` command on `system2`.

Task 4 – Logging Messages by Using the loghost Alias and ifdef Statements

Complete the following steps:

1. On both systems, edit the /etc/syslog.conf file, and uncomment the line that identifies auth.notice messages.

auth.notice `ifdef('LOGHOST', /var/log/authlog, @loghost)`

Which two destinations are possible for these messages?

/var/log/authlog – This local host's log file

@loghost – The syslog facility on the "loghost"

2. On both systems, examine the /etc/inet/hosts file, and identify the name of the host associated with the loghost alias.

In the default /etc/inet/hosts file, the loghost alias is associated with the host name of the local system.

3. On both systems, cause the syslogd daemon to reread the /etc/syslog.conf file by sending it a refresh command.

svcadm refresh system/system-log

4. On both systems, run the following m4 commands, and record the line for auth.notice messages.

/usr/ccs/bin/m4 -D LOGHOST /etc/syslog.conf

auth.notice /var/log/authlog

/usr/ccs/bin/m4 /etc/syslog.conf

auth.notice @loghost

5. On both systems, open a terminal window, and use the tail command to view new entries as they arrive in the /var/log/authlog file.

tail -f /var/log/authlog

6. On `system2`, use the `rlogin` command to log in to your own system, and then exit the connection.

```
# rlogin system2
Password: xxxxxx
...
# exit
```

On `system2`, which message is displayed in the window running the `tail` command?

A message similar to the following displays:

```
Mar 31 09:15:23 system2 login: [ID 254462 auth.notice] ROOT LOGIN
/dev/pts/7 FROM system2
```

On `system1`, does a new message display in the window running the `tail` command (yes or no)?

No.

7. On `system2`, change to the `/etc/inet` directory, and make a backup copy of the `/etc/inet/hosts` file. Edit the `/etc/inet/hosts` file to remove the `loghost` alias from the entry for `system2`, and add it to the entry for `system1`.

```
# cd /etc/inet
# cp hosts hosts.bak
# vi hosts
```

8. On `system2`, force the `syslogd` daemon to reread the `/etc/syslog.conf` file using `svcadm`.

```
# svcadm refresh system/system-log
```

9. On `system2`, use the `rlogin` command to log in to your own system, and then exit the connection.

```
# rlogin system2
Password: xxxxxx
...
# exit
```

On `system2`, does a new message display in the window running the `tail` command (yes or no)?

No.

On `system1`, which message is displayed in the window running the `tail` command?

A message similar to the following displays:

```
Nov 06 09:34:46 system2 login: [ID 254462 auth.notice] ROOT LOGIN
/dev/pts/7 FROM system2
```

Task 5 – Completing the Exercise

Complete the following steps:

1. On both systems, stop the tail command in any window where it is running.
2. On *system2*, replace the /etc/inet/hosts file with the backup copy you made earlier.
3. On both systems, replace the /etc/syslog.conf file with the backup copy you made earlier.
4. On both systems, ensure that the syslogd daemon rereads the /etc/syslog.conf file by sending it a refresh command.

```
# svcadm refresh system/system-log
```

Module 8

Using Name Services

Objectives

Name services centralize shared information on a network. There are several services that store and provide access to this information.

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

- Describe the name services concept
- Describe the name services switch file /etc/nsswitch.conf
- Describe the name services cache daemon (nscd)
- Get name services information

Introducing the Name Service Concept

The original text-file based UNIX® name service was developed for standalone UNIX systems and was then adapted for network use. While UNIX operating systems still support and use this text-based name service, it is not appropriate for large, complex networks. The name service concept uses domains, which are defined as a collection of network nodes.

The concept of a name service centralizes the shared information in a network. A single system, the name server, maintains the information previously maintained on each individual host. The name servers provide information such as host names, Internet Protocol (IP) addresses, user names, passwords, and automount maps.

Note – Clients may still require local text files, for example the /etc/inet/hosts file, to configure the network interface.

Other hosts in the name service domain (called *clients*), request the information from the name server. This name server system responds to clients, and translates, or resolves their requests from its memory-based (cached) or disk-based databases.

Figure 8-1 shows one possible name service scenario. Later, this module describes alternatives to this scenario.

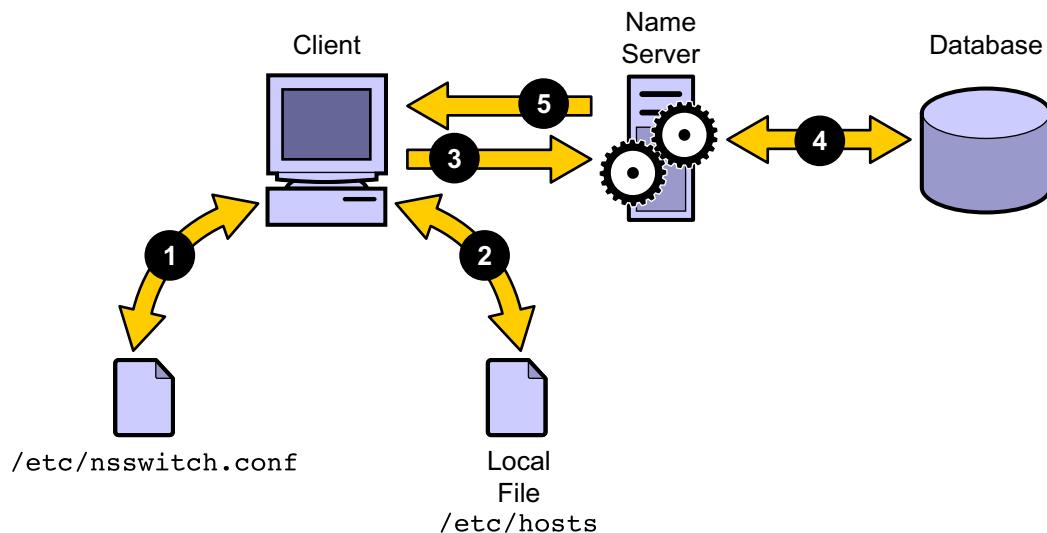


Figure 8-1 Name Service Scenario

The basic process is as follows:

1. The client requires administrative data to be accessed due to some process request. The client references its local name service switch file to determine the possible name service sources to search.
2. The name service switch file instructs the client to first search the local file for the information.
3. When the information is not located in the local files, the client's name service switch file redirects the search to a network name server.
4. The name server searches its database and locates the information.
5. The name server returns the information to its requesting client.

The name service concept provides the following benefits:

- A single point of administration for name service data
- Consistent name service information for systems within the domain
- All clients have access to changed data
- Assurance that clients do not miss updates

In a file-based scheme, updates distributed by using File Transfer Protocol (FTP) could be missed if a host was 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 server's database, receive changes and updates to the database from the master, and participate in client query resolution. Therefore, they not only overcome a single point-of-failure, but they also play a role in improved name service performance by balancing the workload of answering client requests among multiple systems.

Domain Name System (DNS)

Domain Name System (DNS) is an Internet-wide naming system for resolving host names to IP addresses and IP addresses to host names. DNS supports name resolution for both local and remote hosts, and uses the concept of domains to allow hosts with the same name to coexist on the Internet, so long as they are in different domains. For example:

`www.sun.com` and `www.microsoft.com`

The collection of networked systems that use DNS is referred to as the DNS *namespace*. The DNS namespace is divided into a hierarchy of domains. A DNS domain is a group of systems. Each domain is usually supported by two or more name servers, a master name server, and one or more slave name servers. Each server implements DNS by running the `in.named` daemon. On the client's side, DNS is implemented through the *resolver*. The resolver library resolves users' queries. The resolver queries a name server, which then returns either the requested information or a referral to another DNS server.

Figure 8-2 shows that the DNS namespace for the Internet begins with the nameless root domain and includes all subdomains, each of which is headed by a top-level domain.

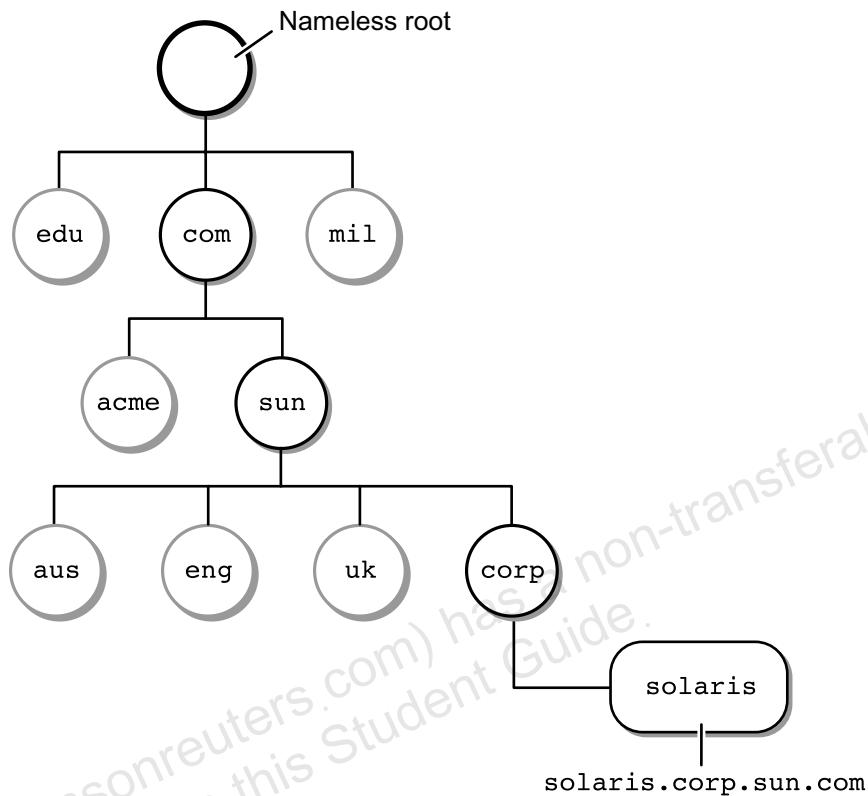


Figure 8-2 DNS Domain Structure

The top-level domains are administered by various organizations, all of which report to the governing authority called the Internet Corporation for Assigned Names and Numbers (ICANN). Administration of the lower-level domains is delegated to the various organizations that are registered domain name members within the top-level domain.

The top-level domain that 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.

Everything below the connection to the domain falls into a zone of authority maintained by the connection to the domain. For example, everything below `sun.com` resides within the zone of authority for Sun Microsystems, Inc. and is, therefore, maintained by Sun Microsystems, Inc.



The DNS name servers store the host and IP address information in files called *zone files*. The `svc:/network/dns/server:default` service starts the DNS server during the boot process if the DNS server has been configured.

Note – Setting up a DNS server is covered in SA-300-S10, *Network Administration for the Solaris 10 OS*.

Network Information Service (NIS)

Network Information Service (NIS) was developed independently of DNS and has a slightly different focus. DNS focuses on making communication easier by using host names instead of numerical IP addresses. NIS focuses on making network administration more manageable by providing centralized control over a variety of network information. NIS stores information about host names, IP addresses, users, groups, and others. This collection of network information is referred to as the NIS namespace.

NIS namespace information is stored in files called NIS *maps*. NIS maps were designed to supplement many of the UNIX `/etc` files. These maps store much 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 in a directory that you specify). By default, these maps are stored in the `/var/yp/domainname` directory on NIS servers. For example, the set of maps that contain hosts information include:

- `hosts.byaddr`
- `hostsbyname`

Note – You can obtain a list of the full set of maps from an NIS-configured system by running the `ypwhich -m` command.



NIS uses domains to define who can access the host names, user information, and other administrative data in its namespace. However, NIS does not use a domain hierarchy to store its data; therefore, the NIS namespace is flat.

You cannot look up addresses on the Internet by using just NIS. However, organizations that want to use NIS and also want to look up addresses on the Internet can combine NIS with DNS. You can use NIS to manage all local information and use DNS for Internet host lookup. The Solaris OS also allows you to set up the /etc/nsswitch.conf file so that lookup requests for hosts do the following:

- Query DNS
- Query DNS and then NIS, if the requests are not found by DNS
- Query NIS and then DNS, if the requests are not found by NIS

NIS uses a client-server arrangement similar to DNS. Replicated NIS servers provide services to NIS clients. The principal server is called a master server, and, for reliability, it has a backup, or a slave server. Both master and slave servers use the NIS information retrieval software and both store NIS maps.

Each server implements NIS by running the `ypserv` daemon. All NIS clients and servers must run the `ypbind` daemon to exchange NIS information. The `svc:/network/nis/server:default` service starts the NIS server during the boot process. NIS processes are only started if the NIS server has been configured.

The `svc:/network/nis/client:default` service starts the NIS client during the boot process.

Network Information Service Plus (NIS+)

Network Information Service Plus (NIS+) is similar to NIS but provides many more features. NIS+ is not an extension of NIS. NIS+ is a different software program.



Note – NIS+ is a mature naming service. Sun's customers have indicated a preference for using IETF standards for naming services based on Lightweight Directory Access Protocol (LDAP). Sun is indicating formally that there are plans for NIS+ to be removed sometime after the Solaris 10 OS release, however, removal will not occur in the next release of the Solaris OS.

You can configure the NIS+ name service to match the requirements of the organization using it. NIS+ enables you to store information about machine addresses, security information, mail information, Ethernet interfaces, and network services in central locations where all machines on a network can have access to the information. This configuration of network information is referred to as the NIS+ namespace.

The NIS+ namespace is hierarchical and is similar in structure to the UNIX directory tree. The hierarchical structure allows an NIS+ namespace to be configured to conform to the logical hierarchy of an organization. The namespace's layout of information is unrelated to its physical arrangement. Therefore, an NIS+ namespace can be divided into multiple domains that can be administered independently. Clients might have access to information in other domains in addition to their own if they have the appropriate permissions.

NIS+ uses a client-server model to store and gain access to the information contained in an NIS+ namespace. Each domain is supported by a set of servers. The principal server is called the root server, and the backup servers are called replica servers. The network information is stored in NIS+ tables in an internal NIS+ database. Both root and replica servers run NIS+ server software as well as maintain copies of NIS+ tables. Unlike NIS, the NIS+ namespace is dynamic because updates can occur and be put into effect at any time by any authorized user. Changes made to the NIS+ data on the root server are automatically and incrementally propagated to the replica servers.

NIS+ includes a sophisticated security system to protect the structure of the namespace and its information. NIS+ uses authentication and authorization to verify whether a client's request for information should be fulfilled. Authentication determines whether the information requester is a valid user on the network. Authorization determines whether a particular user is allowed to have or to modify the information requested.

Each server implements NIS+ by running the `rpc.nisd` daemon. NIS+ clients and servers run the `nis_cachemgr` daemon to enhance data access performance. The `svc:/network/rpc/nisplus:default` service starts the NIS+ name service during the boot process. NIS+ processes are only started if a NIS+ server has been configured and enabled with the `svcadm enable svc:/network/rpc/nisplus:default` command.

Lightweight Directory Access Protocol (LDAP)

LDAP is the protocol clients use to communicate with a directory server. It is a vendor independent protocol and can be used on common TCP/IP networks.

LDAP Directory Server

A directory server is not necessarily an LDAP server. However, in the context of this module, the term *Directory Server* is synonymous with *LDAP Server*. The Solaris 10 Operating System comes with an LDAP client and LDAP server, however, these products are not installed by default. The LDAP Directory Server is called the Sun Java™ System Directory Server.

The Sun Java System Directory Server must be set up and then configured to support Solaris LDAP clients.

Directory Entries

A directory server stores information in a Directory Information Tree (DIT). Clients can query the directory server for information or make changes to the information stored on the server.

The hierarchy of the directory tree structure is similar to that of the UNIX file system. Entries are named according to their position in this tree structure by a distinguished name (DN). The DN is similar to an absolute path name in UNIX. A Relative Distinguished Name (RDN) is similar to a relative path name in UNIX. As in the UNIX file system, sibling directory entries must have unique RDNs.

A directory entry is composed of attributes that have a type and one or more values. The syntax for each attribute defines the allowed values, or the allowed data type of the attribute values, such as American Standard Code for Information Interchange (ASCII) characters or a numerical data. LDAP also defines how those values are interpreted during a directory operation, for example, determining if a search or compare is case sensitive.

Like the DNS namespace, LDAP directory entry names (or DNs) start (from left) with the least significant component and proceed to the most significant; in other words, those just below root. The DN is constructed by concatenating the sequence of components up to the root of the tree.

Figure 8-3 shows an example of a Solaris LDAP Directory Information Tree.

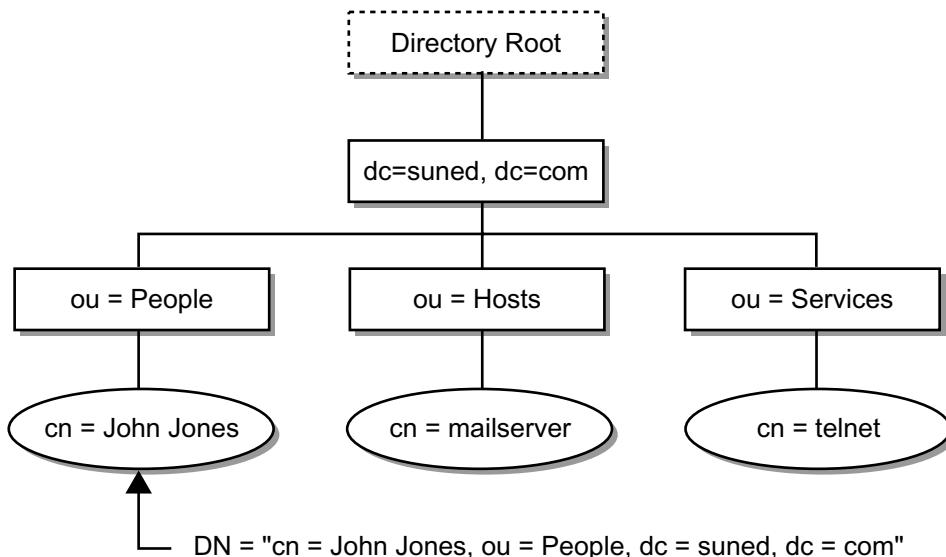


Figure 8-3 Solaris LDAP Directory Information Tree

Name Service Features Summary

Table 8-1 lists and compares the name services available in the Solaris OS.

Table 8-1 Name Service Features

| Feature | DNS | NIS | NIS+ | LDAP |
|---------------------|--------------------------------------|--------------------------|-------------------------------------|----------------------|
| Namespace | Hierarchical | Flat | Hierarchical | Hierarchical |
| Data storage | Files/resource records | Two column maps | Multicolumn tables | Directories (varied) |
| Server types | Master/slave/caching only/forwarding | Master/slave | Root master/non-root master/replica | Master/replica |
| Transport | IP | IP | IP | IP |
| Scale | Wide area network (WAN) | Local area network (LAN) | LAN | WAN |

Introducing the Name Service Switch File

The name service switch file determines which name services a system uses to search for information, and in which order the name service request is resolved. All Solaris OS systems use the /etc/nsswitch.conf file as the name service switch file. The nsswitch.conf file is loaded with the contents of a template file during the installation of the Solaris OS, depending on the name service that is selected, as shown in Table 8-2.

Table 8-2 Name Service Template Files

| Name Service | Name Service Template |
|--------------|-----------------------|
| Local files | /etc/nsswitch.files |
| DNS | /etc/nsswitch.dns |
| NIS | /etc/nsswitch.nis |
| NIS+ | /etc/nsswitch.nisplus |
| LDAP | /etc/nsswitch.ldap |

The following example is the /etc/nsswitch.conf file configured to support the NIS name service using the /etc/nsswitch.nis template.

```
#  
# /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.  
  
# NIS service requires that svc:/network/nis/client:default be enabled  
# and online.  
  
# 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
```

```
# Note that IPv4 addresses are searched for in all of the ipnodes
# databases before searching the hosts databases.
ipnodes:      files

#ipnodes:      nis [NOTFOUND=return] files

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
project:     files nis
```

The /etc/nsswitch.conf file includes a list of databases that are sources of information about IP addresses, users, and groups. Data for these can come from a variety of sources. For example, host names and host addresses, are located in the /etc/inet/hosts file, NIS, NIS+, LDAP, or DNS. Each database has zero or more sources; the sources and their lookup order are specified in the /etc/nsswitch.conf file.

Database Sources

There is an entry in the `/etc/nsswitch.conf` file for each database. Some typical examples of these entries are:

- `ipnodes: files`
- `passwd: files nis`
- `hosts: nis [NOTFOUND=return] files`

The information sources are listed in the order that they are searched, and these sources are defined in Table 8-3.

Table 8-3 Information Sources

| Information Sources | Description |
|----------------------|---|
| <code>files</code> | Specifies that entries be obtained from a file stored in the client's <code>/etc</code> directory. For example, <code>/etc/hosts</code> . |
| <code>nisplus</code> | Specifies that entries be obtained from an NIS+ table. For example, the <code>hosts</code> table. |
| <code>nis</code> | Specifies that entries be obtained from an NIS map. For example, the <code>hosts</code> map. |
| <code>dns</code> | Specifies that host information be obtained from DNS. |
| <code>ldap</code> | Specifies that entries be obtained from the LDAP directory. |
| <code>user</code> | Specifies that printer information be obtained from the <code> \${HOME} / .printers</code> file |

There might be a single information source listed, in which case the search terminates if the information is not found. If two or more sources are listed, the first listed source is searched before moving on to the next listed source. The relationships between these name service keywords, when found in the `nsswitch.conf` file, is further explained in Table 8-4 on page 8-15 and Table 8-5 on page 8-15.

Status Codes

When multiple information sources are specified, it is sometimes necessary to define precisely the circumstances under which each source is searched. When a name service is referenced, the attempt to search this source can return one of the following status codes, as shown in Table 8-4.

Table 8-4 Status Message Codes

| Status Message | Meaning of Message |
|----------------|--|
| SUCCESS | The requested entry was found in the specified source. |
| UNAVAIL | The source is not configured on this system and cannot be used. In other words, the NIS or NIS+ processes could not be found or contacted. |
| NOTFOUND | The source responded with No such entry. In other words, the table, map, or file was accessed, but it did not contain the needed information. |
| TRYAGAIN | The source is busy. It might respond if tried again. In other words, the name service is running and was contacted but could not service the request at that moment. |

Actions

For each status code, two actions are possible, as shown in Table 8-5.

Table 8-5 Status Code Actions

| Action | Meaning of Action |
|----------|---------------------------------------|
| return | Stop looking for the information. |
| continue | Try the next source, if there is one. |

Introducing the Name Service Switch File

When the action is not explicitly specified, the default action is to continue the search using the next specified information source, as follows:

- SUCCESS = return
- UNAVAIL = continue
- NOTFOUND = continue
- TRYAGAIN = continue

For example:

```
ipnodes: files
```

In this example, the /etc/inet/ipnodes file is searched for the first entry that matches the requested host name. If no matches are found, an appropriate error is returned, and no further information sources are searched.

Another example:

```
passwd: files nis
```

In this example, the appropriate files in the /etc directory are searched for the corresponding password entry. If the entry is not found, the NIS maps are searched for the entry. If no entry is found in the NIS maps, an appropriate error is returned, and no further information sources are searched.

Another example:

```
hosts: nis [NOTFOUND=return] files
```

In this example, the NIS maps are searched for the entry. If the source (NIS) is not running, the system returns the status UNAVAIL, and continues to search the /etc/inet/hosts file. If the entry returns the status NOTFOUND, an appropriate error is returned, and the search is terminated without searching the /etc/inet/hosts file.

Configuring the Name Service Cache Daemon (nscd)

To properly use the name service cache daemon (nscd), you must be able to perform the following:

- Describe the purpose of the name service cache daemon
- Configure the name service cache daemon
- Stop and start the name service cache daemon

The nscd Daemon

The nscd daemon is a process that provides a cache for the most common name service requests. The nscd daemon starts during multiuser boot. The /etc/nscd.conf configuration file controls the behavior of the nscd daemon. The nscd daemon provides caching for the passwd, group, hosts, ipnodes, exec_attr, prof_attr, and user_attr databases. Solaris OS system calls automatically reference the nscd cache if the nscd cache holds the type of data needed. Standardized calls retrieve the cached data. The calls take the form of `getXbyY`, such as `gethostbyname`, `gethostbyaddr`, and so on.

The data in each cache has a separately defined, time-to-live. Modifying the local database, /etc/inet/hosts, for example, causes the corresponding cache to become invalidated upon the next call to the nscd daemon.

Configuring the nscd Daemon

The /etc/nscd.conf file contains the configuration information for the nscd daemon. Each line specifies either an *attribute* and a *value*, or an *attribute*, a *cache name*, and a *value*. An example of an attribute and a value is:

```
logfile          /var/adm/nscd.log
```

An example of an attribute, a cache name, and a value is:

```
enable-cache    hosts      no
# cat /etc/nscd.conf
#
# Copyright (c) 1994-2001 by Sun Microsystems, Inc.
```

Configuring the Name Service Cache Daemon (nscd)

```
# All rights reserved.  
#  
#ident  "@(#)nscd.conf 1.6      01/01/26 SMI"  
  
#  
#      Currently supported cache names: passwd, group, hosts, ipnodes  
#          exec_attr, prof_attr, user_attr  
  
#      logfile          /var/adm/nscd.log  
#      enable-cache     hosts        no  
  
      debug-level       0  
  
      positive-time-to-live  passwd    600  
      negative-time-to-live passwd    5  
      suggested-size       passwd    211  
      keep-hot-count      passwd    20  
      old-data-ok         passwd    no  
      check-files         passwd    yes  
  
      positive-time-to-live  group    3600  
      negative-time-to-live group    5  
      suggested-size       group    211  
      keep-hot-count      group    20  
      old-data-ok         group    no  
      check-files         group    yes  
  
      positive-time-to-live  hosts    3600  
      negative-time-to-live hosts    5  
      suggested-size       hosts    211  
      keep-hot-count      hosts    20  
      old-data-ok         hosts    no  
      check-files         hosts    yes  
  
      positive-time-to-live  ipnodes  3600  
      negative-time-to-live ipnodes  5  
      suggested-size       ipnodes  211  
      keep-hot-count      ipnodes  20  
      old-data-ok         ipnodes  no  
      check-files         ipnodes  yes  
  
      positive-time-to-live  exec_attr  3600  
      negative-time-to-live exec_attr  300  
      suggested-size       exec_attr  211
```

| | | |
|-----------------------|-----------|------|
| keep-hot-count | exec_attr | 20 |
| old-data-ok | exec_attr | no |
| check-files | exec_attr | yes |
| positive-time-to-live | prof_attr | 3600 |
| negative-time-to-live | prof_attr | 5 |
| suggested-size | prof_attr | 211 |
| keep-hot-count | prof_attr | 20 |
| old-data-ok | prof_attr | no |
| check-files | prof_attr | yes |
| positive-time-to-live | user_attr | 3600 |
| negative-time-to-live | user_attr | 5 |
| suggested-size | user_attr | 211 |
| keep-hot-count | user_attr | 20 |
| old-data-ok | user_attr | no |
| check-files | user_attr | yes |

Stopping and Starting the nsqd Daemon

Proper updates to the name service databases notify the nsqd daemon to update its cache, as needed. However, the nsqd daemon's cache might become out of date due to various abnormal circumstances or due to hand-editing files. A common way to force the nsqd daemon to update its cache is to stop and start the daemon.

Disabling the nsqd Daemon

The nsqd daemon is managed by the Service Management Facility (SMF), under the service identifier:

```
svc:/system/name-service-cache:default
```

The Solaris 10 OS installation has the name-service-cache service enabled by default. To stop and disable it, which prevents it from being started on subsequent boots, use the svcadm command as follows:

```
# svcadm disable system/name-service-cache:default
```

Enabling the nscd Daemon

You can manually start the nscd daemon and cause it to be started on subsequent boots, by using the svcadm command as follows:

```
# svcadm enable system/name-service-cache:default
```

Restarting the nscd Daemon

When modifying Role Based Access Control (RBAC) configuration or while testing name service clients, clearing the cache by restarting the daemon can be helpful in removing old cached data:

```
# svcadm restart system/name-service-cache:default
```

Retrieving Name Service Information

There are many tools available for acquiring information stored within the various name service information sources. Selecting the correct tool can reduce troubleshooting time when isolating name service malfunctions. The getent command provides a generic retrieval interface to search many name service databases.

The getent Command

As a system administrator, you can query name service information sources with tools, such as the ypcat, nslookup, niscat, and ldaplist commands.

You can use the ypcat command to query the NIS namespace. You can use the nslookup command to query the DNS namespace. However, when trying to isolate a problem, using one of these tools can return different results than standard system search operations, because the nsswitch.conf file is not referenced by these commands.

The getent command has these advantages:

- The primary advantage is that the command searches the information sources in the order in which they are configured in the name service switch file.
- A secondary advantage is that by using the name service switch file, the defined status message codes and actions are tested as they are currently configured. Therefore, if a return action is improperly placed in the name service switch file, the getent command finds the problem, whereas the specific name service commands used to test the name service information sources, such as ypcat or nslookup, do not find the problem because they directly use the name service database without referencing the nsswitch.conf file.

Using the getent Command

The getent command retrieves a list of entries from the administrative database specified by *database*. The sources for the database are specified in the /etc/nsswitch.conf file. The syntax is:

```
getent database [key] ...
```

where:

database The name of the database to be examined. This name can be passwd, group, hosts, ipnodes, services, protocols, ethers, networks, or netmasks.

key A value that corresponds to an entry in a database. The *key* must be in a format appropriate for searching on the respective database. For example, it can be a user name or numeric user ID (UID) for passwd, or a host name or IP address for hosts. The *key* cannot be a wildcard character.

For the following examples, the /etc/nsswitch.conf file is configured to search files and then to search NIS.

```
# getent passwd lp  
lp:x:71:8:LinePrinter Admin:/usr/spool/lp:  
  
# getent group 10  
staff:::10:  
  
# getent hosts sys44  
192.168.30.44 sys44 loghost
```

The previous example assumes that the /etc/nsswitch.conf file is configured to search files and then to search NIS. If the /etc/nsswitch.conf file is configured to search NIS and then to search files, the output of the final search would be:

```
# getent hosts sys44  
192.168.30.44 sys44
```

Notice the absence of loghost in this output. The loghost alias is a feature of the sys44 entry in the /etc/inet/hosts file but not the NIS map. Therefore, when the /etc/nsswitch.conf file search order is altered, the getent command looks up the entry in the NIS map before consulting the /etc/inet/hosts file.

Exercise: Reviewing Name Services

In this lab, you evaluate your understanding of the name 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. List the name services that can be configured in the /etc/nsswitch.conf file.

2. Which name service is selected by default during the installation of the Solaris 10 OS?

3. What are the two main services provided by DNS?

4. What types of information are stored within the NIS+ namespace?

5. Which file is referred to as the name service switch file, and why?

6. If you decide to use the LDAP for name service resolution, which template file would you use to create the name service switch file?

7. How is the following entry in the name service switch file interpreted?

hosts: nis [NOTFOUND=return] files

8. Is the following an appropriate entry to the /etc/nsswitch.conf file? Why or why not?

group: dns files nis

Exercise Summary

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

- Experiences
- Interpretations
- Conclusions
- Applications



Exercise Solutions: Reviewing Name Services

In this lab, you evaluate your understanding of the name services concepts presented in this module.

Preparation

If necessary, refer to your lecture notes to answer these exercise questions.

Tasks and Solutions

1. List the name services that can be configured in the /etc/nsswitch.conf file.
Local files, DNS, NIS, NIS+, and LDAP.
2. Which name service is the default selection during the installation of the Solaris 10 OS?
NIS+ is selected by default during a Solaris 10 OS installation.
3. What are the two main services provided by DNS?
DNS provides host name-to-IP address translation and IP address-to-host name translation.
4. What types of information are stored within the NIS+ namespace?
The NIS+ namespace stores information about workstation addresses, security information, mail information, Ethernet interfaces, printers, and network services.
5. Which 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 the operating system uses it to determine where to go for any information lookups. This file indicates whether DNS, NIS, NIS+, LDAP, or local files are to be used for name service resolution. If more than one name service is to be used, this file indicates the order in which these services should be accessed.
6. If you decide to use the LDAP for name service resolution, which template file would you use to create the name service switch file?
/etc/nsswitch.ldap
7. How is the following entry in the name service switch file interpreted?

Exercise Solutions: Reviewing Name Services

hosts: nis [NOTFOUND=return] files

Assuming that the NIS name service is running and available, the syntax for this entry means that the NIS hosts table is searched. If an NIS server is busy or unavailable, the local files are searched. If an NIS server has no map entry for a host lookup, the system would not reference the local files.

8. Is the following an appropriate entry to the /etc/nsswitch.conf file? Why or why not?

group: dns files nis

This is not an appropriate entry in the /etc/nsswitch.conf file, because dns only applies to the hosts entry in the name service switch file.

Module 9

Configuring Name Service Clients

Objectives

This module explains how to configure a client to use DNS or LDAP as the name service. Setting up the DNS server is described in the *SA-300-S10: Network Administration for the Solaris™ 10 Operating System* course. Setting up the LDAP server is described in the *IN-351: Using LDAP as a Naming Service* course.

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

- Configure a DNS client
- Configure an LDAP client

Configuring a DNS Client

The following configuration files affect the name resolution process using the Internet domain name system:

| | |
|--------------------|---|
| /etc/resolv.conf | Contains directives telling the resolver library how to perform the DNS query |
| /etc/nsswitch.conf | Should contain the reference to DNS for the hosts entry |

Configuring the DNS Client During Installation

During the system identification phase of a Solaris 10 OS installation, you use several windows to configure the name service. You use function keys or the Escape key to continue through the different windows, depending on the type of installation. For this demonstration, the Escape keys are used.

Note – Text in these screens has been edited for readability, and to fit on the page.



To configure the system to use DNS, complete the following steps:

1. In the Name Service window, select DNS as the name service, then press Esc-2 to continue.

--Name Service-----

On this screen you must provide name service information. Select the name service that will be used by this system, or None if your system will either not use a name service at all, or if it will use a name service not listed here.

- > To make a selection, use the arrow keys to highlight the option and press Return to mark it [X].

Name service

- [] NIS+
- [] NIS
- [X] DNS**
- [] LDAP
- [] None

Esc-2_Continue Esc-6_Help

2. In the Domain Name window, enter the DNS domain name to which the client will belong and press Esc-2 to continue.

--Domain Name-----

On this screen you must specify the domain where this system resides. Make sure you enter the name correctly including capitalization and punctuation.

Domain name: **suned.sun.com**

Esc-2_Continue Esc-6_Help

Configuring a DNS Client

3. In the DNS Server Address window, enter the IP addresses of up to three DNS servers that the client will use for lookups, then press Esc-2 to continue.

--DNS Server Addresses-----

On this screen you must enter the IP address of your DNS server(s). You must enter at least one address. IP addresses must contain four sets of numbers separated by periods (for example 129.200.9.1).

Server's IP address: **192.168.30.61**

Server's IP address:

Server's IP address:

Esc-2_Continue Esc-6_Help

4. In the DNS Search List window, enter search suffixes that will supplement searches for names that are not fully qualified (names that do not include a complete domain name), then press Esc-2 to continue.

--DNS Search List-----

On this screen you can enter a list of domains that will be searched when a DNS query is made. If you do not enter any domains, DNS will only search the DNS domain chosen for this system. The domains entered, when concatenated, may not be longer than 250 characters.

Search domain: **suned.sun.com**

Search domain: **training.sun.com**

Search domain: **classroom.sun.com**

Search domain:

Search domain:

Search domain:

Esc-2_Continue Esc-6_Help

5. In the Confirm Information window, verify that you have provided accurate information, then press Esc-2 to continue.

--Confirm Information-----

- > Confirm the following information. If it is correct, press F2; to change any information, press F4.

Name service: DNS
Domain name: suned.sun.com
Server address(es): 192.168.30.61
Search domain(s): suned.sun.com
 training.sun.com
 classroom.sun.com

Esc-2_Continue Esc-4_Change Esc-6_Help

Editing DNS Client Configuration Files

The installation window only allows the selection of DNS with the default of local files for the name service. Therefore, to use DNS with another name service, such as NIS or LDAP, you must manually modify the configuration files after the system is configured.

Editing the /etc/resolv.conf File

The /etc/resolv.conf file contains configuration directives for the DNS resolver. The directives include:

| | |
|------------|--|
| nameserver | Specifies the IP address of a name server for the DNS domain in which the host is located. You can list up to three name servers, one on each line. |
| domain | Specifies the local domain name. Specifying the local domain name allows queries using just the host name. |
| search | Provides a list of domain names, separated by spaces or tabs, that is appended to unqualified name queries until a match is found. When used without the presence of the domain directive, the first domain listed in the search list is the local domain. |

Domain and search are both valid directives used in the /etc/resolv.conf file, and if both appear together, the last directive listed is used.

The following resolv.conf example shows two name servers for the suned.sun.com domain.

```
# cat /etc/resolv.conf
nameserver 192.168.10.11
nameserver 192.168.20.88
domain suned.sun.com
```

Copying the /etc/nsswitch.dns File to the /etc/nsswitch.conf File

To configure a client to use DNS in combination with the system's local files, copy the /etc/nsswitch.dns file to the /etc/nsswitch.conf file. This action tells the Name Service Switch library to use the DNS source, if nothing is found in the local files for the hosts and ipnodes naming databases:

```
# cat /etc/nsswitch.conf
...
hosts: files dns
...
ipnodes: files dns
...
```

 **Note** – If you want to add DNS name resolution to a system currently running a name service, such as NIS or NIS+, you cannot copy a nsswitch template into the nsswitch.conf file. You must manually edit the current nsswitch file, and place the dns keyword on the hosts line in the specific location, along with other keywords.

The following example shows that DNS is queried after NIS and the /etc/hosts file.

```
# cat /etc/nsswitch.conf
...
hosts: files nis dns
...
```

Setting Up an LDAP Client

Native LDAP is the client implementation of the LDAP name service. An LDAP server, such as the Sun Java Directory Server that is bundled with the Solaris 10 OS, must exist on the network.



Note – The LDAP server cannot be a client of itself. Getting this configuration to work properly requires changes to the LDAP server and the LDAP client.

Client Authentication

An LDAP client can use an LDAP server either in authenticated or in anonymous modes. In the authenticated mode, an LDAP client must establish a session with an LDAP server. This authentication process is known as binding. After a client is authenticated, it can then perform operations, such as “search and modify,” on the data. Authorization is the granting of access to controlled system resources.

By default, Solaris OS LDAP clients have read-only access to name service data, such as host names, email aliases, and net groups. Users have read-write access to certain data, such as their own passwords. Privileged administrator accounts have read-write access to other data. When finished, the client unbinds, or closes, the session.

Details on how the client is authenticated and what data the client is authorized to access is maintained on the LDAP server. To simplify Solaris OS client setup and to avoid having to re-enter the same information for each and every client, a single client profile is created on the directory server.

Client Profile and Proxy Account

A single client profile defines the configuration parameters for a group of Solaris OS clients allowed to access the LDAP database.

A client profile:

- Contains the client's credential information
- Describes how authentication is to take place
- Provides the client with various configuration parameters

A proxy account can be created to allow multiple clients to bind to the server with the same access privileges. Only one name and password is needed for all the clients in a group to bind to the LDAP server, rather than configuring each client with its own account name and password.

Client Initialization

The client profile and proxy account can be created as part of the Sun Java Directory Server setup procedures on the Solaris 10 OS. By default, the client profile named `default` and the proxy account `proxyagent` are created under a special profile directory entry.

When the Solaris LDAP client is initialized, a copy of the client profile is retrieved from the server and stored on disk. On the LDAP client, the `ldap_cachemgr` daemon is responsible for maintaining and updating the changes to the client profile information. The `ldap_cachemgr` daemon keeps a copy of the profile in memory and uses it when binding to the server.

Configuring the LDAP Client During Installation

To configure an LDAP client with an anonymous connection to the Directory Server, perform the following steps:

1. In the Name Service window, select LDAP as the name service, and press Esc-2 to continue.

--Name Service-----

On this screen you must provide name service information. Select the name service that will be used by this system, or None if your system will either not use a name service at all, or if it will use a name service not listed here.

- > To make a selection, use the arrow keys to highlight the option and press Return to mark it [X].

Name service

- [] NIS+
- [] NIS
- [] DNS
- LDAP
- [] None

Esc-2_Continue Esc-6_Help



Note – When you specify LDAP as the name service, the client host name must exist in the ou=hosts container on the LDAP server.

2. In the Domain Name window, enter the domain name where the system is located and press Esc-2 to continue.

--Domain Name-----

On this screen you must specify the domain where this system resides. Make sure you enter the name correctly including capitalization and punctuation.

Domain name: **suned.sun.com**

Setting Up an LDAP Client

Esc-2_Continue Esc-6_Help

3. In the LDAP Profile window, enter the profile name and server IP address, and press Esc-2 to continue.

--LDAP Profile-----

On this screen you must specify the name of the LDAP profile to be used to configure this system, as well as the IP address of the server that contains the profile.

Profile name: **sunedprofile**

Profile server IP address: **192.168.0.1**

Esc-2_Continue Esc-6_Help

4. In the LDAP Proxy Bind window, select No and press Esc-2 to continue.

--Provide LDAP Proxy Bind Information-----

If the profile you are using specifies a proxy credential level and the authentication method is NOT none, provide LDAP proxy bind information.

- > Use the arrow keys to select the option and press Return to mark it [X].

Specify LDAP Proxy Bind Information

- [**X**] No
[] Yes

Esc-2_Continue Esc-6_Help

5. In the Confirm Information window, verify that you have provided accurate information, and press Esc-2 to continue.

--Confirm Information-----

- > Confirm the following information. If it is correct, press F2; to change any information, press F4.

Name service: LDAP
Domain name: suned.sun.com
Profile name: sunedprofile
Profile server IP address: 192.168.0.1
Specify LDAP Proxy Bind Information: No

Esc-2_Continue Esc-4_Change Esc-6_Help



Note – The information that must be supplied during the installation is some of the same information that you would enter using the `ldapclient` command.

Initializing the Native LDAP Client

You execute the `ldapclient` command on the client system once to initiate the client as a native LDAP client. The required command-line arguments include the LDAP server's IP address.

The following example describes a typical client initialization:

```
# ldapclient init -a proxyPassword=proxy \
-a proxyDN=cn=proxyagent,ou=profile,dc=suned,dc=sun,dc=com\
-a domainName=suned.sun.com 192.168.0.100
System successfully configured
```

where:

| | |
|---------------|---|
| init | Initializes the host as an LDAP client |
| proxyPassword | The password for the proxyagent |
| proxyDN | The DN for the proxyagent |
| domainname | The domain for which the server is configured |
| 192.168.0.100 | LDAP server IP address |

Clients bind to the directory using a proxy account. Different proxy accounts can be configured so that LDAP users only have access to the directory data that they should have access to. This is different for an anonymous account, which has access to all of the data stored in the directory.

Each proxy account should have a password. The password is stored on the LDAP client.

The `ldapclient` command creates two files in the `/var/ldap` directory on the LDAP client. These files contain the information that the LDAP clients use when binding to and accessing the LDAP database.

Note – The two files in the `/var/ldap` directory are currently ASCII files, but might not be in the future. The `ldapclient list` command is the best way to see this information.



The `ldap_client_cred` file contains the proxy agent information that the client uses for LDAP authentication; for example:

```
# cat /var/ldap/ldap_client_cred
#
# Do not edit this file manually; your changes will be lost. Please use
ldapclient (1M) instead.
#
NS_LDAP_BINDDN= cn=proxyagent,ou=profile,dc=suned,dc=sun,dc=com
NS_LDAP_BINDPASSWD= {NS1}ecc423aad0
```

The `ldap_client_file` file contains the configuration information from the client profile in the LDAP server database; for example:

```
# cat /var/ldap/ldap_client_file
#
# Do not edit this file manually; your changes will be lost. Please use
ldapclient (1M) instead.
#
NS_LDAP_FILE_VERSION= 2.0
NS_LDAP_SERVERS= 192.168.0.100
NS_LDAP_SEARCH_BASEDN= dc=suned,dc=sun,dc=com
NS_LDAP_AUTH= simple
NS_LDAP_SEARCH_REF= FALSE
NS_LDAP_SEARCH_SCOPE= one
NS_LDAP_SEARCH_TIME= 30
NS_LDAP_CACHTTL= 43200
NS_LDAP_PROFILE= default
NS_LDAP_CREDENTIAL_LEVEL= proxy
NS_LDAP_BIND_TIME= 10
```

Note – Do not modify the `/var/ldap/ldap_client_file` file directly.



You can also use the `ldapclient` command to view the current client's local configuration. Refer to the `ldapclient` man page for a description of these attributes.

```
# ldapclient list
NS_LDAP_FILE_VERSION= 2.0
NS_LDAP_BINDDN= cn=proxyagent,ou=profile,dc=suned,dc=sun,dc=com
NS_LDAP_BINDPASSWD= {NS1}ecc423aad0
NS_LDAP_SERVERS= 192.168.0.100
NS_LDAP_SEARCH_BASEDN= dc=suned,dc=sun,dc=com
NS_LDAP_AUTH= simple
NS_LDAP_SEARCH_REF= FALSE
NS_LDAP_SEARCH_SCOPE= one
NS_LDAP_SEARCH_TIME= 30
NS_LDAP_PROFILE= default
```

```
NS_LDAP_CREDENTIAL_LEVEL= proxy  
NS_LDAP_BIND_TIME= 10
```

Copying the /etc/nsswitch.ldap File to the /etc/nsswitch.conf File

During LDAP client initialization, the /etc/nsswitch.ldap file is copied over the /etc/nsswitch.conf file.

The default nsswitch.conf file for an LDAP client follows.

```
# more nsswitch.conf  
#  
# An example file that could be copied over to /etc/nsswitch.conf; it  
# uses LDAP in conjunction with files.  
#  
# "hosts:" and "services:" in this file are used only if the  
# /etc/netconfig file has a "-" for nametoadr_libs of "inet" transports.  
  
# LDAP service requires that svc:/network/ldap/client:default be enabled  
# and online.  
  
# the following two lines obviate the "+" entry in /etc/passwd and  
# /etc/group.  
passwd:      files ldap  
group:      files ldap  
  
# consult /etc "files" only if ldap is down.  
hosts:      ldap [NOTFOUND=return] files  
  
# Note that IPv4 addresses are searched for in all of the ipnodes  
# databases  
# before searching the hosts databases.  
ipnodes:    ldap [NOTFOUND=return] files  
  
networks:   ldap [NOTFOUND=return] files  
protocols:  ldap [NOTFOUND=return] files  
rpc:        ldap [NOTFOUND=return] files  
ethers:     ldap [NOTFOUND=return] files  
netmasks:   ldap [NOTFOUND=return] files  
bootparams: ldap [NOTFOUND=return] files  
publickey:  ldap [NOTFOUND=return] files  
  
netgroup:   ldap
```

```
automount: files ldap
aliases: files ldap

# for efficient getservbyname() avoid ldap
services: files ldap

printers: user files ldap

auth_attr: files ldap
prof_attr: files ldap

project: files ldap
```

Listing LDAP Entries

You use the `ldaplist` command to list the naming information from the LDAP servers. This command uses the application programming interface (API) to access the information. Refer to the `ldaplist` man page for additional information.

Without any arguments, the `ldaplist` command returns all of the containers in the current search baseDN. For example:

```
# ldaplist
dn: ou=Hosts,dc=suned,dc=sun,dc=com

dn: ou=Group,dc=suned,dc=sun,dc=com

dn: ou=rpc,dc=suned,dc=sun,dc=com

dn: ou=protocols,dc=suned,dc=sun,dc=com

dn: ou=networks,dc=suned,dc=sun,dc=com

dn: ou=netgroup,dc=suned,dc=sun,dc=com

dn: ou=aliases,dc=suned,dc=sun,dc=com

dn: ou=people,dc=suned,dc=sun,dc=com

dn: ou=services,dc=suned,dc=sun,dc=com

dn: ou=Ethers,dc=suned,dc=sun,dc=com
```

Setting Up an LDAP Client

```
dn: ou=profile,dc=suned,dc=sun,dc=com  
dn: nismapname=auto_home,dc=suned,dc=sun,dc=com  
dn: nismapname=auto_direct,dc=suned,dc=sun,dc=com  
dn: nismapname=auto_master,dc=suned,dc=sun,dc=com
```

Unconfiguring an LDAP Client

To unconfigure an LDAP client, use the `ldapclient` command with the `uninit` option. This command removes the client files from the `/var/ldap` directory and restores the previous `/etc/nsswitch.conf` file. The `ldap_cachemgr` process is also stopped. The changes to the client name service configuration are dynamic; therefore, no reboot is needed.

```
# ldapclient uninit  
System successfully unconfigured
```

Exercise: Configuring a System to Use DNS and LDAP

In this exercise, you configure the Solaris 10 OS client system to use DNS and LDAP as name services.

Preparation

Refer to the lecture notes to perform the tasks listed. The instructor's system is configured as a DNS server and as an LDAP server for the classroom network, using a domain name of suned.sun.com.

Tasks

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Add DNS to the name service by copying the /etc/nsswitch.dns file to the /etc/nsswitch.conf file.
3. Create the /etc/resolv.conf file and complete the following steps:
 - a. Add a nameserver directive by using the address of your instructor system, for example: 192.168.30.30.
 - b. Add a domain directive, for example: suned.sun.com.
 - c. Add a search list directive, for example: suned.sun.com.

Note – The name server IP address and domain might be differ from those used in this example. If you need help, check with your instructor.



4. Ensure that the DNS client service is running by enabling it through SMF.
5. Verify that you can access another system in the classroom by using the ping command.
First, use only the host name, and then use the fully qualified domain name, for example: *hostname.suned.sun.com*.
6. Set up your domain name on your client system, for example:
vi /etc/defaultdomain

Exercise: Configuring a System to Use DNS and LDAP

```
suned.sun.com  
# domainname suned.sun.com
```

7. Complete the following steps:
 - a. Use the ldapclient command to initialize the system.
The name of the profile is default.
 - b. Use the IP address of your instructor system.

```
# ldapclient -v init -a proxyPassword=proxy \  
-a proxyDN=cn=proxyagent,ou=profile,dc=suned,dc=sun,dc=com \  
-a domainname=suned.sun.com <IP address of instructor>
```

8. Verify the name service switch file has been updated with the LDAP configuration.
hosts: ldap [NOTFOUND=return] files
9. Verify that you can access another system in the classroom by using the ping command.
First, use only the host name, and then use the fully qualified domain name, for example: *hostname.suned.sun.com*.
10. Display the DIT containers.
11. Display the Hosts container.
12. Unconfigure the LDAP client.
13. Verify that the LDAP configuration has been removed from the name service switch file.

Exercise Summary



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

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise Solutions: Configuring a System to Use DNS and LDAP

In this exercise, you configure the Solaris 10 OS client system to use DNS and LDAP as name services.

Preparation

Refer to the lecture notes to perform the tasks listed. The instructor's system is configured as a DNS server and as an LDAP server for the classroom network, typically using a domain name of suned.sun.com. Your instructor will provide the correct domain name to use if it is not suned.sun.com.

Tasks and Solutions

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Add DNS to the name service by copying the /etc/nsswitch.dns file to the /etc/nsswitch.conf file.

```
# cp /etc/nsswitch.dns /etc/nsswitch.conf  
cp: overwrite /etc/nsswitch.conf (yes/no)? yes
```

3. Create the /etc/resolv.conf file and complete the following steps:
 - a. Add a nameserver directive by using the address of your instructor system, for example: 192.168.30.30.
 - b. Add a domain directive, for example: suned.sun.com.
 - c. Add a search list directive, for example: suned.sun.com.

Note – The name server IP address and domain might be differ from those used in this example. If you need help, check with your instructor.



```
# vi /etc/resolv.conf  
nameserver 192.168.30.30  
domain suned.sun.com  
search suned.sun.com
```

4. Ensure that the DNS client service is running by enabling it through SMF.

```
# svcadm enable svc:/network/dns/client:default
```

5. Verify that you can access another system in the classroom by using the ping command.

First, use only the host name, and then use the fully qualified domain name, for example: *hostname.suned.sun.com*.

```
# ping sys43
```

sys43 is alive

```
# ping sys43.suned.sun.com
```

sys43.suned.sun.com is alive

6. Set up your domain name on your client system, for example:

```
# vi /etc/defaultdomain
```

suned.sun.com

```
# domainname suned.sun.com
```

7. Complete the following steps:

- a. Use the ldapclient command to initialize the system.

The name of the profile is default.

- b. Use the IP address of your instructor system.

```
# ldapclient -v init -a proxyPassword=proxy \
-a proxyDN=cn=proxyagent,ou=profile,dc=suned,dc=sun,dc=com \
-a domainname=suned.sun.com <IP address of instructor>
```

Parsing proxyPassword=proxy

Parsing proxyDN=cn=proxyagent,ou=profile,dc=suned,dc=sun,dc=com

Parsing domainname=suned.sun.com

(output omitted for brevity)

restart: milestone/name-services:default... success

System successfully configured

```
#
```

8. Verify the name service switch file has been updated with the LDAP configuration.

hosts: ldap [NOTFOUND=return] files

```
# grep hosts: /etc/nsswitch.conf
```

"hosts:" and "services:" in this file are used only if the

hosts: ldap [NOTFOUND=return] files

9. Verify that you can access another system in the classroom by using the ping command.

First, use only the host name, and then use the fully qualified domain name, for example: *hostname.suned.sun.com*.

```
# ping sys43
sys43 is alive
# ping sys43.suned.sun.com
sys43.suned.sun.com is alive
```

10. Display the DIT containers.

```
# ldaplist
dn: cn=Directory Administrators, dc=suned,dc=sun,dc=com
(output omitted for brevity)
#
```

11. Display the Hosts container.

```
# ldaplist hosts
(output omitted for brevity)
dn: cn=sys-01,ou=hosts,dc=suned,dc=sun,dc=com
dn: cn=sys-01.suned.sun.com,ou=hosts,dc=suned,dc=sun,dc=com
#
```

12. Unconfigure the LDAP client.

```
# ldapclient -v uninit
Arguments parsed:
Handling uninit option
Restoring machine to previous configuration state
(output omitted for brevity)
restart: milestone/name-services:default... success
System successfully recovered
#
```

13. Verify that the LDAP configuration has been removed from the name service switch file.

```
# grep hosts: /etc/nsswitch.conf
# "hosts:" and "services:" in this file are used only if the
hosts:      files
#
```

Module 10

Introduction to Zones

Objectives

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

- Identify the different zones features
- Understand how and why zone partitioning is used
- Configure zones
- Install zones
- Boot zones
- Move a zone
- Migrate a zone
- Delete a zone
- Administer packages with zones
- Upgrade the Solaris 10 OS with installed zones
- Use lx Branded Zones

Introducing Solaris Zones

Solaris zones technology enables software partitioning of a Solaris 10 OS to support multiple independent operating systems with independent process space, allocated resources, and users. Zones are ideal for environments that consolidate a number of applications on a single server. The cost and complexity of managing numerous machines makes it advantageous to consolidate several applications on larger, more scalable servers.

Server Consolidation Solutions

When planning to consolidate servers, there are many solutions in the marketplace. Consumers can choose from three categories of server consolidation solutions:

- Domains and Partitions – These are consolidation schemes based on hardware solutions. This includes Sun Fire™ Domains and IBM LPARs.
- Virtual Machine – This is an application-level consolidation solution. This includes IBM VM, VMware and xVM Server.
- Operating System Partitions – This is an operating system-level solution. This includes FreeBSD Jails and Linux Vservers.

Solaris Zones are in the Operating System Partitions category.

Zones provide virtual operating system services that look like different Solaris instances to users and applications. This architecture isolates processes, hides the underlying platform, and enables the global administrator to allow the use of system resources on a granular level. This separation can create a more secure environment, where multiple applications that previously had to run on different physical systems can coexist, in different zones, on one machine.

Note – All native zones (Solaris 10 zones on Solaris 10) must run the same version of the Solaris Operating System.



Resource Sharing

Zones allow the root user of the global zone to dedicate system resources to individual zones. Each zone maintains its own root password and user information, separate from other zones and the global zone. Each zone exists with separate process and file system space, and can only monitor and interact with local processes. A single processor and single disk system can support several zones, each with separate resources, users, and process space as shown in Figure 10-1.

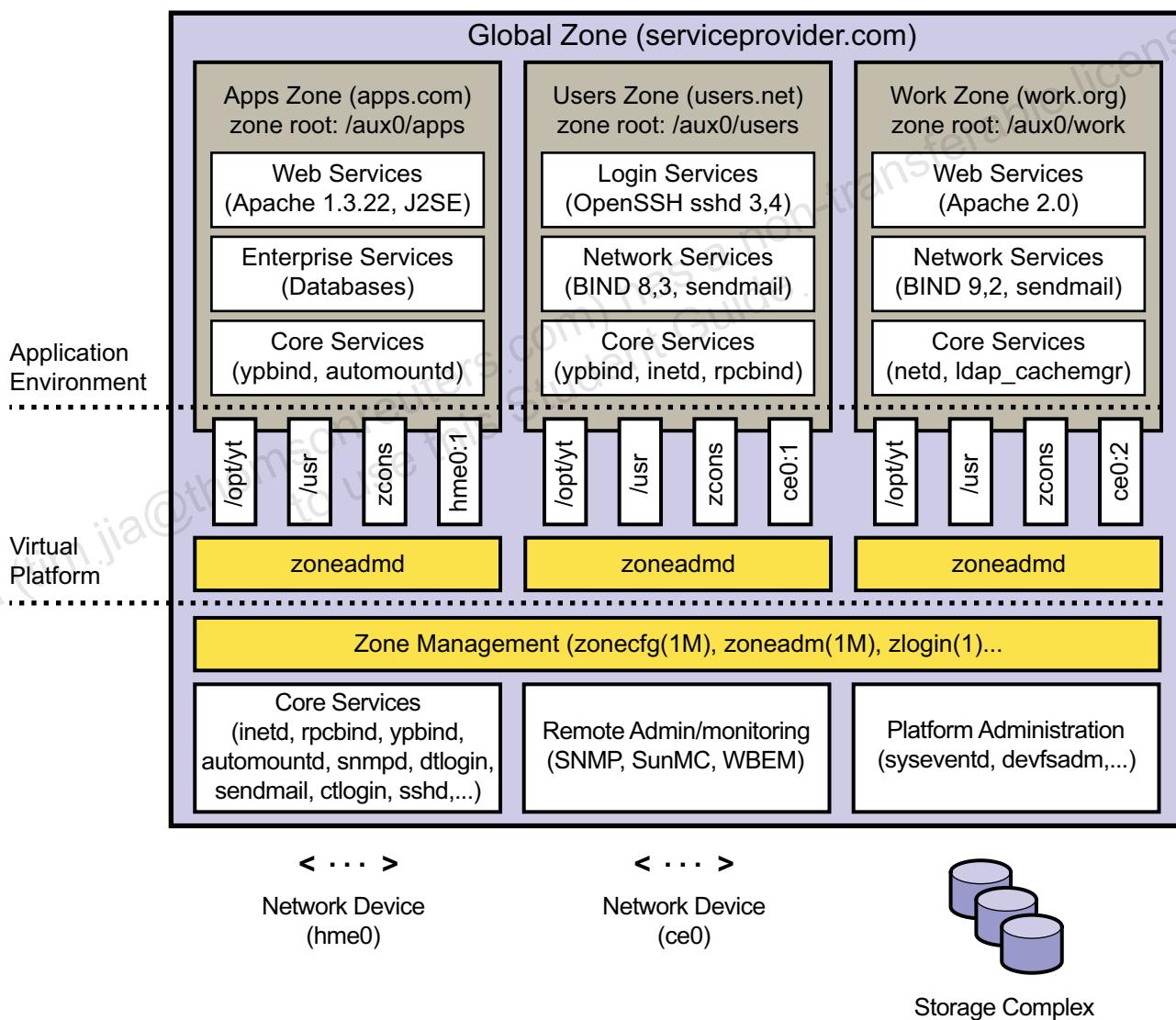


Figure 10-1 Typical Zones Environment

Zones allow multiple Solaris instances to operate at the same time on a single hardware platform. File systems, processors, and network interfaces can be shared by multiple zones. Allotment of physical resources to more than one instance allows scaling and sharing of available resources on an as-needed basis. Individual zones can gain files and configurations from the global zone.

Zone Features

- **Security** – Network services can be run in a zone, limiting the potential damage in the event of a security violation. Processes running within a zone, even one with superuser credentials, cannot affect activity in other zones. Certain activities, such as rebooting or shutting down the whole system, are only permitted in the global zone. An administrator logged into the global zone can monitor the activity of applications running in other zones and control the system as a whole.
- **Isolation** – Zones allow the deployment of multiple applications on the same machine, even if the applications operate in different trust domains, require exclusive use of a global resource, or present difficulties with global configurations. Individual zones have their own set of users and their own root password. When these zones are rebooted, any other zones running on the system are unaffected.
- **Virtualization** – Zones provide an artificial environment that can hide such details as physical devices, the system's primary Internet protocol (IP) address, and host name from the application. Since the same environment can be maintained on different physical machines, this can be useful in supporting rapid deployment and redeployment of applications.
- **Granularity** – Zones can provide isolation at arbitrary granularity. A zone does not require a dedicated central processing unit (CPU), physical device, or chunk of physical memory. These resources can be multiplexed across a number of zones running within a single system, or allocated on a per-zone basis, using resource management features available in the OS.
- **Transparency** – Except when necessary to achieve security and isolation, zones avoid changing the environment in which applications execute. Zones do not present a new API or application binary interface (ABI) to which applications must be ported. Instead they provide the standard Solaris interfaces and application environment, with some restrictions on applications attempting to perform privileged operations.

Zone Concepts

To understand the Solaris zone partitioning technology, the following concepts should be understood:

- Zone types
- Zone daemons
- Zone file systems
- Zone networking
- Zone states

Zone Types

The Solaris Operating System supports three types of zones:

- Global zone
- Non-global zone
- Branded zones

Global Zones

Every Solaris system contains a global zone (Figure 10-1 on page 10-3). The global zone has two functions. The global zone is both the default zone for the system and the zone used for system-wide administrative control. The global zone is the only zone from which a non-global zone can be configured, installed, managed, or uninstalled. All processes run in the global zone if no non-global zones are created by the global administrator.

Only the global zone is bootable from the system hardware. The global zone contains a complete installation of the Solaris system software packages. It can contain additional software not installed through packages.

Note – Exclusive IP stack zones allow for management of network interfaces, including routing.

The global zone is the only zone from which a non-global zone can be configured, installed, managed, or uninstalled. Appropriately privileged processes running in the global zone can access objects associated with other zones.

Unprivileged processes in the global zone might be able to perform operations not allowed to privileged processes in a non-global zone. For example, users in the global zone can view information about every process in the system. If this capability presents a problem for your site, you can restrict access to the global zone.

The global zone provides a complete database containing information about all installed components. It also holds configuration information specific to the global zone only, such as the global zone host name and file system table. The global zone is the only zone that is aware of all devices and all file systems.

Each zone, including the global zone, is assigned a zone name. The global zone always has the name *global*. Each zone is also given a unique numeric identifier, which is assigned by the system when the zone is booted. The global zone is always mapped to zone ID 0. The system assigns a new non-zero zone ID to a non-global zone when it reboots.

Non-Global Zone

The non-global zones contain an installed subset of the complete Solaris Operating System software packages. They can also contain Solaris software packages shared from the global zone and additional installed software packages not shared from the global zone.

Non-global zones can contain additional software created on the non-global zone that are not installed through packages or shared from the global zone.

The non-global zones share operation under the Solaris kernel booted from the global zone. They are assigned a non-zero zone ID by the system when the zone is booted and must have a user defined name.

The non-global zone is not aware of the existence of any other zones. It cannot install, manage, or uninstall itself or any other zones.

Branded Zones

Branded zones are available beginning with the Solaris 10 8/07 release.

The branded zone (BrandZ) framework extends the Solaris Zones infrastructure. BrandZ provides the framework to create non-global branded zones that contain non-native operating environments used for running applications.

Branded zones are used on the Solaris Operating System to run applications. The first brand available is the lx brand, Solaris Containers for Linux Applications. The lx brand provides a Linux environment for your applications and runs on x86 and x64 machines. The lx brand uses the branded zones framework to enable Linux binary applications to run unmodified on a machine with a Solaris Operating System kernel.

You cannot run Solaris applications inside an lx zone. However, the lx zone enables you to use the Solaris system to develop, test, and deploy Linux applications. For example, you can place a Linux application in an lx zone and analyze it using Solaris tools run from the global zone. You can then make improvements and deploy the tuned application on a native Linux system.

Note – See

<http://opensolaris.org/os/community/brandz/applications> for a list of some applications that have been successfully run under the lx brand.

A brand can provide a simple or a complex environment. For example, a simple environment could replace the standard Solaris utilities with their GNU equivalents. A complex environment could provide a complete Linux user space which supports the execution of Linux applications.

Every zone is configured with an associated brand. The default is the native brand, Solaris. A branded zone will support exactly one brand of non-native binary, which means that a branded zone provides a single operating environment.

The branded zone framework extends the zones tools in the following ways:

- The `zonecfg` command is used to set a zone's brand type when the zone is configured.
- The `zoneadm` command is used to report a zone's brand type as well as administer the zone.

Note – You can change the brand of a zone in the configured state. Once a branded zone has been installed, that brand cannot be changed or removed.

Zone Daemons

The system uses two daemons to control zone operation: `zoneadmd` and `zsched`.

The `zoneadmd` daemon is the primary process for managing the zone's virtual platform. There is one `zoneadmd` process running for each active (ready, running, or shutting down) zone on the system.

The `zoneadmd` daemon is responsible for:

- Managing zone booting and shutting down
- Allocating the zone ID and starting the `zsched` system process.
- Setting zone-wide resource controls
- Preparing the zone's devices as specified in the zone configuration
- Plumbing virtual network interfaces
- Mounting loopback and conventional file systems

Unless the `zoneadmd` daemon is already running, it is automatically started by the `zoneadm` command.

Every active zone has an associated kernel process, `zsched`. The `zsched` process enables the zones subsystem to keep track of per-zone kernel threads. Kernel threads doing work on behalf of the zone are owned by `zsched`.

Zone File Systems

There are two models for installing root file systems in non-global zones, the sparse root model and the whole root model.

Sparse Root Model

The sparse root model installs a minimal number of files from the global zone when a non-global zone is installed. In this model, only certain *root* packages are installed in the non-global zone. These include a subset of the required root packages that are normally installed in the global zone, as well as any additional root packages that the global administrator might have selected. Any files that need to be shared between a non-global zone and the global zone can be mounted as read-only loopback file systems.

By default, the directories `/lib`, `/platform`, `/sbin`, and `/usr` are mounted in this manner. An example of shared file systems is shown in Figure 10-2.

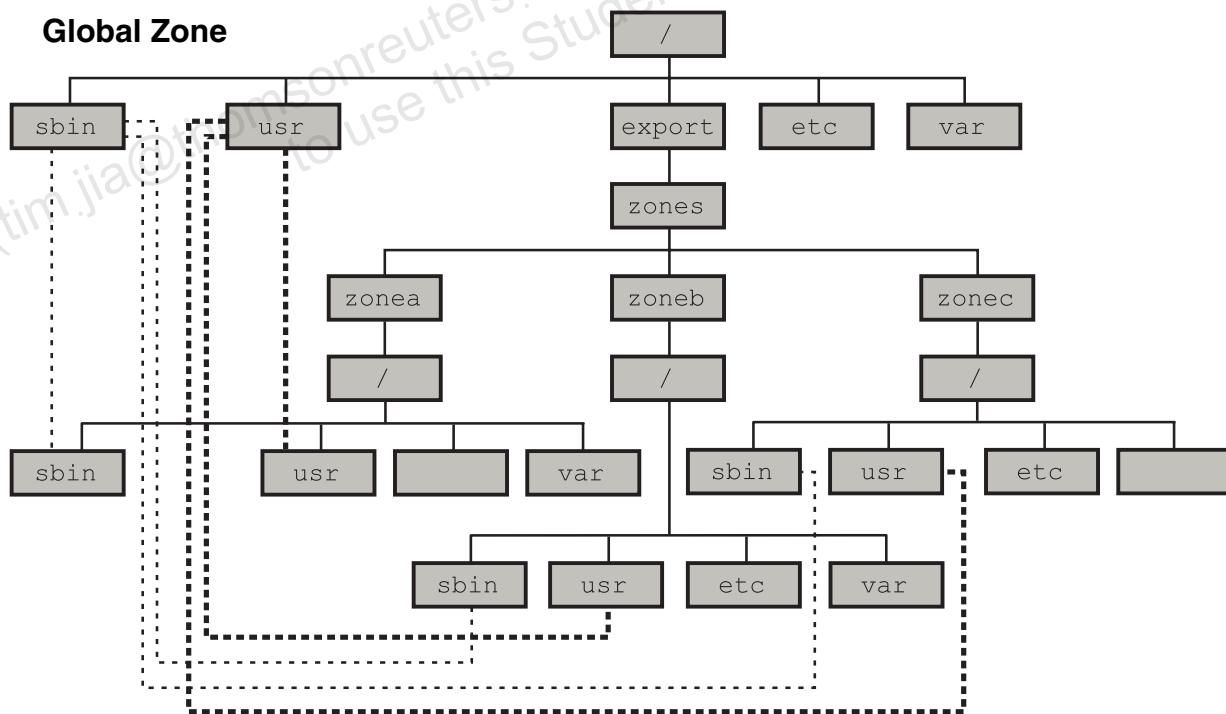


Figure 10-2 Shared File System Example

Once a zone is installed it is no longer dependent on the global zone unless a file system is mounted using a loopback file system. If a critical file is removed from a zone, only that zone is affected. If a critical file is removed from the global zone, and the global zone operating system fails, then each zone would also fail. If the global operating system did not fail, and the zone was not in need of that removed file, the zones would be unaffected.

For files that are mounted using the loopback file system, the removal of a critical file from the global zone would be the same as if it were in a typical client-server situation. The zone's dependence on the file would determine the effect of its removal on the zone.

Note – A non-global zone cannot be an NFS server.



Whole Root Model

The whole root model provides the maximum flexibility. All of the required and any selected optional Solaris packages are installed into the private file systems of the zone. The advantages of this model include the capability for global zone administrators to customize their zones file system layout. This would be done, for example, to add arbitrary unbundled or third-party packages. The disk requirements for this model are determined by the disk space used by the packages currently installed in the global zone.

Note – Once a zone is installed as a sparse root or a whole root, the zone cannot be changed without uninstalling and reinstalling the zone.

Zone Networking

There are two IP types available for non-global zones: shared-IP stack and exclusive-IP stack.

The shared-IP zone shares a network interface and the exclusive-IP zone must have a dedicated network interface.

Each non-global zone that requires network connectivity has one or more dedicated IP addresses. These addresses are associated with logical network interfaces. For example, if the primary network interface in the global zone is `ce0`, then the non-global's logical network interface is `ce0:1`.

Zone interfaces configured by `zonecfg` will automatically be plumbed and placed in the zone when it is booted. Only the global zone administrator can modify the interface configuration and the network routes.

IPMP can be configured in the global zone, and the functionality extended to non-global zones. The functionality is extended by placing the zone's IP address in an IPMP group when configuring the zone. Then, if one of the interfaces in the global zone fails, the non-global zone addresses will migrate to another network interface card.

Zone States

To understand the operability of a zone you need to understand its state. Zones behave like typical Solaris 10 OS installations, but do not have resources such as power-on self-test (POST) or OpenBoot Programmable Read-only Memory (OBP). Instead, these settings and tests are managed by the global zone. As a zone is configured, enabled, and used, its status field in the `zoneadm list` command output changes. Figure 10-3 shows the zone states.

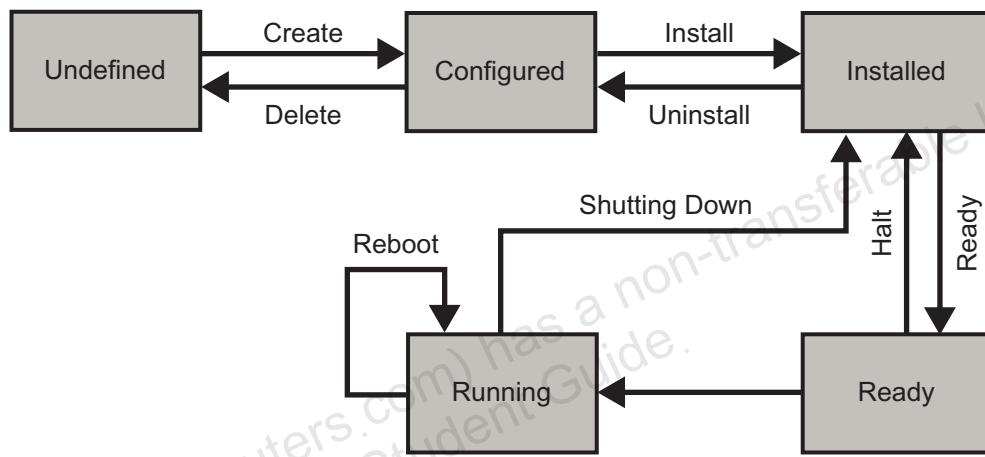


Figure 10-3 Zone States

The possible zone states are as follows:

- **Undefined** – In this state, the zone's configuration has not been completed and committed to stable storage. This state also occurs when a zone's configuration has been deleted.
- **Configured** – In this state, the zone's configuration is complete and committed to stable storage. However, those elements of the zone's application environment that must be specified after initial boot are not yet present.
- **Incomplete** – This is a transitional state. During an install or uninstall operation, `zoneadm` sets the state of the target zone to incomplete. Upon successful completion of the operation, the state is set to the correct state. However, a zone that is unable to complete the install process will stop in this state.

- Installed – During this state, the zones configuration is instantiated on the system. The zoneadm command is used to verify that the configuration can be successfully used on the designated Solaris system. Packages are installed under the zones root path. In this state, the zone has no associated virtual platform.
- Ready – In this state, the virtual platform for the zone is established. The kernel creates the zsched process, network interfaces are plumbed, file systems are mounted, and devices are configured. A unique zone ID is assigned by the system. At this stage, no processes associated with the zone have been started.
- Running – In this state, the user processes associated with the zone application environment are running. The zone enters the running state as soon as the first user process associated with the application environment (init) is created.
- Shutting – down and Down - These states are transitional states that are visible while the zone is being halted. However, a zone that is unable to shut down for any reason will stop in one of these states.

Configuring Zones

To configure a zone, you must perform these tasks:

- Identify the components that will make up the zone.
- Configure the zone.
- Verify and commit the configured zone.

Identifying Zone Components

When planning zones for your environment, you must consider the components that make up each zones configuration. These components include:

- A zone name
- A zone path to the zone's root
- The zone network interfaces
- The file systems mounted in zones
- The configured devices in zones

Note – IP networking can now be configured in two different ways, depending on whether the zone is assigned an exclusive IP instance or shares the IP layer configuration and state with the global zone. IP types are configured by using the zonecfg command.



Allocating File System Space

There are no limits on how much disk space can be consumed by a zone. The global zone administrator is responsible for space restriction. Even a small single processor system can support a number of zones running simultaneously. The nature of the packages installed in the global zone affects the space requirements of the non-global zones that are created. The number of packages and space requirements are factors.

- As a general guideline, about 100 megabytes of free disk space per non-global zone is required when the global zone has been installed with all of the standard Solaris packages.

- By default, any additional packages installed in the global zone also populate the non-global zones. The amount of disk space required must be increased accordingly. The directory location in the non-global zone for these additional packages is specified through the `inherit-pkg-dir` resource.

You can use standard partitions to divide disk slices or soft partitions to divide logical volumes into partitions. You can use these partitions as zone roots, and thus limit per-zone disk consumption. The soft partition limit is 8192 partitions.

An additional 40 megabytes of RAM per zone are suggested, but not required on a machine with sufficient swap space.

Using the `zonecfg` Command

The `zonecfg` command is used to configure a zone. The `zonecfg` command can be used in interactive mode, in command-line mode, or in command-file mode. The following operations can be performed using this command:

- You can create or delete a zone configuration.
- You can set properties for resources added to a configuration.
- You can query or verify a configuration.
- You can commit to a configuration.
- You can revert to a previous configuration.
- You can exit from a `zonecfg` session.

Note – There are many other operations that can be accomplished with the `zonecfg` command, but they are outside of the scope of this course.



To simplify the user interface, `zonecfg` utilizes the concept of a *scope*. The default scope is `global`. You can use the `add` and `select` subcommands to select a specific resource, at which point the scope changes to that resource. The `zonecfg` interactive command prompt changes to reflect the current scope. The `end` and `cancel` subcommands are used to complete the resource specification, at which time the scope reverts back to `global`. Certain subcommands, such as `add`, `remove` and `set`, have different semantics in each scope.

There are several subcommands to configure and provision zones within the zonecfg utility, as shown in Table 10-1. Several subcommands affect the environment, depending on the current scope. The zonecfg prompt indicates if the scope is global or resource scope. Many of the subcommands also allow the -f, or force, flag. If this flag is given, the subcommand does not use interactive questioning safeguards.

Table 10-1 The zonecfg Subcommands

| Command | Description |
|---------|--|
| add | Add a resource to the zone. |
| cancel | Exits from resources scope back to global. Partially specified resources are abandoned. |
| commit | Verifies settings and commits proper settings from memory to disk. The revert subcommand will return to this point. |
| create | Create an in-memory configuration for the specified zone. |
| delete | Delete the configuration from memory. |
| end | Verify that parameters have been assigned and return to the global scope. |
| export | Print the configuration to stdout, or to the output file specified, in form that can be used in a command file. |
| info | Display current configuration for resource settings or global zonepath, autoboot or pool. |
| remove | Removes one or more resource depending on scope. |
| select | Find a resource whose parameters are matched within the curly braces and change to its scope. |
| set | Set an in-memory value for a parameter. |
| verify | Verify the in-memory configuration and that all resources have required properties specified and the zonepath is specified for the zone. |
| revert | Discard any in-memory configurations and return to the last time a commit was performed. |

Table 10-1 The zonecfg Subcommands (Continued)

| Command | Description |
|---------|---|
| exit | Commit current in-memory settings and exit the zonecfg utility. This command will automatically commit the configuration information to stable storage. |

The zonecfg Resources Parameters

Resource types within the zonecfg utility include the following:

- zonename – Defines the zone name and identifies the zone to the configuration utility.
- zonepath – Defines the zone path resource and is the path to the zone root.
- autoboot – Determines if the zone will reboot when the global zone reboots.
- fs – Assigns resource parameters for file systems. Use of the special parameter allows the local zone to mount global file system resources under separate directories. Table 10-2 shows parameters associated with the fs resource.

Table 10-2 The fs Resource Parameters

| | |
|---------|--|
| dir | Mount point in the non-global zone |
| special | Block device file specifying the location of the file system |
| raw | Device file to use for the fsck command |
| type | The file system type |
| options | Allow parameters similar to those found with the mount command |

- inherit-pkg-dir – Gives access to software packages from the global system. The contents of software packages in the inherit-pkg-dir directory are inherited by the non-global zone in a read-only mode. The default inherit-pkg-dir resources are: /lib, /platform, /sbin, and /usr.

- net – Provisions logical interfaces of the global systems interfaces to non-global zones. The network interfaces are plumbed when the zone transitions from the installed state to the ready state.
- device – References devices for the select, add, or remove commands. Each zone can have devices that should be configured when the zone transitions from the installed state to the ready state.
- attr – Enables the global administrator to assign generic-attribute settings, such as name type and value. The type must be int, uint (unsigned), Boolean or string.

Resource Limits and Resource Controls

The resource limit facility (rlimit) allows administrators to set one or more numerical limits on the amount of resources a process can consume. These limits include per-process CPU time used, per-process core file size, and per-process maximum heap size. Heap size is the amount of scratch memory that is allocated for the process data segment.

The resource controls facility (rctl) provides compatibility interfaces for the resource limits facility. Existing applications that use resource limits continue to run unchanged. These applications can be observed in the same way as applications that are modified to take advantage of the resource controls facility.

Zone Configuration Walk-Through

To create a zone, you must log into the global system as root or role based access control (RBAC)-allowed user. The following shows an example of configuring a zone named work-zone:

```

1  global# zonecfg -z work-zone
2  zonecfg:work-zone> create
3  zonecfg:work-zone> set zonepath=/export/work-zone
4  zonecfg:work-zone> set autoboot=true
5  zonecfg:work-zone> add fs
6  zonecfg:work-zone:fs> set dir=/mnt
7  zonecfg:work-zone:fs> set special=/dev/dsk/c0t0d0s7
8  zonecfg:work-zone:fs> set raw=/dev/rdsck/c0t0d0s7
9  zonecfg:work-zone:fs> set type=ufs
10 zonecfg:work-zone:fs> add options [logging]
11 zonecfg:work-zone:fs> end
12 zonecfg:work-zone> add inherit-pkg-dir
13 zonecfg:work-zone:inherit-pkg-dir> set dir=/opt/sfw
14 zonecfg:work-zone:inherit-pkg-dir> end
15 zonecfg:work-zone> add net
16 zonecfg:work-zone:net> set physical=ce0
17 zonecfg:work-zone:net> set address=192.168.0.1
18 zonecfg:work-zone:net> end
19 zonecfg:work-zone> add device
20 zonecfg:work-zone:device> set match=/dev/sound/*
21 zonecfg:work-zone:device> end
22 zonecfg:work-zone> add attr
23 zonecfg:work-zone:attr> set name=comment
24 zonecfg:work-zone:attr> set type=string
25 zonecfg:work-zone:attr> set value="The work zone."
26 zonecfg:work-zone:attr> end
27 zonecfg:work-zone> verify
28 zonecfg:work-zone> commit
29 zonecfg:work-zone> exit
```

Line 1 – This line starts the zonecfg utility in interactive mode. The zone is called work-zone.

Line 2 – This line begins the in-memory configuration.

Line 3 – The zone path resource, /export/work-zone in this example, is the path to the zone root. Each zone has a path to its root directory that is relative to the global zone’s root directory. This path must exist at installation time. The global zone directory is required to have restricted visibility. It must be owned by root with the mode 700. In this example the global zone directory is /export.

Line 4 – This indicates that a zone should be booted automatically at system boot.

Line 5 – This line begins a file system configuration. The command scope changes to file systems.

Line 6 – Set the mount point for the file system, /mnt in this example.

Line 7 – Specify that /dev/dsk/c0t0d0s7 block special file in the global zone is to be mounted as /mnt in the work-zone.

Line 8 – Specify that /dev/rdsk/c0t0d0s7 raw special file. The zoneadm daemon automatically runs the fsck command in non-interactive check only mode on this device before it mounts the file system.

Line 9 – This line specifies that the file system type is UFS.

Line 10 – This line specifies the file system-specific option, enable file system logging in this procedure.

Line 11 – This line ends the file system configuration section in this procedure.

Line 12 – This line begins the configuration of a read only shared file system that is loopback-mounted from the global zone.

Line 13 – This line specifies that /usr/sfw is to be loopback mounted from the global zone.

Line 14 – This line ends the mount loopback section in this procedure.

Line 15 – This line begins the network configuration section in this procedure.

Line 16 – This line specifies the physical network interface to be used by this zone is a GigaSwift.

Line 17 – This line specifies the IP address for the network interface, 192.168.0.1 in this procedure.

Line 18 – This line ends the network configuration section in this procedure.

Line 19 – This line begins the device configuration section in this procedure.

Line 20 – This line gives the non-global zone visibility to devices that match the pattern /dev/sound/* in the global zone.

Line 21 – This line ends the device configuration section in this procedure.

Line 22 – This line begins the attribute configuration section in this procedure.

Line 23 – This line sets the name of the name of the attribute, comment in this procedure.

Line 24 – This line sets the type of attribute as a string of characters.

Line 25 – This line assigns a value to the string of characters, “The work zone.” in this procedure.

Line 26 – This line ends the attribute configuration section in this procedure.

Line 27 – This line verifies the current configuration for correctness. It ensure that all resources have all of their required properties specified.

Line 28 – This line commits the current configuration from memory to stable storage. Until the in-memory configuration is committed, changes can be removed with the revert subcommand. A configuration must be committed to be used by the zoneadm command. This operation is attempted automatically when you complete a zonecfg session. Because only a correct configuration can be committed, the commit operation automatically does a verify.

Line 29 – This line exits the zonecfg session. You can use the -F (force) option with exit.

The zone is now ready to install, boot, and use.

Viewing the Zone Configuration

You can use the zonecfg command to view the zone configuration.

```
# zonecfg -z work-zone info
zonepath: /export/work-zone
autoboot: true
inherit-pkg-dir:
    dir: /lib
inherit-pkg-dir:
    dir: /platform
inherit-pkg-dir:
    dir: /sbin
inherit-pkg-dir:
    dir: /usr
inherit-pkg-dir:
    dir: /opt/sfw
fs:
    dir: /mnt
    special: /dev/dsk/c0t0d0s7
    raw: /dev/rdsck/c0t0d0s7
    type: ufs
    options: [logging]
net:
    address: 192.168.0.1
    physical: ce0
device
    match: /dev/sound/*
attr:
    name: comment
    type: string
    value: "The work zone."
#
```

When you commit the zone configuration to stable storage, the file is stored in the /etc/zones directory in XML format.



Caution – The /etc/zones file is never to be edited. This file exists for implementation purposes only.

For example:

```
# more /etc/zones/work-zone.xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE zone PUBLIC "-//Sun Microsystems Inc//DTD Zones//EN"
 "file:///usr/share/lib/xml/dtd/zo
necfg.dtd.1">
<zone name="work-zone" zonepath="/export/work-zone" autoboot="true">
...
...
```

Using the zoneadm Command

The zoneadm command is the primary tool used to install and administer non-global zones. Operations using the zoneadm command must be run from the global zone. The following tasks can be performed using the zoneadm command:

- Verify a configured zone
- Install a configured zone
- Boot a zone
- Halt a zone
- Reboot a zone
- Log into and work with a running zone
- Move a zone
- Migrate a zone
- Delete a zone

Verifying a Configured Zone

You can verify a zone prior to installing it. If you skip this procedure, the verification is performed automatically when you install the zone. You must be the global administrator in the global zone to perform this procedure.

You use the zoneadm -z *zone_name* verify command to verify a zone's configuration. For example:

```
global# zoneadm -z work-zone verify
```

Warning: /export/work-zone does not exist, so it cannot be verified. When zoneadm install is run, install will try to create /export/work-zone, and verify will be tried again, but the verify may fail if: the parent directory of /export/work-zone is group- or other-writable or /export/work-zone overlaps with any other installed zones.

In this example, a message is displayed warning the administrator that the zonepath does not exist. This illustrates the type of messages output by the zoneadm command.

If no error messages are displayed, you can install the zone.

Installing a Configured Zone

You use the `zoneadm -z zone_name install` command to perform installation tasks for a non-global zone. You must be the global administrator to perform the zone installation. For example:

```
global# zoneadm -z work-zone install
```

You use the `zoneadm list -iv` command to list the installed zones and verify the status:

```
global# zoneadm list -iv
ID  NAME      STATE     PATH
0   global    running   /
-   work-zone installed /export/work-zone
```

In this example, the work-zone has reached the installed state. The zone ID will be assigned during the zone boot process.

 **Note** – By default, `list` only shows running zones. To show all zones on a system you would run `zoneadm list -cv`.

Booting a Zone

Booting a zone places the zone in the running state. If you set the `autoboot` resource property in a zones configuration to `true`, that zone is automatically booted when the global zone is booted. The default setting is `false`.

 **Note** – Before a zone is booted for the first time you should connect to the zone console device using `zlogin -C` to enter the system configuration information required to use the zone.

A zone can be manually booted from the installed state. You use the `zoneadm -z zone_name boot` command to boot a zone:

```
global# zoneadm -z work-zone boot
global# zoneadm list -v
ID  NAME      STATE     PATH
0   global    running   /
1   work-zone running  /export/work-zone
```

In this example, the work-zone has reached the running state. The zone ID 1 has been assigned during the zone boot process.

Solaris Zones Boot Argument Enhancements

The following zones boot arguments are now supported as part of zoneadm boot and reboot commands.

- **-i altinit** - Selects an alternative executable to be the first process. The altinit must be a valid path to an executable.
- **-m smf_options** - Controls the boot behavior of SMF. There are two categories of options, services options and messages options.
 - Messages options: Determines the type and number of messages that displays during boot. For example:
 - **debug** - Prints standard per-service output and all svc.startd messages to log.
 - **quiet** - Prints standard per-service output and error messages requiring administrative intervention.
 - **verbose** - Prints standard per-service output and messages providing more information.
 - Service options determine the services that are used to boot the system.
 - **-s** - Boots only to milestone svc:/milestone/single-user:default. This milestone is equivalent to init level s.

Halting a Zone

The zoneadm halt command is used to remove both the application environment and the virtual platform for a zone. The zone is then brought back to the installed state. All processes are killed, devices are unconfigured, network interfaces are unplumbed, file systems are unmounted, and the kernel data structures are destroyed.

```
global# zoneadm -z work-zone halt
global# zoneadm list -v
ID  NAME      STATE     PATH
0   global    running    /
-   work-zone  installed /export/work-zone
```

The halt command does not run any shutdown scripts within the zone. The root user of a zone may shut down the zone when logged into the zone and run any shutdown scripts.

Rebooting a Zone

The zoneadm reboot command is used to reboot a zone. The zone is halted and then booted again.

```
global# zoneadm -z work-zone reboot
global# zoneadm list -v
ID  NAME      STATE     PATH
0   global    running   /
2   work-zone running   /export/work-zone
```

In this example, before rebooting the zone ID is set to 1. After the zone is rebooted, the zone ID has changed to 2.

Logging Into and Working With The Zone

Use the zlogin command to log in to and access the deployed zone from the global zone. Be aware that root users are not allowed to log in by default. To log into the zone as if you were on its console use the -C option.

```
# zlogin -C work-zone
[Connected to zone 'work-zone' console]
```

The first time the zone boots after an install, you are asked to provide a terminal type, host name, time zone, and root password.



Note – If using a CDE terminal window, choose dtterm. If using another type of window, choose vt100 or xterm.

After you enter the appropriate information, you see the following output:

```
System identification is completed.
rebooting system due to change(s) in /etc/default/init
[NOTICE: zone rebooting]
SunOS Release 5.10 Version s10 64-bit
Copyright 1983-2004 Sun Microsystems, Inc. All rights reserved.
```

Using the zoneadm Command

Use is subject to license terms.

Hostname: twilight

The system is coming up. Please wait.

starting rpc services: rpcbind done.

syslog service starting.

Creating new rsa public/private host key pair

Creating new dsa public/private host key pair

The system is ready.

twilight console login: **root**

Password:

Dec 16 12:37:07 twilight login: ROOT LOGIN /dev/console

Sun Microsystems Inc. SunOS 5.10 s10 Dec 2004

After using the console interface to log into the zone, take a look at how the operating system views its resources.

twilight# **hostname**

twilight

twilight# **uname -a**

SunOS twilight 5.10 s10 sun4u sparc SUNW,Netra-T12

twilight# **df -k**

| File system | kbytes | used | avail | capacity | Mounted on |
|-------------|----------|---------|----------|----------|-------------|
| / | 678457 | 69941 | 547455 | 12% | / |
| /dev | 678457 | 69941 | 547455 | 12% | /dev |
| /lib | 33265565 | 1893804 | 31039106 | 6% | /lib |
| /platform | 33265565 | 1893804 | 31039106 | 6% | /platform |
| /sbin | 33265565 | 1893804 | 31039106 | 6% | /sbin |
| /usr | 33265565 | 1893804 | 31039106 | 6% | /usr |
| proc | 0 | 0 | 0 | 0% | /proc |
| mnttab | 0 | 0 | 0 | 0% | /etc/mnttab |
| fd | 0 | 0 | 0 | 0% | /dev/fd |
| swap | 7949040 | 32 | 7949008 | 1% | /var/run |
| swap | 7949008 | 0 | 7949008 | 0% | /tmp |

twilight# **ps -ef | grep z**

| UID | PID | PPID | C | STIME | TTY | TIME | CMD |
|------|------|------|---|----------|-----|------|--------|
| root | 6965 | 6965 | 0 | 12:35:38 | ? | 0:00 | zsched |

twilight# **ifconfig -a**

lo0:1: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1

inet 127.0.0.1 netmask ff000000

ce0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2

inet 192.168.0.1 netmask ffffff00 broadcast 192.168.0.255

twilight# ~.

[Connection to zone 'work-zone' console closed]



Note – The zone is now up and running. If you add (or delete) resources to the running zone using the zonecfg command, you must restart the zone for the changes to take effect.

To remove a resource from a non-global zone, run the zonecfg command and choose the remove subcommand with a reference to the device and parameters.

```
# zonecfg -z work-zone
zonecfg:work-zone> remove net physical=ce0
zonecfg:work-zone> commit
zonecfg:work-zone> exit
```

Moving a Zone

The move zone feature allows you to relocate a non-global zone from one point on a system to another point on the same system. This is implemented so that it works both within and across file systems, subject to the existing rules for zonepath. When crossing file system boundaries, the data is copied and the original directory is removed. Internally, the copy is implemented using the cpio command with the proper options to preserve all of the data such as ACLs.

To move a non-global zone, use the zoneadm command. For example:

```
# zoneadm -z work-zone move /newpath
```

In this example, /newpath specifies the new zonepath for the zone.

Note – The non-global zone must be halted while being moved.



Migrating a Zone

Using the `zonecfg` and `zoneadm` commands, you can migrate a non-global zone from one system to another. This procedure detaches a halted zone from its current location, and attaches the zone at a new location. The global zone on the target system must be running the following:

- The same release as the original host
- The same versions of operating system packages and patches as the original host

The zone detach process creates the information necessary to attach the zone on a different system. The zone attach process verifies that the new machine has the correct configuration to host the zone. You can make the `zonepath` available on the new host in several ways. Therefore, the actual movement of the `zonepath` from one system to another is a manual process that is performed by the zone administrator.

Migrating a zone from one system to another involves the following steps:

1. Detaching the Zone – This leaves the zone on the originating system in the *configured* state. Behind the scenes, the system generates a manifest of the information needed to validate that the zone can be successfully attached to a new host machine.

To detach a zone, first halt the zone, then perform the detach operation:

```
host1# zoneadm -z work-zone halt
host1# zoneadm -z work-zone detach
```



Note – The detach operation generates metadata describing the versions of the packages and patches installed on the host. This is stored in an XML file in the `zonepath`, alongside the `root` and `dev` directories. This facilitates easy movement of the `zonepath` from one system to another.

2. Data Migration – You move the data which represents the zone to a new host system. The following is an example of migration zone data from host1 to host2.

On host1:

```
host1# cd /export/zones
host1# tar cf work-zone.tar work-zone
host1# sftp host2
Connecting to host2...
```

Password:

```
sftp> cd /export/zones
sftp> put work-zone.tar
```

Uploading work-zone.tar to /export/zones/work-zone.tar
sftp> quit

On host2:

```
host2# cd /export/zones
host2# tar xf my-zone.tar
```

3. Zone Configuration – You create the zone configuration on the new host using the zonecfg command. For example:

```
host2# zonecfg -z work-zone
work-zone: No such zone configured
```

Use the create subcommand to begin configuring a new zone.

```
zonecfg:work-zone> create -a /export/zones/work-zone
```



Note – The -a option uses the XML description of the detached zone (created during detach operation) to configure the new zone instance. The path /export/zones/work-zone is the path to the zone root. Alternately, the new zone can be configured using the traditional zonecfg operations and then zoneadm attach can be used to attach the zone root.



Note – Be sure to make any required adjustments to the configuration. For example, the network physical device might be different on the new host:
zonecfg:work-zone> select net physical=bge0
zonecfg:work-zone:net> set physical=e1000g0

```
zonecfg:work-zone> commit
```

```
zonecfg:work-zone> exit
```

4. Attaching the zone – This validates that the host is capable of supporting the zone before the attach can succeed. The zone is left in the *installed* state.

The syntax for attaching a zone is:

```
host2# zoneadm -z work-zone attach
```

Pre-Validating Zone Migration

You can perform a “dry-run” before the zone is moved to the new system by using the `-n` option of the `zoneadm detach` command.

The `zoneadm detach -n` command generates a manifest on a running zone without actually detaching the zone. The state of the zone on the originating system is not changed. The zone manifest is sent to the standard output. The global administrator can direct this output to a file or pipe it to a remote command to be immediately validated on the target host.

You can use the `zoneadm attach -n` command to read this manifest and verify that the target system has the correct configuration to host the zone without actually doing an attach.

To pre-validate a zone migration, use the following methods:

- Generate the manifest on the source host:

```
# zoneadm -z twilight-zone detach -n
```

The zone manifest is sent to the standard output. You can also direct the zone manifest to a file, or pipe it to a remote command that will immediately validate the target host:

```
# zoneadm -z twilight-zone detach -n | ssh remotehost  
zoneadm attach -u -n -
```

- If a remote command is not piped in, you can copy the zone manifest to the target system and pre-validate it there as follows:

```
# zoneadm attach -u -n path_to_manifest
```

Deleting a Zone

When deleting a zone, be sure to back up any files that you want to keep. The first stage in deleting a zone is halting the Solaris 10 OS and freeing the system memory.

In the following example, the zone is removed from the global system:



Caution – This operation is not a graceful or controlled shutdown of the zone. Data loss is possible to processes running in the zone.

```
# zoneadm list -cp
0:global:running:/
3:work-zone:running:/export/work-zone
# zoneadm -z work-zone halt
# zoneadm list -cp
0:global:running:/
-:work-zone:installed:/zones/work-zone
```

At this point, the zone is not using system resources other than file system space. Uninstall the zone to remove the zone's file usage.

```
# zoneadm -z work-zone uninstall
Are you sure you want to uninstall zone work-zone (y/[n])? y
# zoneadm list -cp
0:global:running:/
-:work-zone:configured:/export/work-zone
```

The final step is to delete the configuration of the zone from the global system with the delete subcommand.

```
# zonecfg -z work-zone delete
Are you sure you want to delete zone work-zone (y/[n])? y
# zoneadm list -cp
0:global:running:/
```

Installing Packages in Zones

The standard Solaris package management tools, for example, pkgadd and pkgrm, are used to administer packages on a system with zones installed. The global administrator can use these tools to manage the software on every zone in the system.

Package parameters listed in the `pkginfo` file for a package control how the Solaris package tools can administer the package. These package parameters determine how package content can be distributed and made visible among zones, both global and non-global, in a system.

Currently, three package parameters control how packages are administered. They are:

- `SUNW_PKG_ALLZONES` – Defines the zone *scope* of a package. The scope determines the type of zone in which an individual package can be installed.
- `SUNW_PKG_HOLLOW` – Defines the *visibility* of a package if that package is required to be installed on all zones and be identical in all zones (for example, a package that has `SUNW_PKG_ALLZONES=true`).
- `SUNW_PKG_THISZONE` – Defines whether a package must be installed in the current zone only.

Values of these parameters can only be set to true or false. If one of these parameters is not defined in a package, the package management tools assume its value to be false. More information about the specific effects of these parameters and how they interact is available in `pkginfo` (4).

Packaging for Sparse and Whole Root Zones

Sparse root zones are much more restrictive with package contents than whole root zones. For sparse root zones, any packages that deliver files into `/usr`, `/lib`, `/sbin`, or `/platform` (or any of the sub-directories) must be installed in the global and all non-global zones.

If more flexibility is required with what packages are required within a zone, you may want to use full root zones. This would allow for installing applications into non-global zones which write files into `/usr`, `/lib`, `/sbin`, or `/platform`.

Listing Parameters for Packages

You can list parameters for packages using the `pkgparam` command. To display the list of parameters and their values in a package, use `pkgparam -v package`. For example:

```
# pkgparam -v SUNWzoneu
CLASSES='none'
BASEDIR='/'
LANG='C'
PATH='/sbin:/usr/sbin:/usr/bin:/usr/sadm/install/bin'
OAMBASE='/usr/sadm/sysadm'
PKG='SUNWzoneu'
NAME='Solaris Zones (Usr)'
ARCH='sparc'
VERSION='11.10.0,REV=2005.01.21.15.53'
SUNW_PRODNAME='SunOS'
SUNW_PRODVERS='5.10/Generic'
SUNW_PKGTYPE='usr'
MAXINST='1000'
CATEGORY='system'
DESC='Solaris Zones Configuration and Administration'
VENDOR='Sun Microsystems, Inc.'
HOTLINE='Please contact your local service provider'
EMAIL=''
SUNW_PKGVERS='1.0'
SUNW_PKG_ALLZONES='true'
SUNW_PKG_HOLLOW='false'
PSTAMP='gaget20050121155950'
PKGINST='SUNWzoneu'
PKGSAV=' /var/sadm/pkg/SUNWzoneu/save'
INSTDATE='Jan 26 2005 10:21'
#
```

The `pkgadd` utility can be used with the `-G` option in the global zone to add the package to the global zone only. The package is not propagated to any other zones. Note that if `SUNW_PKG_THISZONE=true`, you do not have to use the `-G` option. If `SUNW_PKG_THISZONE=false`, the `-G` option will override it.

When you run the pkgadd utility in the global zone, the following actions apply.

- The pkgadd utility is able to add a package:
 - To the global zone only, unless the package is `SUNW_PKG_ALLZONES=true`
 - To the global zone and to all non-global zones
 - To all non-global zones only, if the package is already installed in the global zone
 - To the current zone only, if `SUNW_PKG_THISZONE=true`
- The pkgadd utility cannot add a package:
 - To any subset of the non-global zones
 - To all non-global zones, unless the package is already installed in the global zone
- If the pkgadd utility is run without the `-G` option and `SUNW_PKG_THISZONE=false`, the specified package is added to all zones by default. The package is not marked as installed in the global zone only.
- If the pkgadd utility is run without the `-G` option and `SUNW_PKG_THISZONE=true`, then the specified package is added to the current (global) zone by default. The package is marked as installed in the global zone only.
- If the `-G` option is used, the pkgadd utility adds the specified package to the global zone only. The package is marked as installed in the global zone only. The package is not installed when any non-global zone is installed.

Package Operations Possible in the Global Zone

If the package is not currently installed in the global zone and not currently installed in any non-global zone, the package can be installed according to the following guidelines:

- Only in the global zone, if `SUNW_PKG_ALLZONES=false`
- In the current zone only, which is the global zone in this case, if `SUNW_PKG_THISZONE=true`
- In the global zone and all non-global zones

If the package is currently installed in the global zone only, the following guidelines apply:

- The package can be installed in all non-global zones.
- The package can be removed from the global zone.

If a package is currently installed in the global zone and currently installed in only a subset of the non-global zones, the following guidelines apply:

- `SUNW_PKG_ALLZONES` must be set to false.
- The package can be installed in all non-global zones. Existing instances in any non-global zone are updated to the revision being installed.
- The package can be removed from the global zone.
- The package can be removed from the global zone and from all non-global zones.

If a package is currently installed in the global zone and currently installed in all non-global zones, the package can be removed from the global zone and from all non-global zones.

These rules ensure the following:

- Packages that are installed in the global zone are either installed in the global zone only, or installed in the global zone and all non-global zones.
- Packages that are installed in the global zone and also installed in any non-global zone are the same across all zones.

Package Operations Possible in a Non-Global Zone

The package operations possible in any non-global zone are the following:

- If a package is not currently installed in the non-global zone, the package can be installed only if `SUNW_PKG_ALLZONES=false`.
- The package can be installed in the current zone, which is the non-global zone in this case, if `SUNW_PKG_THISZONE=true`.



- If a package is currently installed in the non-global zone, the following guidelines apply:
 - The package can be installed over the existing instance of the package only if `SUNW_PKG_ALLZONES=false`.
 - The package can be removed from the non-global zone only if `SUNW_PKG_ALLZONES=false`.

Note – For more information about managing packages and patches on Solaris systems with zones installed, see Chapter 23, *About Packages and Patches on a Solaris System with Zones Installed (Overview)* in the *Zones* section of the *System Administration Guide: Solaris Containers-Resource Management and Solaris Zones* found on <http://docs.sun.com/app/docs/doc/817-1592/6mhahuor0?a=view>.

Using 1x Branded Zone

These are zones that contain non-native operating environments. The term non-native is intentionally vague, as the infrastructure allows for the creation of a wide range of operating environments, such as Linux, FreeBSD, or older versions of Solaris.

Each type of environment supported is referred to as a *brand*. A brand is an attribute of a non-global zone set at zone creation time, and each brand provides zone install and boot scripts. Every zone is configured with a brand, with the default being *native*. The brand type is used to determine which scripts are executed at install and boot time and to identify the correct application type at launch time. Once a zone has been assigned a brand, it cannot be changed or removed.

Note – The Solaris 10 8/07 release includes the framework for a single brand, 1x, which enables Linux binary applications to run unmodified on Solaris x86 and x64-based systems. The 1x brand comprises the user-visible feature referred to as the Linux Application Environment (LAE). This combination of BrandZ and the 1x brand form the product, *Solaris Containers for Linux Applications*.

Planning for 1x Branded Zone

The following section describes the requirements, dependencies, and limitations of using 1x branded zone.

Supported Processor Types

The machine running the 1x branded zone must have one of the following i686 processor types:

| Intel | AMD |
|--------------------------------|-----------------------|
| Pentium Pro/II/III | Opteron |
| Celeron | Athlon XP/64/64 X2/FX |
| Xeon | Duron |
| Pentium 4/M/D/ Extreme Edition | Sempron |
| Core/ Core 2 | Turion 64/64 X2 |

Supported Linux Distributions

The `lx` branded zone supports the installation of CentOS 3.x or Red Hat Enterprise Linux 3.x distribution inside a non-global zone. Before you can install the `lx` branded zone, you must first obtain the Linux archives. The archives are distributed in the following forms:

- A compressed tar archive
- A set of CD-ROM or DVD discs
- A group of ISO images

You can obtain the Linux distribution using one of the following methods:

- Download a compressed tar archive at
<http://opensolaris.org/os/community/brandz/downloads>
- Obtain a set of CD-ROM or DVD discs from CentOS site at
<http://www.centos.org> or the Red Hat site at
<http://www.redhat.com>
- Obtain an ISO image from the CentOS site at
<http://www.centos.org> or the Red Hat site at
<http://www.redhat.com>

Application Support

The Solaris system imposes no limit on the number of Linux applications you can run in an `lx` branded zone. Regardless of the underlying kernel, only 32-bit Linux applications are able to run. You should ensure that sufficient disk is made available to hold files that are unique with each `lx` branded zone. The disk space requirements for an `lx` branded zone are determined by the size and number of Linux packages that you plan to install.

The `lx` zone supports only user-level Linux applications. You cannot use Linux device drivers, Linux kernel modules, or Linux file systems from inside an `lx` zone. Further, you cannot run Solaris applications inside an `lx` zone. However, the `lx` zone enables you to use the Solaris system to develop, test, and deploy Linux applications. For example, you can place a Linux application in an `lx` zone and analyze it using Solaris tools run from the global zone. You can then make improvements and deploy the tuned application on a native Linux system.

Debugging Tools

You can apply Solaris debugging tools, such as dtrace and mdb to Linux processes executing inside the zone, but the tools themselves must be running in the global zone. Any core files generated are produced in the Solaris format and can only be debugged with Solaris tools.

Installing and Configuring an 1x Branded Zone

The typical installation and configuration of an 1x branded zone involves the following tasks:

1. Create an 1x brand zone using the zonecfg command.
2. Install and boot the 1x zone using the zoneadm command.
3. Log on to the 1x zone using the zlogin command.
4. Enable networking in the 1x zone by editing the system configuration file in the zone.

Exercise 1: Configuring Zones

In this exercise, you will perform the following tasks:

- Create and install a non-global zone
- Administer users and data within zones
- Install software packages within zones
- Remove zones

Preparation

This exercise requires software packages and programs located in /opt/ses/lab/sparc/zones or /opt/ses/lab/x86/zones, for SPARC or x86/x64 architecture systems, respectively.

If you are on an x86-based machine, you use the following packages:

- The top-3.5.1-sol10-zones-intel-local package
- The sudo-1.6.8p9-sol10-zones-intel-local package

If you are on a SPARC-based machine, you use the following packages:

- The top-3.5.1-sol10-zones-sparc-local package
- The sudo-1.6.8p4-sol10-zones-sparc-local package

Remove any user-defined non-root users from the system that may exist because of previous lab exercises.

```
# userdel -r user9
```

Remove the SUNWoptdir and SUNWusrdir packages from the system that may exist because of previous lab exercises.

```
# pkgrm SUNWoptdir  
# pkgrm SUNWusrdir
```

This exercise requires a system that has a spare disk available. During the exercise, you will use the format utility to create three partitions on the spare disk; two 1 Gbyte partitions and one partition representing the remainder of the disk will be required. To complete this exercise you must already be familiar with the format utility.

Your instructor will provide an IP address, for the non-global zone you will create.

Use the `ifconfig -a` command to determine what device is used as the primary Ethernet interface on your system.

Task 1 – Creating and Installing a Non-Global Zone

Perform the following steps to create a non-global zone called `zone1`:

1. Use the `format` utility to create three partitions on the spare disk in your system. Set partitions 0 and 1 to 1Gbyte each, and set partition 6 to use the remainder of the disk. The ‘all free hog’ method available in the partition menu of the `format` utility may offer the easiest method.
2. Use `newfs` to create a UFS file system on each of the partitions 0 and 1 you created. Replace the example slice names with those that are correct for your system.
3. Create a directory `/export/zone1`. The files specific to the non-global zone you will create will be installed in file system you will mount below this directory.
4. Verify that the `/export` directory is owned by the `root` user and group.
5. Edit the `/etc/vfstab` file on your system and add an entry so that the filesystem on partition 0 on your spare disk will mount automatically below `/export/zone1`. Use device names that are correct for your system.
6. Use the `mount` command to mount the new file system.
7. Change the permission mode of `/export/zone1` to 700.
8. Create a non-global zone called `zone1`. In the `special` and `raw` properties, specify the special and raw device file names for slice 1 on your spare disk.

Specify the device used as the primary Ethernet interface on your system in the `physical` property. Specify the IP address supplied by your instructor in the `address` property.

9. Use the `zoneadm` command to verify that no non-global zone is currently installed.
10. Open a terminal window. Use the window to install `zone1`. Zone installation takes different amounts of time to complete, depending on your system’s capabilities.

Exercise 1: Configuring Zones

The zoneadm command checks if the zone configuration is valid, and reports errors that it finds. Messages expected from zoneadm include the following:

Preparing to install zone <zone1>.
Checking <ufs> file system on device </dev/rdsk/c2d0s1> to be mounted at </export/zone1/root>
Creating list of files to copy from the global zone.
Copying <2422> files to the zone.
Initializing zone product registry.
Determining zone package initialization order.
Preparing to initialize <974> packages on the zone.
Initializing package <243> of <974>: percent complete: 24%

11. After the installation processes for the non-global zone completes, use the zoneadm command to verify the status of the zone.
12. Verify the disk space used below /export by the zone.
13. Verify your system's current network interface configuration. Note that no virtual interface has yet been configured for the zone.
14. Use the zlogin -C command before booting the zone.
15. Boot zone1. The initial zone boot process can take a few minutes to complete.
16. Verify that zone1 is running.
17. Verify your system's current network interface configuration. Note that a virtual interface has now been configured for the zone. Verify that zone1 is using the IP address that you assigned to it.
18. Use zlogin -C to connect to the console of zone1, and respond to the system identification questions that are presented. Be certain to select a terminal type that is appropriate for your system.

Note – If you are connected to your system remotely, it may be useful to specify an alternate escape character for zlogin console connections. Do this to avoid inadvertently disconnecting your remote connection. For example, to use the caret symbol instead of the tilde symbol (the default), you would use: zlogin -C -e \^ zone1

After you choose the language, locale, and terminal type appropriate for your system, an interactive GUI utility prompts you for the remaining required information. After you enter this information, the zone reboots automatically.

Host name for eri0:n = **zone1**



Configure kerberos security = **no**
Name service = **none**
Use the NFSv4 domain derived by the system = <**select**>
Default time zone = <**location specific**>
Root password = **cangetin**

19. Exit the zlogin console session. If you specified an alternate escape character in your zlogin command line, enter it instead of the tilde character.

Task 2 – Administering Users and Data Within Zones

Perform the following steps to verify how user and data resources are distinct among zones:

1. Verify that the non-global zone has mounted the file system you specified in its configuration steps, and that you can create a file within that file system.
2. Log in to zone1 and create a user called user1.
3. Verify that user1 exists in the /etc/passwd file in zone1, and then log out of zone1.
4. Check if user1 is listed in the /etc/passwd file of the global zone.

Note – The user called user1 will exist only in zone1.



Task 3 – Installing Software Packages in Zones

Perform the following steps to install packages in zones:

1. In the global zone on your system, change to the /opt/ses/lab/sparc/zones or /opt/ses/lab/x86/zones directory, depending on your system architecture. Here you will find the packages for the top and sudo applications. Ask your instructor if you need help.

If you are on an x86-based machine, you use the following packages:

- The top-3.5.1-sol10-zones-intel-local package
- The sudo-1.6.8p9-sol10-zones-intel-local package

Exercise 1: Configuring Zones

If you are on a SPARC-based machine, you use the following packages:

- The top-3.5.1-sol10-zones-sparc-local package
 - The sudo-1.6.8p4-sol10-zones-sparc-local package
2. Use pkginfo to display the names of the packages contained in the package files.
 3. Use the pkgparam command to display the value of the SUNW_PKG_ALLZONES parameter for the SMCsudo and SMCTop packages. If the SUNW_PKG_ALLZONES parameter is not configured, the result is equal to the parameter being set to false.



Note – When set to false, the SUNW_PKG_ALLZONES parameter determines that the package can be installed: 1) in the global zone only, or 2) in the global zone and all non-global zones, or 3) from any non-global zone to the same non-global zone.

4. Install the SMCTop package in the global zone only. All files for this package are installed below the /opt/local directory.
5. Log in to zone1 and verify that the SMCTop package was not installed in that zone. Log out from zone1 when finished.
6. From the global zone, install the SMCsudo package in the global zone and the non-global zone.
7. Log in to zone1 and verify that the SMCsudo package has been installed. Log out from zone1 when finished.

Task 4 - Removing Zones

1. From the global zone list the current configured zones with zoneadm command.
2. Use the zoneadm command to halt any running non global zones.
3. Use the zoneadm command to uninstall zone1 and zone2 .
4. Delete the configuration of zone1 and zone2 from the global system.
5. Umount the /export/zone1 filesystem and remove the entry for /export/zone1 from /etc/vfstab.

Exercise Summary



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

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise 1 Solutions: Configuring Zones

In this exercise, you will perform the following tasks:

- Create and install a non-global zone
- Administer users and data within zones
- Install software packages within zones
- Remove zones

Preparation

This exercise requires software packages and programs located in /opt/ses/lab/sparc/zones or /opt/ses/lab/x86/zones, for SPARC or x86/x64 architecture systems, respectively.

If you are on an x86-based machine, you use the following packages:

- The top-3.5.1-sol10-zones-intel-local package
- The sudo-1.6.8p9-sol10-zones-intel-local package

If you are on a SPARC-based machine, you use the following packages:

- The top-3.5.1-sol10-zones-sparc-local package
- The sudo-1.6.8p4-sol10-zones-sparc-local package

To complete this exercise you must already be familiar with the format utility.

Remove any user-defined non-root users from the system that may exist because of previous lab exercises.

```
# userdel -r user9
```

Remove the SUNWoptdir and SUNWusrdir packages from the system that may exist because of previous lab exercises.

```
# pkgrm SUNWoptdir  
# pkgrm SUNWusrdir
```

This exercise requires a system that has a spare disk available. During the exercise, you will use the format utility to create three partitions on the spare disk; two 1 Gbyte partitions and one partition representing the remainder of the disk will be required.

Your instructor will provide an IP address, for the non-global zone you will create.

Use the ifconfig -a command to determine what device is used as the primary Ethernet interface on your system.

Task 1 – Creating and Installing a Non-Global Zone

Perform the following steps to create a non-global zone called zone1:

1. Use the format utility to create three partitions on the spare disk in your system. Set partitions 0 and 1 to 1Gbyte each, and set partition 6 to use the remainder of the disk. The ‘all free hog’ method available in the partition menu of the format utility may offer the easiest method.
2. Use newfs to create a UFS file system on each of the partitions 0 and 1 you created. Replace the example slice names with those that are correct for your system.

```
# newfs /dev/rdsk/c2d0s0
# newfs /dev/rdsk/c2d0s1
```

3. Create a directory /export/zone1. The files specific to the non-global zone you will create will be installed in file system you will mount below this directory.

```
# mkdir /export/zone1
```

4. Verify that the /export directory is owned by the root user and group.

```
# chown root:root /export
# ls -ld /export
```

```
drwxr-xr-x 5 root      root          512 Apr 30 13:52 /export
```

5. Edit the /etc/vfstab file on your system and add an entry so that the filesystem on partition 0 on your spare disk will mount automatically below /export/zone1. Use device names that are correct for your system. For example:

```
/dev/dsk/c2d0s0 /dev/rdsk/c2d0s0      /export/zone1    ufs      2      yes      -
```

6. Use the mount command to mount the new file system.

```
# mount /export/zone1
```

7. Change the permission mode of /export/zone1 to 700.

```
# chmod 700 /export/zone1
```

Exercise 1 Solutions: Configuring Zones

8. Create a non-global zone called zone1. In the special and raw properties, specify the special and raw device file names for slice 1 on your spare disk.

Specify the device used as the primary Ethernet interface on your system in the physical property. Specify the IP address supplied by your instructor in the address property.

```
# zonecfg -z zone1
zone1: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:zone1> create
zonecfg:zone1> set zonepath=/export/zone1
zonecfg:zone1> set autoboot=true
zonecfg:zone1> add fs
zonecfg:zone1:fs> set dir=/dir1
zonecfg:zone1:fs> set special=/dev/dsk/c2d0s1
zonecfg:zone1:fs> set raw=/dev/rdsck/c2d0s1
zonecfg:zone1:fs> set type=ufs
zonecfg:zone1:fs> end
zonecfg:zone1> add net
zonecfg:zone1:net> set physical=nge0
zonecfg:zone1:net> set address=10.7.10.114
zonecfg:zone1:net> end
zonecfg:zone1> add attr
zonecfg:zone1:attr> set name=comment
zonecfg:zone1:attr> set type=string
zonecfg:zone1:attr> set value="zone one"
zonecfg:zone1:attr> end
zonecfg:zone1> verify
zonecfg:zone1> commit
zonecfg:zone1> info
zonename: zone1
zonepath: /export/zone1
autoboot: true
pool:
limitpriv:
inherit-pkg-dir:
    dir: /lib
inherit-pkg-dir:
    dir: /platform
inherit-pkg-dir:
    dir: /sbin
inherit-pkg-dir:
    dir: /usr
fs:
    dir: /dir1
```

```

special: /dev/dsk/c2d0s1
raw: /dev/rdsck/c2d0s1
type: ufs
options: []
net:
address: 10.7.10.114
physical: nge0
attr:
name: comment
type: string
value: "zone one"
zonecfg:zone1> exit
#

```

9. Use the zoneadm command to verify that no non-global zone is currently installed.

```
# zoneadm list -i
global
#
```

10. Open a terminal window. Use the window to install zone1. Zone installation takes different amounts of time to complete, depending on your system's capabilities.

```
# zoneadm -z zone1 install
```

The zoneadm command checks if the zone configuration is valid, and reports errors that it finds. Messages expected from zoneadm include the following:

```

Preparing to install zone <zone1>.
Checking <ufs> file system on device </dev/rdsck/c2d0s1> to be mounted at
</export/zone1/root>
Creating list of files to copy from the global zone.
Copying <2422> files to the zone.
Initializing zone product registry.
Determining zone package initialization order.
Preparing to initialize <974> packages on the zone.
Initializing package <243> of <974>; percent complete: 24%

```

11. After the installation processes for the non-global zone completes, use the zoneadm command to verify the status of the zone.

```
# zoneadm list -iv
ID NAME      STATUS     PATH          BRAND    IP
  0 global    running    /
  - zone1    installed   /export/zone1 native   shared
#

```

Exercise 1 Solutions: Configuring Zones

12. Verify the disk space used below /export by the zone.

```
# du -sk /export/zone*
```

58525 /export/zone1

#

13. Verify your system's current network interface configuration. Note that no virtual interface has yet been configured for the zone.

```
# ifconfig -a
```

lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
 inet 127.0.0.1 netmask ff000000
nge0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
 inet 10.7.10.14 netmask ffff0000 broadcast 10.7.255.255
 ether 0:e0:81:73:52:cc

#

14. Boot zone1. The initial zone boot process can take a few minutes to complete.

```
# zoneadm -z zone1 boot
```

15. Verify that zone1 is running.

```
# zoneadm list -iv
```

| ID | NAME | STATUS | PATH | BRAND | IP |
|----|--------|---------|---------------|--------|--------|
| 0 | global | running | / | native | shared |
| 2 | zone1 | running | /export/zone1 | native | shared |

#

16. Verify your system's current network interface configuration. Note that a virtual interface has now been configured for the zone. Verify that zone1 is using the IP address that you assigned to it.

```
# ifconfig -a
```

lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
 inet 127.0.0.1 netmask ff000000
lo0:1: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
 zone zone1
 inet 127.0.0.1 netmask ff000000
nge0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
 inet 10.7.10.14 netmask ffff0000 broadcast 10.7.255.255
 ether 0:e0:81:73:52:cc
nge0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
 zone zone1
 inet 10.7.10.114 netmask ffff0000 broadcast 10.7.255.255

#

17. Use `zlogin -C` to connect to the console of zone1, and respond to the system identification questions that are presented. Be certain to select a terminal type that is appropriate for your system.



Note – If you are connected to your system remotely, it may be useful to specify an alternate escape character for `zlogin` console connections. Do this to avoid inadvertently disconnecting your remote connection. For example, to use the caret symbol instead of the tilde symbol (the default), you would use: `zlogin -C -e \^ zone1`

```
# zlogin -C zone1
[Connected to zone 'zone1' console]
...
```

Select a Language

- 0. English
- 1. French
- 2. German
- 3. Italian
- 4. Japanese
- 5. Korean
- 6. Simplified Chinese
- 7. Spanish
- 8. Swedish
- 9. Traditional Chinese

Please make a choice (0 - 9), or press h or ? for help:
0

...
<enter <cr> as required to display these selections>
...

Select a Locale

- 57. Thai UTF-8
- 58. Turkey (ISO8859-9)
- 59. Turkey (UTF-8)
- 60. U.S.A. (UTF-8)
- 61. U.S.A. (en_US.ISO8859-1)
- 62. U.S.A. (en_US.ISO8859-15)
- 63. Go Back to Previous Screen

Press Return to show more choices.

Exercise 1 Solutions: Configuring Zones

Please make a choice (0 - 63), or press h or ? for help: **61**

...

What type of terminal are you using?

- 1) ANSI Standard CRT
- 2) DEC VT52
- 3) DEC VT100
- 4) Heathkit 19
- 5) Lear Siegler ADM31
- 6) PC Console
- 7) Sun Command Tool
- 8) Sun Workstation
- 9) Televideo 910
- 10) Televideo 925
- 11) Wyse Model 50
- 12) X Terminal Emulator (xterms)
- 13) CDE Terminal Emulator (dtterm)
- 14) Other

Type the number of your choice and press Return: **3**

...

After you choose the language, locale, and terminal type appropriate for your system, an interactive GUI utility prompts you for the remaining required information. After you enter this information, the zone reboots automatically.

Host name for *erio0:n* = **zone1**
Configure kerberos security = **no**
Name service = **none**
Use the NFSv4 domain derived by the system = <**select**>
Default time zone = <**location specific**>
Root password = **cangetin**

18. Exit the zlogin console session. If you specified an alternate escape character in your zlogin command line, enter it instead of the tilde character.

```
zone1 console login: ~.  
[Connection to zone 'zone1' console closed]  
#
```

Task 2 – Administering Users and Data Within Zones

Perform the following steps to verify how user and data resources are distinct among zones:

1. Verify that the non-global zone has mounted the file system you specified in its configuration steps, and that you can create a file within that file system.

```
# zlogin zone1
[Connected to zone 'zone1' pts/5]
Sun Microsystems Inc. SunOS 5.10          Generic January 2005
# uname -a
SunOS zone1 5.10 Generic_118855-33 i86pc i386 i86pc
# df -h dir1
Filesystem           size   used  avail capacity  Mounted on
[dir1]              996M   1.0M  935M      1%    /dir1
# cd /dir1
# touch file1
# ls
file1      lost+found
# exit

[Connection to zone 'zone1' pts/5 closed]
#
```

2. Log in to zone1 and create a user called user1.

```
# zlogin zone1
[Connected to zone 'zone1' pts/5]
Last login: Mon Apr 30 14:16:01 on pts/5
Sun Microsystems Inc. SunOS 5.10          Generic January 2005
# mkdir /export/home
# useradd -u 1000 -g 10 -md /export/home/user1 user1
64 blocks
# passwd user1
New Password:
Re-enter new Password:
passwd: password successfully changed for user1
#
```

Exercise 1 Solutions: Configuring Zones

3. Verify that user1 exists in the /etc/passwd file in zone1, and then log out of zone1.

```
# grep user1 /etc/passwd
user1:x:1000:10::/export/home/user1:/bin/sh
# exit
[Connection to zone 'zone1' pts/5 closed]
#
```

4. Check if user1 is listed in the /etc/passwd file of the global zone

```
# grep user1 /etc/passwd
#
```

Note – The user called user1 will exist only in zone1.



Task 3– Installing Software Packages in Zones

Perform the following steps to install packages in zones:

1. In the global zone on your system, change to the /opt/ses/lab/sparc/zones or /opt/ses/lab/x86/zones directory, depending on your system architecture. Here you will find the packages for the top and sudo applications. Ask your instructor if you need help.

If you are on an x86-based machine, you use the following packages:

- The top-3.5.1-sol10-zones-intel-local package
- The sudo-1.6.8p9-sol10-zones-intel-local package

If you are on a SPARC-based machine, you use the following packages:

- The top-3.5.1-sol10-zones-sparc-local package
- The sudo-1.6.8p4-sol10-zones-sparc-local package

2. Use pkginfo to display the names of the packages contained in the package files.

```
# pkginfo -d top-3.5.1-sol10-zones-intel-local
application SMCTop top
# pkginfo -d sudo-1.6.8p9-sol10-zones-intel-local
application SMCsudo sudo
#
```

3. Use the pkgparam command to display the value of the SUNW_PKG_ALLZONES parameter for the SMCsudo and SMCTop packages. If the SUNW_PKG_ALLZONES parameter is not configured, the result is equal to the parameter being set to false.

```
# pkgparam -d sudo-1.6.8p9-sol10-zones-intel-local SMCsudo SUNW_PKG_ALLZONES
false
# pkgparam -d top-3.5.1-sol10-zones-intel-local SMCTop SUNW_PKG_ALLZONES
false
#
```



Note – When set to false, the SUNW_PKG_ALLZONES parameter determines that the package can be installed: 1) in the global zone only, or 2) in the global zone and all non-global zones, or 3) from any non-global zone to the same non-global zone.

4. Install the SMCTop package in the global zone only. All files for this package are installed below the /opt/local directory.

```
# pkgadd -d top-3.5.1-sol10-zones-intel-local -G
```

The following packages are available:

```
1  SMCTop      top
   (intel) 3.5.1
```

Select package(s) you wish to process (or 'all' to process all packages). (default: all) [?,??,q]: **1**

Processing package instance <SMCTop> from </opt/ses/lab/x86/zones/top-3.5.1-sol10-zones-intel-local>

top(intel) 3.5.1
William LeFebvre et al

The selected base directory </opt/local> must exist before installation is attempted.

Do you want this directory created now [y,n,?,q] **y**

Exercise 1 Solutions: Configuring Zones

Using </opt/local> as the package base directory.

```
## Processing package information.  
## Processing system information.  
## Verifying disk space requirements.  
## Checking for conflicts with packages already installed.  
## Checking for setuid/setgid programs.
```

Installing top as <SMCTop>

```
## Installing part 1 of 1.  
/opt/local/bin/top  
/opt/local/doc/top/Changes  
/opt/local/doc/top/FAQ  
/opt/local/doc/top/INSTALL  
/opt/local/doc/top/README  
/opt/local/doc/top/SYNOPSIS  
/opt/local/doc/top/Y2K  
/opt/local/doc/top/metatop  
/opt/local/man/man1/top.1  
[ verifying class <none> ]
```

Installation of <SMCTop> was successful.

```
#
```

5. Log in to zone1 and verify that the SMCTop package was not installed in that zone. Log out from zone1 when finished.

```
# zlogin zone1  
[Connected to zone 'zone1' pts/5]  
Last login: Mon Apr 30 14:18:44 on pts/5  
Sun Microsystems Inc. SunOS 5.10 Generic January 2005  
# pkginfo SMCTop  
ERROR: information for "SMCTop" was not found  
# exit
```

```
[Connection to zone 'zone1' pts/5 closed]  
#
```

6. From the global zone, install the SMCsudo package in the global zone and the non-global zone.

```
# pkgadd -d sudo-1.6.8p9-sol10-zones-intel-local
```

The following packages are available:

```
1 SMCsudo      sudo  
          (intel) 1.6.8p9
```

Select package(s) you wish to process (or 'all' to process

```
all packages). (default: all) [?,??,q]: 1
## Verifying package <SMCsudo> dependencies in zone <zone1>
```

Files that are setuid and/or setgid will be installed and/or modified for package <SMCsudo> on zone <zone1>.

Do you want to continue with the installation of <SMCsudo> [y,n,?] **y**

```
Processing package instance <SMCsudo> from </opt/ses/lab/x86/zones/sudo-1.6.8p9-sol10-zones-intel-local>
```

```
## Installing package <SMCsudo> in global zone
```

```
sudo(intel) 1.6.8p9
```

Todd Miller

Using </opt/local> as the package base directory.

```
## Processing package information.
```

```
## Processing system information.
```

3 package pathnames are already properly installed.

```
## Verifying disk space requirements.
```

```
## Checking for conflicts with packages already installed.
```

```
## Checking for setuid/setgid programs.
```

The following files are being installed with setuid and/or setgid permissions:

```
/opt/local/bin/sudoedit <setuid root>
```

Do you want to install these as setuid/setgid files [y,n,?,q] **y**

Installing sudo as <SMCsudo>

```
## Installing part 1 of 1.
```

(output omitted)

Installation of <SMCsudo> was successful.

```
## Installing package <SMCsudo> in zone <zone1>
```

```
sudo(intel) 1.6.8p9
```

The selected base directory </opt/local> must exist before installation is attempted.

Do you want this directory created now [y,n,?,q] **y**

Using </opt/local> as the package base directory.

```
## Processing package information.
```

```
## Processing system information.
```

Exercise 1 Solutions: Configuring Zones

Installing sudo as <SMCsudo>

```
## Installing part 1 of 1.  
(output omitted)
```

```
/opt/local/bin/sudo <linked pathname>  
/opt/local/man/man1m/sudo.1m <linked pathname>
```

Installation of <SMCsudo> on zone <zone1> was successful.

```
#
```

7. Log in to zone1 and verify that the SMCsudo package has been installed. Log out from zone1 when finished.

```
# zlogin zone1
```

```
[Connected to zone 'zone1' pts/5]
```

```
Last login: Mon Apr 30 14:38:07 on pts/5
```

```
Sun Microsystems Inc. SunOS 5.10 Generic January 2005
```

```
# pkginfo SMCsudo
```

```
application SMCsudo sudo
```

```
# exit
```

```
[Connection to zone 'zone1' pts/5 closed]
```

```
#
```

Task 4 - Removing Zones

Perform the following steps to remove the zone, zone1:

1. From the global zone list the current configured zones with zoneadm command.

```
# zoneadm list -cv
```

| ID | NAME | STATUS | PATH | BRAND | IP |
|----|--------|---------|---------------|--------|--------|
| 0 | global | running | / | native | shared |
| 1 | zone1 | running | /export/zone1 | native | shared |

2. Use the zoneadm command to halt any running non global zones.

```
# zoneadm -z zone1 halt
```

```
# zoneadm list -cv
```

| ID | NAME | STATUS | PATH | BRAND | IP |
|----|--------|-----------|---------------|--------|--------|
| 0 | global | running | / | native | shared |
| 1 | zone1 | installed | /export/zone1 | native | shared |

3. Use the zoneadm command to uninstall zone1 and zone2.

```
# zoneadm -z zone1 uninstall
```

Are you sure you want to uninstall zone zone1 (y/[n])? **y**

```
# zoneadm list -cv
```

| ID | NAME | STATUS | PATH | BRAND | IP |
|----|--------|------------|---------------|--------|---------|
| 0 | global | running | / | native | shared |
| 1 | zone1 | configured | /export/zone1 | native | sharedI |

4. Delete the configuration of zone1 and zone2 from the global system.

```
# zonecfg -z zone1 delete
```

Are you sure you want to delete zone zone1 (y/[n])? **y**

```
# zoneadm list -cv
```

| ID | NAME | STATUS | PATH | BRAND | IP |
|----|--------|---------|------|--------|--------|
| 0 | global | running | / | native | shared |

5. Umount the /export/zone1 filesystem and remove the entry for /export/zone1 from /etc/vfstab.

```
# umount /export/zone1
```

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

Exercise 2: Validating a Zone Migration Before the Migration Is Performed

In this exercise, you complete the following tasks:

- Configure a Non-Global Zone
- Validate the Zone Migration

Preparation

This exercise requires the following:

- This lab requires you to work on two systems running Solaris operating system.
- Ensure that the PermitRootLogin value is set to ‘yes’ in the /etc/ssh/sshd_config file and that the sshd daemon is running on both the systems.
- You must be the global administrator in the global zone to perform the tasks in this exercise.

Further, refer to the lecture notes and additional resources to perform the steps listed.

Task 1 - Configuring a Non-Global Zone

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Use the zonecfg command to create a non-global zone called zone2 and set the zonepath property to /myzone/zone2.
3. Use the zoneadm command to verify that no non-global zone is currently installed.
4. Open another terminal window and use the zoneadm command to install zone2. Zone installation takes different amounts of time to complete, depending on your system’s capabilities.
5. After the installation processes for the non-global zone completes, use the zoneadm command to verify the status of the zone.

6. Use the zoneadm command to boot zone2. The initial zone boot process can take a few minutes to complete.
7. Verify that zone2 is running.
8. Use zlogin -C to connect to the console of zone2, and respond to the system identification questions that are presented. Be certain to select a terminal type that is appropriate for your system.
9. Exit the zlogin console session.

Task 2 - Creating and Transferring Zone Manifest

Complete the following steps:

1. On the host system, use the zoneadm command to generate the manifest for the zone, zone2 and direct the output to a file.
2. Use the tar command to create an archive of the zone2-manifest.txt file on host system and transfer it to the target system using the sftp command.
3. On the target system, extract the tar file.

Task 3 - Validating Zone Migration Before the Actual Migration

Complete the following steps

1. On the target system, use the zonecfg command to create a non-global zone called zone2 and set the zonepath property to /myzone/zone2.
2. Use the zoneadm command to validate the feasibility of zone migration.

Exercise Summary

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



- Experiences
- Interpretations
- Conclusions
- Applications

Exercise 2 Solutions: Validating a Zone Migration Before the Migration Is Performed

In this exercise, you complete the following tasks:

- Configure a Non-Global Zone
- Create and Transfer Zone Manifest
- Validate Zone Migration Before the Actual Migration

Preparation

This exercise requires the following:

- This lab requires you to work on two systems running Solaris operating system.
- Ensure that the PermitRootLogin value is set to ‘yes’ in the /etc/ssh/sshd_config file and that the sshd daemon is running on both the systems.
- You must be the global administrator in the global zone to perform the tasks in this exercise.

Task 1 - Configuring a Non-Global Zone

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.
2. Use the zonecfg command to create a non-global zone called zone2 and set the zonepath property to /myzone/zone2.

```
# zonecfg -z zone2
zone2: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:zone1> create
zonecfg:zone1> set zonepath=/myzone/zone2
zonecfg:zone1> verify
zonecfg:zone1> commit
zonecfg:zone1> exit
```

3. Use the zoneadm command to verify that no non-global zone is currently installed.

```
# zoneadm list -i
```

4. Open another terminal window and use the zoneadm command to install zone2. Zone installation takes different amounts of time to complete, depending on your system's capabilities.

```
# zoneadm -z zone2 install
```

5. After the installation processes for the non-global zone completes, use the zoneadm command to verify the status of the zone.

```
# zoneadm list -iv
```

6. Use the zoneadm command to boot zone2. The initial zone boot process can take a few minutes to complete.

```
# zoneadm -z zone2 boot
```

7. Verify that zone2 is running.

```
# zoneadm list -iv
```

8. Use zlogin -C to connect to the console of zone2, and respond to the system identification questions that are presented. Be certain to select a terminal type that is appropriate for your system.

```
# zlogin -C zone2
```

9. Exit the zlogin console session.

```
zone2 console login: ~.  
[Connection to zone 'zone2' console closed]  
#
```

Task 2 - Creating and Transferring Zone Manifest

Complete the following steps:

1. On the host system, use the zoneadm command to generate the manifest for the zone, zone2 and direct the output to a file.

```
# zoneadm -z zone2 detach -n > /export/home/zone2-manifest.txt
```

2. Use the tar command to create an archive of the zone2-manifest.txt file on host system and transfer it to the target system using the sftp command.

```
host1# cd /export/home  
host1# tar cf zone2-manifest.tar zone2-manifest.txt  
host1# sftp host2  
Connecting to host2...  
Password:  
sftp> cd /export/home  
sftp> put zone2-manifest.tar
```

Exercise 2 Solutions: Validating a Zone Migration Before the Migration Is Performed

```
Uploading zone2-manifest.tar to /export/home/zone2-
manifest.tar
sftp> quit
```

3. On the target system, extract the tar file.

```
host2# cd /export/home
host2# tar xf zone2-manifest.tar
```

Task 3 - Validating Zone Migration Before the Actual Migration

Complete the following steps

1. On the target system, use the zonecfg command to create a non-global zone called zone2 and set the zonepath property to /myzone/zone2.

Note – You can specify a different value for the zone name and a different filesystem path for the zonepath property.



```
# zonecfg -z zone2
zone2: No such zone configured
Use 'create' to begin configuring a new zone.
zonecfg:zone1> create
zonecfg:zone1> set zonepath=/myzone/zone2
zonecfg:zone1> verify
zonecfg:zone1> commit
zonecfg:zone1> exit
```

2. Use the zoneadm command to validate the feasibility of zone migration.

```
host2# zoneadm -z zone2 attach -n /export/home/zone2-
manifest.txt
```

A successful attach does not show any output and returns you to the command prompt.

Note – The output of the validation command shows what will happen when you do the migration for real, without the -n option.



Exercise 3: Using Solaris Containers for Linux Applications

In this exercise, you complete the following tasks:

- Configure the `lx` Branded Zone
- Install the `lx` Branded Zone
- Boot the `lx` Branded Zone
- Log on to the `lx` Branded Zone

Preparation

This exercise requires the following:

- An x86/x64 based system running Solaris 10 8/09 or later update release.
- A Linux distribution as a compressed tar archive. You can download a tarball from the opensolaris brandz community download site. The tarball contains a complete file system image from an installed CentOS distribution.
- You must be the global administrator in the global zone to perform the tasks in this exercise.

Further, refer to the lecture notes and additional resources to perform the steps listed.

Task 1 - Configuring the `lx` Branded Zone

Complete the following steps:

1. Set up a zone configuration with the zone name, `lx-zone`.
2. Create the new zone configuration for the `lx` branded zone, `SUNWlx`.
3. Set the zonepath as `/myzone/lx-zone`.
4. Verify the zone configuration for the zone, `lx-zone`.
5. Commit the zone configuration for the zone, `lx-zone`.
6. Exit the `zonecfg` command.
7. Use the `zonecfg` command to display the configuration information of
`lx-zone` zone.

Task 2 - Installing the 1x Branded Zone

Complete the following steps:

1. Use the `zoneadm` command to install the 1x branded zone. Zone installation takes different amounts of time to complete, depending on your system's capabilities.
2. When the installation completes, use the `zoneadm` command with the `-i` and `-v` options to list the installed zones and verify the status.

Task 3 - Booting the 1x Branded Zone

Complete the following steps:

1. Use the `zoneadm` command to boot the 1x-zone zone. The initial zone boot process can take a few minutes to complete.
2. Verify that 1x-zone zone is running.

Task 4 - Logging On to the 1x Branded Zone

Complete the following steps:

1. Use the `zlogin` command to connect to the 1x-zone zone.
2. Use the `uname -a` command to verify that 1x-zone zone is running Linux operating system under Solaris.
3. Exit the `zlogin` console session.

Exercise Summary

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



- Experiences
- Interpretations
- Conclusions
- Applications

Exercise 3 Solutions: Using Solaris Containers for Linux Applications

In this exercise, you complete the following tasks:

- Configure the lx Branded Zone
- Install the lx Branded Zone
- Boot the lx Branded Zone
- Log on to the lx Branded Zone

Preparation

This exercise requires the following:

- An x86/x64 based system running Solaris 10 8/09 or later update release.
- A Linux distribution as a compressed tar archive. You can download a tarball from the OpenSolaris brandz community download site. The tarball contains a complete filesystem image from an installed CentOS distribution.
- You must be the global administrator in the global zone to perform the tasks in this exercise.

Further, refer to the lecture notes and additional resources to perform the steps listed.

Task 1 - Configuring the lx Branded Zone

Complete the following steps:

1. Set up a zone configuration with the zone name, lx-zone.

```
# zonecfg -z lx-zone  
lx-zone: No such zone configured  
Use 'create' to begin configuring a new zone.
```

2. Create the new zone configuration for the lx branded zone, SUNWlx.

```
zonecfg:lx-zone> create -t SUNWlx
```

3. Set the zonepath as /myzone/lx-zone.

```
zonecfg:lx-zone> set zonepath=/myzone/lx-zone
```

4. Verify the zone configuration for the zone, lx-zone.

```
zonecfg:lx-zone> verify
```

5. Commit the zone configuration for the zone, lx-zone.

```
zonecfg:lx-zone> commit
```

6. Exit the zonecfg command.

```
zonecfg:lx-zone> exit
```

7. Use the zonecfg command to display the configuration information of lx-zone zone.

```
# zonecfg -z lx-zone info
```

Task 2 - Installing the lx Branded Zone

Complete the following steps:

1. Use the zoneadm command to install the lx branded zone. Zone installation takes different amounts of time to complete, depending on your system's capabilities.

```
# zoneadm -z lx-zone install -d /path/to/linux-archive
```

2. When the installation completes, use the zoneadm command with the -i and -v options to list the installed zones and verify the status.

```
# zoneadm list -iv
```

Task 3 - Booting the lx Branded Zone

Complete the following steps:

1. Use the zoneadm command to boot the lx-zone zone. The initial zone boot process can take a few minutes to complete.

```
# zoneadm -z lx-zone boot
```

2. Verify that lx-zone zone is running.

```
# zoneadm list -iv
```

Task 4 - Logging On to the lx Branded Zone

Complete the following steps:

1. Use the zlogin command to connect to the lx-zone zone.

```
# zlogin lx-zone
```

2. Use the uname -a command to verify that lx-zone zone is running Linux operating system under Solaris.

```
# uname -a
```

```
Linux lx-zone 2.4.21 BrandZ fake linux i686 i686 i386  
GNU/Linux
```

3. Exit the zlogin console session.

Notes:

Module 11

Introduction to LDAP

Objectives

This module introduces the LDAP protocol for querying and modifying directory services running over TCP/IP.

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

- Understand the use of LDAP as a naming service
- Describe basic LDAP concepts and terminology
- Identify the Directory Server Enterprise Edition requirements
- Identify Solaris LDAP Client requirements

LDAP As A Naming Service

Organizations, carriers, and service providers are developing and deploying more identity enabled applications to more users every day. These applications rely on a directory to provide secure, reliable access to digital identities and their credentials. Older naming services like NIS or NIS+ are not robust enough in their design to meet the challenges of today's namespace requirements. NIS used flat files that contained simple lookup tables. LDAP is hierarchical in design and provides the functionality to meet those challenges.

The Lightweight Directory Access Protocol (LDAP)

LDAP is a protocol for querying and modifying directory services running over TCP/IP. A directory in this context is defined as a set of objects with attributes organized in a logical and hierarchical manner. A directory can store identity data that can be used by the client systems such as user names and passwords.

In today's increasingly networked environments, a directory must offer much more than just a repository for identity data. It must provide easy, yet secure access to information in multiple repositories, maintain the highest possible levels of availability, and be able to scale dramatically to keep pace with constantly growing and changing groups of internal and external users. A directory that is only a data repository cannot possibly address all these challenges. For this reason, many enterprises have turned to multiple point solutions to provide all the directory-related functionality they need. This can result in an unnecessarily complex directory environment that can be extremely costly both to deploy and to administer.

Sun Microsystems provides the Sun Java System Directory Server Enterprise Edition (DSEE) to deliver a secure, highly-available, scalable directory service for storing and managing accurate and reliable identity data. It serves as the backbone to an enterprise identity infrastructure, enabling today's mission-critical enterprise applications and large-scale extranet applications to access consistent, accurate, and reliable identity data for significant operational and cost efficiencies.

This solution provides a solid foundation for identity management by providing a central repository for storing and managing identity profiles, access privileges, and application and network resource information. It integrates smoothly into multi-platform environments, and provides secure, on-demand password synchronization with Microsoft Windows Active Directory.

Basic LDAP Concepts and Terminology

This section reviews LDAP and establishes a baseline for LDAP concepts used in the rest of the course. The following are examples of concepts:

- Directory service definition
- LDAP
- Four defined LDAP models:
 - An informational model that describes what you can put in the directory
 - A naming model that describes how you arrange and refer to directory data
 - A functional model that describes what you can do with directory data
 - A security model that describes how directory data can be protected from unauthorized access
- LDAP search parameters
- LDAP Data Interchange Format (LDIF)

LDIF is a standard interchange format for LDAP directory data. An LDIF file is a handy mechanism for editing, viewing, and moving an LDAP database.

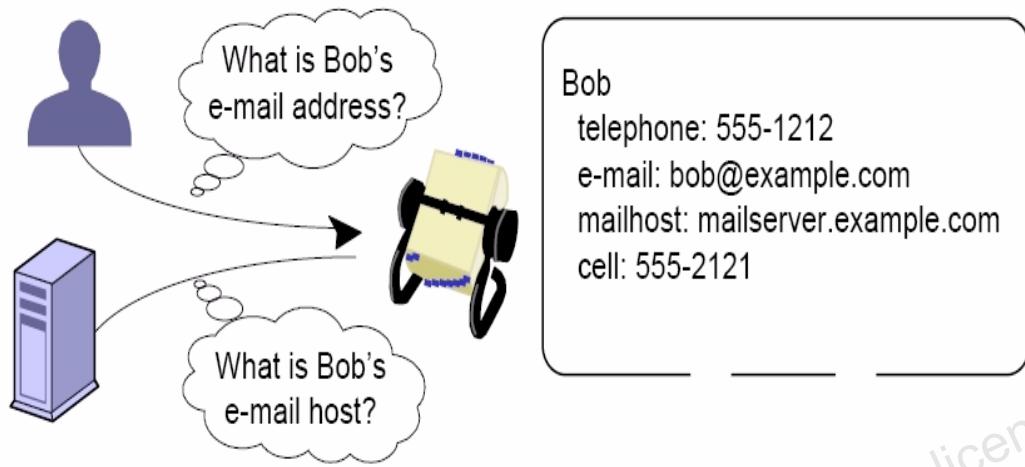


Figure 11-1 Defining A Directory Service

Defining a Directory Service

A directory service is a service that provides information about people and resources to a client requesting information. The most commonly known directory service is one provided by a telephone company, where a client communicates a request to the operator by giving a name and address. The operator searches the directory database and returns the telephone number.

An example of an Internet-based directory service, as illustrated in the overhead, is when a user needs an e-mail address. The user can look in a directory service, find the entry, and retrieve the e-mail address for that entry. The user can then send e-mail to the person.

Another example might be a mail server on the Internet that routes mail for a given user in the To field. The mail server application can search the same directory service, find the entry for the user, and retrieve the e-mail host address to which to send the e-mail.

The long-term goal for a directory services solution is to have a global directory service built on open standards that lets users and applications access information about people and resources easily from anywhere on the Internet, intranet, or both.

The directory service solution should also include ways to limit access to information based on the users requesting services and the resource they are trying to access.

Information Systems should also be able to maintain a copy of the directory database in one central location and distribute (replicate) that information as needed to other servers.

Directory Schema

A directory schema defines the rules of storing directory data in a Directory Server. A directory schema maintains the integrity of the data stored in your directory by imposing constraints on the size, range, and format of data values. A schema defines what types of entries your directory contains (such as people, devices, and organizations) and the attributes available to each entry.

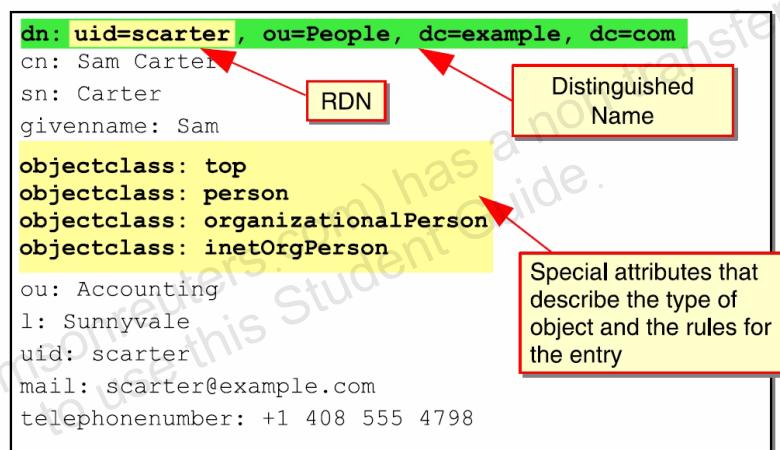


Figure 11-2 LDAP Directory Schema

For example, in Figure 12-2 the first line of this directory entry is the Distinguished Name (DN), which is represented using the dn: context at the beginning of the entry. The DN uniquely identifies the entry in the directory database and where it is located in the directory tree hierarchy. The directory entry uses the uid (unique identifier) attribute in the distinguished name to uniquely identify the entry within the People branch point. This left-most part of the distinguished name is known as the Relative Distinguished Name (RDN). The cn (common name), sn (surname), and givenname attributes describe the entry in more detail.

The `objectclass` attributes define the rules for the entry. An object class is a special attribute type called `objectclass` that basically defines which attributes are mandatory and which are optional for a specific entry. The `objectclass` attribute also establishes the entry type based on its given value. In the previous example, you can tell this entry is a person by looking at the values given for the `objectclass` attributes (`objectclass: person`, `objectclass: organizationalPerson`, and `objectclass: inetOrgPerson`).

The directory schema can contain the same information found in ascii files such as `/etc/passwd` or `/etc/hosts`.

LDAP

LDAP is a protocol that provides a standard approach for communication between clients and servers. This enables software from different vendors to work together.

The lightweight aspect of the protocol means that it is easy to use and implement while still being highly functional. Older protocols, such as the X.500 Directory Access Protocol (DAP) of which LDAP is based on, use complex encoding methods and require the use of the Open System Interconnection (OSI) network protocol stack.

LDAP runs on top of Transmission Control Protocol/Internet Protocol (TCP/IP). This means it can be used by every major operating system generally available (Microsoft Windows, DOS, UNIX, and Apple OS) because these operating systems provide TCP/IP as standard or as an add-on. This allows for a cross platform namespace. Native LDAP assumes that the LDAP client is configured to use DNS for their host to IP address resolution.

LDAP supports a set of application programming interfaces (APIs) for a variety of languages that support the development of directory-enabled applications. They include C application APIs, Java™ technology APIs, and Perl LDAP.

Four Defined LDAP Models

The LDAP information model defines the basic unit of information in the directory as an entry. The entry is a collection of information about an object. The entry is composed of a more fundamental element known as an attribute. Each attribute has a type, and one or more values.

The four models are:

- Informational model
- Naming model
- Functional model
- Security model

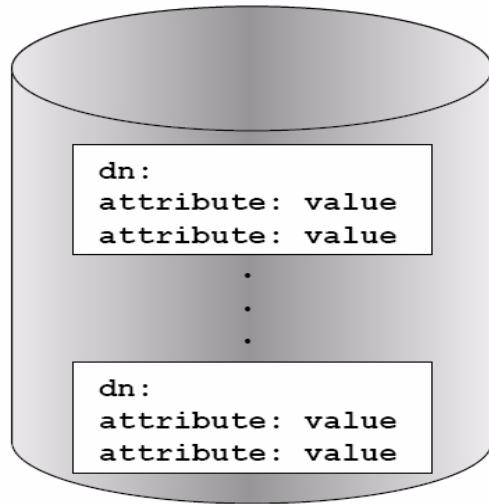


Figure 11-3 Informational Model of a LDAP Directory

LDAP Directory: Information Model

The directory database consists of one or more directory entries. Entries are structured to form a hierarchical tree of information.

Each entry in the database is identified by a unique distinguished name (DN). The DN is a string representation of the entry's location in the directory tree. You can think of the DN as the full path name from the root to the database entry.

An entry can also have one or more attributes that further describe the entry. This is analogous to the files that belong to the directory in UNIX.

You can copy an LDAP database to LDIF and view the database with a text editor or viewer.

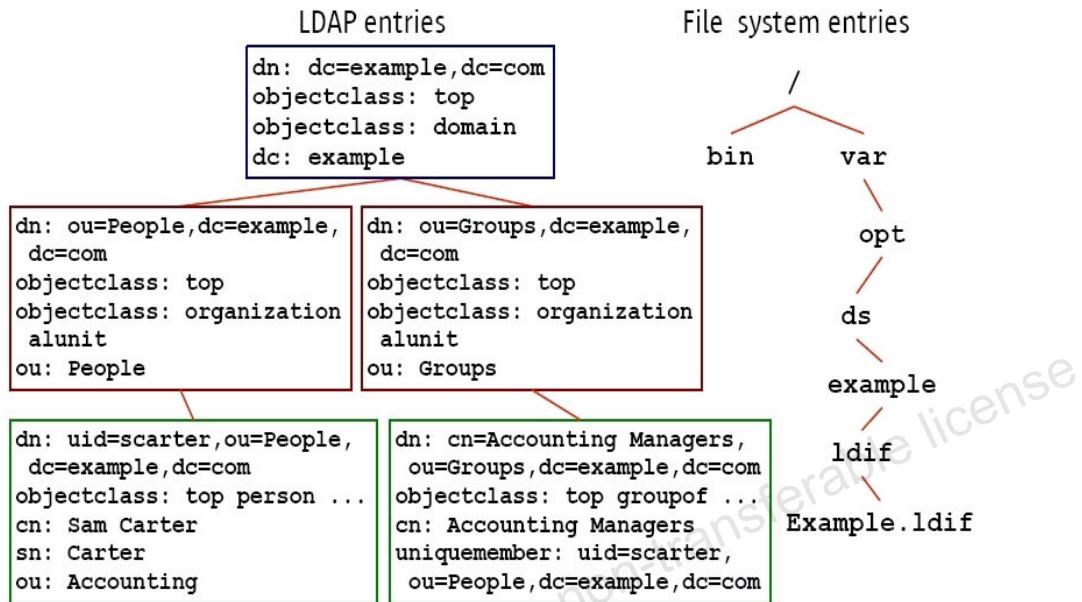


Figure 11-4 Naming Model of a LDAP Directory

LDAP Entries: Naming Model

The naming model defines how you organize and refer to the data. It describes the types of structures built from the individual blocks (the directory entries). The naming model also identifies how to refer to any entry within the structure.

The LDAP naming model specifies that entries are arranged in an inverted tree structure. The naming model identifies how to refer to any particular entry within that structure. This structure is very similar to the UNIX file system. However, the LDAP model has the following three significant differences:

- The root entry is conceptual rather than an actual entry into which you can place data.
- Every node contains data, and any node can be a container. This contrasts with a file system, in which any given node is a file or a directory, but not both.
- Names are in backward order relative to file system names. That is, the names are constructed leaf-to-root rather than root-to-leaf, as they are constructed in file systems.

Basic LDAP Concepts and Terminology

Each entry is identified by a DN. This results in a unique name for all entries in the directory, which allows unambiguous references to any entry in the directory.

LDAP Protocol: Functional Model

The LDAP functional model describes the operations that you can perform on the directory using the LDAP protocol. The model defines three groups of operations as follows:

- Interrogation operations
- Update operations
- Authentication operations

Two LDAP interrogation operations let LDAP clients search the directory and retrieve directory data. They include the following operations:

- Search – Searches the directory for entries and retrieves individual directory entries.
- Compare – Checks whether a particular entry contains a particular attribute value. The client submits a compare request to the server, supplying a DN, an attribute name, and a value. The server returns an affirmative response to the client if the entry named by the DN contains the given value in the given attribute type. If not, a negative response is returned.

Four LDAP update operations let you manipulate the data in your directory. They are:

- Add – Creates new directory entries. It has two parameters:
 - The DN of the entry to be created
 - A set of attributes and attribute values to comprise the new entry
- Delete – Removes an entry from the directory. It has a single parameter, the DN of the entry to be deleted.
- Modify – Modifies an entry in the directory. It can be used to add, delete, or replace attributes within an entry.
- Rename, or modify DN – Renames and moves entries in the directory. It has four parameters:
 - The DN of the entry to be renamed
 - The new relative distinguished name (RDN) for the entry
 - An optional argument giving the new parent of the entry
 - The `delete-old-RDN` flag

There are two LDAP authentication operations and one control operation:

- Bind – Used by a client to authenticate itself to the directory. It does so by providing a DN and a set of credentials.
- Unbind – Used to end any outstanding LDAP operations and disconnect.
- Abandon – Used when the client is no longer interested in the results of a previously initiated operation. When the server receives an abandon request, the server terminates processing of the operation.

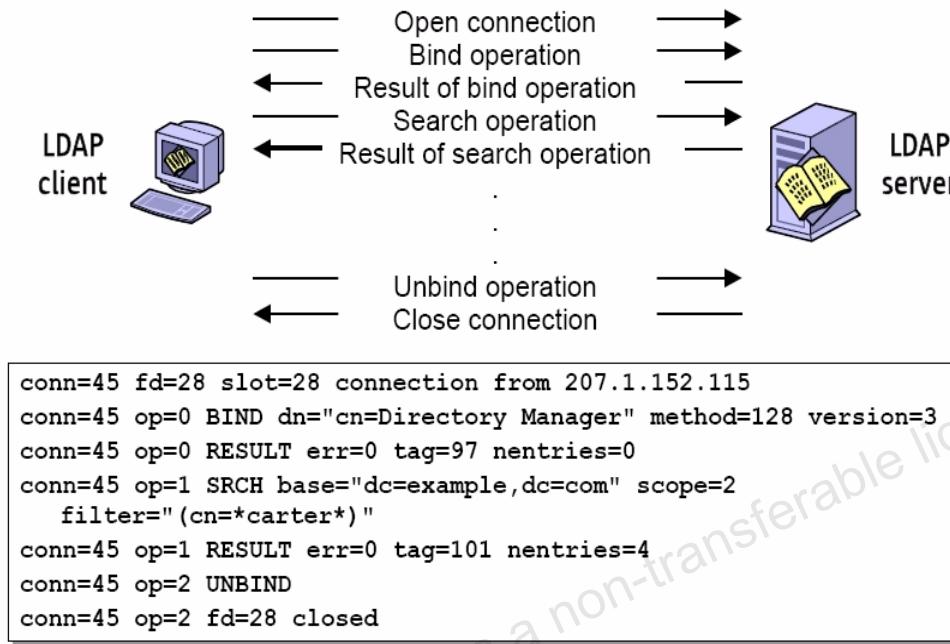


Figure 11-5 Security Model of a LDAP Directory

LDAP Request: Security Model

The LDAP security model provides a framework for protecting the information in the directory from unauthorized access. The security model relies on the fact that LDAP is a connection-oriented protocol. This means that the client opens a connection to an LDAP server and performs a number of protocol operations on the same connection. Information passed between the client and server is not passed in clear text providing much better security than some of the older naming services. The LDAP client can authenticate to the directory server at some point during the connection's lifetime, at which point it might be granted additional (or fewer) privileges.

LDAP is a message-oriented protocol. The client constructs an LDAP message containing a request and sends it to the server. The server processes the request and sends the results back to the client as a series of LDAP messages.

Because LDAP is message-based, the client can issue multiple requests at the same time. The client generates a unique message ID for each request.

Basic LDAP Concepts and Terminology

Returned results for specific requests are tagged with the message ID, which lets the client sort out multiple responses to different requests when results arrive out of order or at the same time.

The following extract from the Directory Server access log shows a typical message sequence between a client and Directory Server:

```
conn=45 fd=28 connection from 207.1.152.115
conn=45 op=0 BIND dn= "method=128 version=3
conn=45 op=0 RESULT err=0 tag 97 nentries=0
conn=45 op=1 SRCH base="dc=dc=example,dc=com" scope=2
filter="(cn=*carter*)"
conn=45 op=1 RESULT err=0 tag=101 nentries=4
conn=45 op=2 UNBIND
conn=45 op=2 fd=28 closed
```

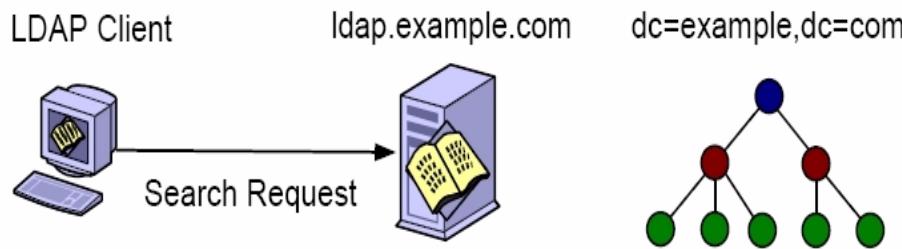


Figure 11-6 LDAP Search Parameters

Defining LDAP Search Parameters

One way users can obtain information from Directory Server is by using the LDAP command-line utility `ldapsearch`. While it is unlikely that users will do this regularly, the information is helpful for administrators and programmers to know.

The `ldapsearch` utility locates and retrieves directory entries. This utility opens an authenticated or anonymous connection to a server and locates entries based on a specified search filter. Search scopes can include a single entry, an entry's immediate sub-entries, or an entire tree or subtree.

When you use the `ldapsearch` utility, you must enter the utility using the following format:

```
ldapsearch [options] filter [attributes]
```

Options

The following list explains the most commonly used `ldapsearch` options.

- `-b` – Specifies the starting point (the base) for the search. The value specified here must be a DN that currently exists in the database. This parameter is optional if the `LDAP_BASEDN` environment variable has been set to a base DN. If the `LDAP_BASEDN` environment variable is not set, you *must* specify the `-b` option with the `ldapsearch` utility. For example, `-b dc=example,dc=com`

The value specified in this parameter should be provided in double quotation marks. If you want to search the root DSA-specific entry (root DSE), specify an empty string. For example, `-b ""`

- `-h` – Specifies the host name or Internet Protocol (IP) address of the machine on which Directory Server is installed. If you do not specify a host, the `ldapsearch` utility uses the local host. For example, `-h example-ds`
- `-p` – Specifies the Transmission Control Protocol (TCP) port number that Directory Server uses. For example, `-p 1049`. The default is: 389.
- `-D` – Specifies the DN with which to authenticate to the server. This parameter is optional if anonymous access is supported by your server. If specified, this value must be a DN recognized by Directory Server, and it must also have the authority to search for the entries. For example, `-D "cn=Directory Manager"`
- `-w` – Specifies the password associated with the distinguished name that is specified in the `-D` option. If you do not specify this parameter, anonymous access is used. For example, `-w dinner89&2`
- `-s` – Specifies the search's scope. The scope can be one of the following:
 - `sub` – Searches the entry specified in the `-b` parameter and all of its descendants (a sub-tree search). The scope `-s sub` is the default scope for searches.
 - `base` – Searches only the entry specified in the `-b` option or defined by the `LDAP_BASEDN` environment variable.
 - `one` – Searches only the immediate children of the entry specified in the `-b` parameter. Only the children are searched; the entry specified in the `-b` parameter is *not* searched.

- `-l` – Specifies the maximum number of seconds to wait for a search request to complete. For example, `-l 300`
Regardless of the value specified here, the ldapsearch utility never waits longer than is allowed by the server's nsslapd-timelimit configuration attribute. The default value for the nsslapd-timelimit configuration attribute is 3600 seconds.
- `-x` – Specifies that the search results are sorted on the server rather than on the client. This is useful if you want to sort according to a matching rule, as with an international search. In general, it is faster to sort on the server than the client.
- `-z` – Specifies the maximum number of entries to return in response to a search request. For example, `-z 1000`

Filter

Search filters provide a way to reduce the number of false responses of a search by specifying unique attributes of the data for which you are searching. You can specify search filters using one of two techniques:

- You can specify a single LDAP search filter on the command line.
- You can specify the ldapsearch utility option `-f filterFile`, where `filterFile` is a text file containing one search filter per line. If `filterFile` contains multiple lines, the ldapsearch utility performs multiple searches.

Note – LDAP search filters and their syntax are defined in RFC 2254. Refer to <http://www.ietf.org/rfc/rfc2254.txt> for more information.



Attributes

You can specify one or more attributes after the search filter. Specifying a list of attributes reduces the scope of the search. Only the attributes specified with the ldapsearch utility are returned in the search results.

If you do not specify an attribute list, then the ldapsearch utility returns values for all attributes (as allowed by the directory's access controls) except *operational attributes* (attributes used by Directory Server for its own purposes).

```
dn: dc=example,dc=com
objectclass: top
objectclass: domain
dc: example

dn: ou=People, dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: People

dn: uid=scarter, ou=People,
dc=example,dc=com
cn: Sam Carter
sn: Carter
givenname: Sam
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
...
...
```

Figure 11-7 The LDAP Data Interchange Format

Utilizing LDIF

LDIF is a specification for representing directory entries in an American Standard Code for Information Interchange (ASCII) format. It was developed by Tim Howes, Mark C Smith, and Gordon Good while at the University of Michigan in the early 1990's. Gordon Good authored RFC 2849 in June 2000 and it is currently the proposed standard.

The Directory Server schema is defined by a series of LDIF files found in the subdirectory:

INSTANCE_PATH/config/schema

where *INSTANCE_PATH* represents the full path to an instance of Directory Server. All information in a directory database can potentially be represented by a single LDIF file. Individual entries can also be displayed using LDIF syntax.

The universal readability of ASCII files makes this format a logical choice for large-scale import or export operations. This is particularly useful for initially populating a directory database—for instance, with information culled from a company's HR database. LDIF is also used by Directory Server to store information such as server configuration data, backup files, and schema files.

Default Directory Information Tree (DIT)

By default, Solaris LDAP clients access the information assuming that the DIT has a given structure. For each domain supported by the LDAP server, there is a sub-tree with an assumed structure. This default structure, however, can be overridden by specifying Service Search Descriptors (SSDs). For a given domain, the default DIT will have a base container that holds a number of well known containers that hold entries for a specific information type. See the following table for the names of these sub-trees. (This information can be found in RFC 2307 and others).

Table 11-1 DIT Default locations

| Default Container | Information type |
|--------------------|--|
| ou=Aliases | aliases(4) |
| ou=Ethers | bootparams(4), ethers(4) |
| ou=Group | group(4) |
| ou=Hosts | hosts(4), ipnodes(4), publickey for hosts |
| ou=Netgroup | netgroup(4) |
| ou=Networks | networks(4), netmasks(4) |
| ou=People | passwd(1), shadow(4), user_attr(4), audit_user(4), publickey for users |
| ou=printers | printers(4) |
| ou=projects | project |
| ou=Protocols | protocols(4) |
| ou=Rpc | rpc(4) |
| ou=Services | services(4) |
| ou=SolarisAuthAttr | auth_attr(4) |
| ou=SolarisProfAttr | prof_attr(4), exec_attr(4) |

Table 11-1 DIT Default locations

| Default Container | Information type |
|---------------------|------------------|
| automountMap=auto_* | auto_* |

General LDAP Tools

LDAP command line tools support a common set of options, including authentication and bind parameters. The following commands can be used to manipulate directory entries directly:

- `ldapsearch(1)`
- `ldapmodify(1)`
- `ldapadd(1)`
- `ldapdelete(1)`

LDIF is typically used by administrators to view and manipulate the actual directory data in an LDAP server database. These utilities allow administrators to populate the directory database from an LDIF file and, conversely, to convert the database to an LDIF file. Any time you see a space character as the first character on a line, that line belongs to the previous line.

Using LDIF format you can move information from one directory to another with commands such as `ldapadd` and `ldapmodify`. Use `ldaplist(1)` with the `-l` option to display the following information. For example:

```
# ldaplist -l hosts myhost
hosts
dn: cn=myhost+ipHostNumber=7.7.7.115,ou=Hosts,dc=mydc,dc=mycom,dc=com
cn: myhost
iphostnumber: 7.7.7.115
objectclass: top
objectclass: device
objectclass: ipHost
description: host 1 - floor 1 - Lab a - building b
```

```
# ldaplist -l passwd user1
passwd
dn: uid=user1,ou=People,dc=mydc,dc=mycom,dc=com
uid: user1
cn: user1
userpassword: {crypt}duTx91g7PoNzE
uidnumber: 199995
gidnumber: 20
gecos: Joe Smith [New York]
homedirectory: /home/user1
loginshell: /bin/csh
objectclass: top
objectclass: shadowAccount
objectclass: account
objectclass: posixAccount
```



Note – For more information on the LDIF specification, refer to: The LDAP Data Interchange Format (LDIF) - Technical Specification - <http://tools.ietf.org/html/rfc2849>

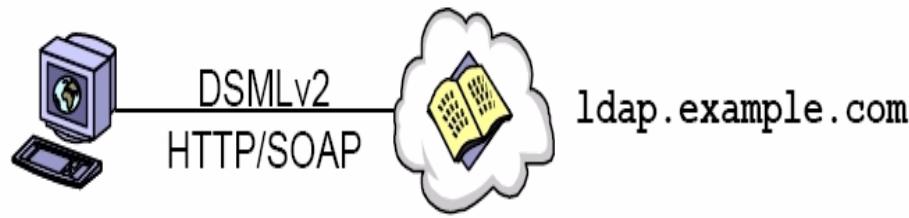


Figure 11-8 DSEE Support of DSMLv2 Over HTTP/SOAP

DSMLv2 Over HTTP/SOAP

Directory Server also supports accessing LDAP data through the use of the Directory Services Markup Language version 2 (DSMLv2) over the Hypertext Transfer Protocol and the Simple Object Access Protocol (HTTP/SOAP).

DSMLv2 over HTTP/SOAP lets non-LDAP clients access an LDAP directory service without using the LDAP protocol. DSMLv2 over HTTP/SOAP expands the number of applications that can access the directory.

Note – For more information about DSMLv2, refer to:
<http://xml.coverpages.org/dsml.html>



Directory Server Requirements

The Directory Server component of Directory Server Enterprise Edition (DSEE) provides the most scalable, high-performance LDAP data store for identity information in the industry and serves as the foundation for the new generation of e-business applications and Web services.

The Directory Server provides for both vertical and horizontal growth without major deployment redesign. This level of scalability becomes increasingly critical as deployment grows.

- The Directory Server is the highest-performing LDAP directory server on the market today, with the ability to provide sustained search performance of over 10,000 entries per second on a single machine and horizontal scalability to tens of thousands of searches per second.
- The requirement to store and update information constantly is increasing with the expansion of use across the organization. Update performance of directory server has been seen near the 1,000 per second range on multi-million entry deployments, allowing for near relational database-write performance.
- As the industry's only 64-bit, enterprise-class directory with linear CPU scalability to 18 CPUs, the Directory Server allows access to maximum memory capacity and delivers high performance accommodating extremely large directories on a single system for maximum hardware benefit.
- Advanced replication mechanism with unlimited number of masters, highly-available change log, prioritized replication and global account lockout let you deploy your service in widely distributed environment to match your geographical constraints.

System Requirements

The system requirements (for a typical 250,000 entries configuration) are:

Directory Server

- Minimum disk space: 4 GB
- Minimum memory: 2 GB

Directory Proxy Server

- Minimum disk space: 2 GB
- Minimum memory: 2 GB

Identity Synchronization for Windows

- Minimum disk space: 400 MB
- Minimum memory: 512 MB

Operating Systems and Platforms

For core Directory Server, Directory Proxy Server, and Directory Server Resource Kit components:

- Sun Solaris 9 and 10 Operating Systems
- Red Hat Enterprise Linux Advanced Server (AS) 3 and 4
- SuSE Linux Enterprise Server 9 and 10
- Hewlett-Packard HP-UX 11.23 (PA-RISC)
- Microsoft Windows 2000 Server and Advanced Server
- Microsoft Windows Server 2003 Standard Edition and Enterprise Edition

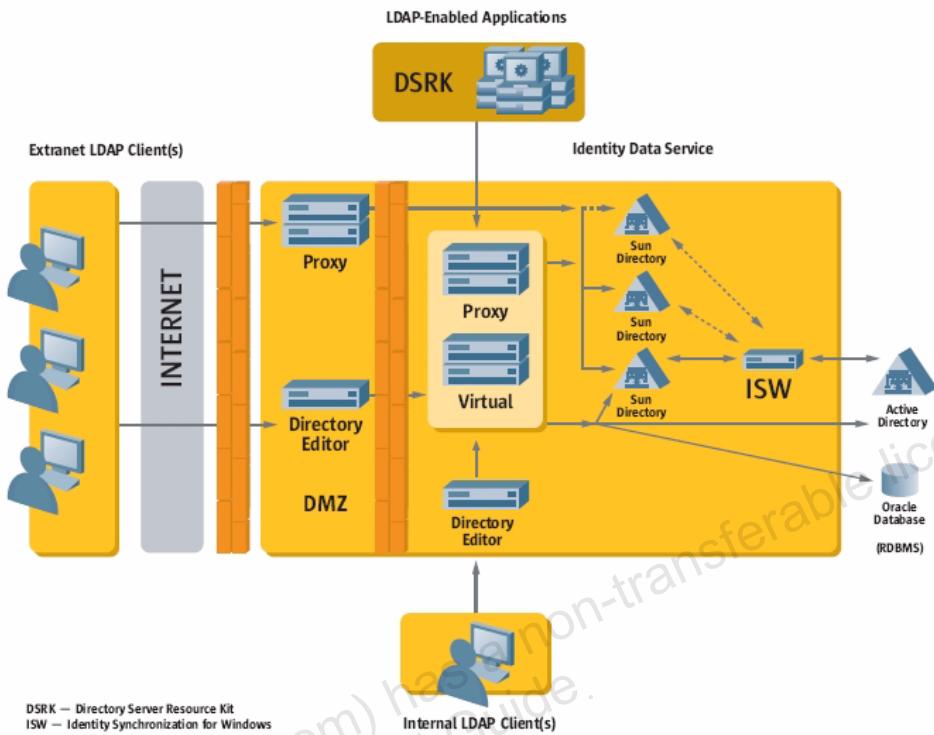


Figure 11-9 Directory Server Enterprise Edition Components

DSEE Components

Directory Server Enterprise Edition includes virtual directory and directory proxy services, and is designed to integrate easily into existing technology infrastructures. For example, because Microsoft Windows provides the desktop interface most frequently found in enterprise environments, Directory Server Enterprise Edition includes identity, group and password synchronization with Microsoft Active Directory. This enables users to change passwords in either Windows or Directory Server Enterprise Edition and keep the password synchronized between the two. Directory Server Enterprise Edition is the only directory solution that offers seamless, non-intrusive integration with the Microsoft environment to enable effortless synchronization.

The components of DSEE are:

- The Directory Server which provides the core directory service
- The web-based Identity and Object Editor which provides an intuitive administration interface for managing users, groups, and organizations
- The Directory Proxy Server for load-balancing, high-availability, virtualization, and distribution capabilities
- The Identity Synchronization for Windows component provides identity data, password, and group synchronization between Microsoft Active Directory and Java System Directory Server
- Directory Server Resource Kit for tuning and optimizing directory server performances

DSEE Security Features

Identity data is by nature sensitive. This mandates a directory solution to provide robust security to protect against illegal access, denial-of-service attacks, and other security risks. In addition to data and communication encryption and password protection, Directory Server Enterprise Edition secures data with Access Control Instructions (ACIs) that define access rights all the way down to the attribute level. The integrated Directory Proxy Server minimizes the risk of unauthorized access to data by providing ACIs at the proxy level, resource usage limits to avoid denial of service attacks, and authentication and encryption enforcement.

Note – Analysis and planning of your DSEE LDAP namespace is covered in the DIR-2217: Sun Java System Directory Server Enterprise Edition 6.x: Analysis and Planning course. DSEE installation, maintenance and administration is covered in the DIR-2340: Sun Java System Directory Server Enterprise Edition 6: Maintenance and Operations course.



Configuring A Client System To Use LDAP

The configuration of the LDAP client requires the following:

- The client's domain name must be served by the LDAP server
- The `nsswitch.conf` file needs to point to LDAP for the required services
- The client needs to be configured with all the given parameters that define its behavior
- `ldap_cachemgr` needs to be running on the client
- At least one server for which a client is configured must be up and running

ldapclient Utility

The `/usr/sbin/ldapclient` utility is the key to setting up an LDAP client, as it performs all of the above steps, except for starting the server.

The `ldapclient (1M)` utility is used to set up LDAP clients in the Solaris system. This utility assumes the server has already been configured with the appropriate client profiles. You must install and configure the server with the appropriate profiles before you can set up clients.

There are two main ways to set up a client by using `ldapclient`.

- Profile

At a minimum, you need to specify the server address containing the profile and domain you want to use. If no profile is specified, then the "default" profile is assumed. The server will provide the rest of the required information, except for proxy and certificate database information. For example:

To initialize a client profile run `ldapclient` with `init`:

```
# ldapclient init \
-a profileName=new \
-a domainName=west.example.com 192.168.0.1
System successfully configured
```

- Manual

You configure the profile on the client itself, which means defining all parameters from the command line. Thus, the profile information is stored in cache files and is never refreshed by the server. For example:

Use `ldapclient manual` to initialize the client.

```
# ldapclient manual \
-a domainName=dc=west.example.com \
-a credentialLevel=proxy \
-a defaultSearchBase=dc=west,dc=example,dc=com \
-a proxyDN=cn=proxyagent,ou=profile,dc=west,dc=example,dc=com \
-a proxyPassword=testtest 192.168.0.1
```

Use `ldapclient list` to verify.

```
# ldapclient list
NS_LDAP_FILE_VERSION= 2.0
NS_LDAP_BINDDN= cn=proxyagent,ou=profile,dc=west,dc=example,dc=com
NS_LDAP_BINDPASSWD= {NS1}4a3788e8c053424f
NS_LDAP_SERVERS= 192.168.0.1
NS_LDAP_SEARCH_BASEDN= dc=west,dc=example,dc=com
NS_LDAP_CREDENTIAL_LEVEL= proxy
```

/etc/nsswitch.conf File

The `/etc/nsswitch.conf` file must be configured for LDAP before enabling the LDAP client service. The template file for this is `/etc/nsswitch.ldap`. You can copy the file over the default `/etc/nsswitch.conf` file. To view the contents of the file, execute the following command:

```
# cat /etc/nsswitch.ldap
#
# Copyright 2006 Sun Microsystems, Inc. All rights reserved.
# Use is subject to license terms.
#
# ident "@(#)nsswitch.ldap      1.10      06/05/03 SMI"
#
# /etc/nsswitch.ldap:
#
# An example file that could be copied over to /etc/nsswitch.conf; it
# uses LDAP 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.  
  
# LDAP service requires that svc:/network/ldap/client:default be enabled  
# and online.  
  
# the following two lines obviate the "+" entry in /etc/passwd and  
/etc/group.  
passwd:      files ldap  
group:       files ldap  
  
# consult /etc "files" only if ldap is down.  
hosts:       ldap [NOTFOUND=return] files  
  
# Note that IPv4 addresses are searched for in all of the ipnodes  
databases  
# before searching the hosts databases.  
ipnodes:     ldap [NOTFOUND=return] files  
  
networks:    ldap [NOTFOUND=return] files  
protocols:   ldap [NOTFOUND=return] files  
rpc:         ldap [NOTFOUND=return] files  
ethers:      ldap [NOTFOUND=return] files  
netmasks:    ldap [NOTFOUND=return] files  
bootparams:  ldap [NOTFOUND=return] files  
publickey:   ldap [NOTFOUND=return] files  
  
netgroup:    ldap  
  
automount:   files ldap  
aliases:     files ldap  
  
# for efficient getservbyname() avoid ldap  
services:    files ldap  
  
printers:    user files ldap  
  
auth_attr:   files ldap  
prof_attr:   files ldap  
  
project:     files ldap  
  
tnrhttp:    files ldap  
tnrhdb:     files ldap  
#
```

Enabling DNS With LDAP

If you want to enable DNS by setting up a /etc/resolv.conf file, add DNS to your hosts lines as shown below.

```
hosts: ldap dns [NOTFOUND=return] files  
The recommended configuration is:  
hosts: files dns  
ipnodes: files dns
```

LDAP Service

The LDAP client service is managed by using the Service Management Facility. The LDAP service requires that

svc:/network/ldap/client:default be enabled and online. To check the state of the LDAP client service execute the following command:

```
# svcs -a | grep ldap  
disabled 15:26:07 svc:/network/ldap/client:default
```

To enable the service execute the following command:

```
# svcadm enable svc:/network/ldap/client:default
```

When you use the Service Management Facility's svcadm command to start the LDAP client, the ldap_cachemgr daemon is automatically invoked. The ldap_cachemgr daemon must be running and functioning correctly at all times. So, if the ldap_cachemgr is not running, the LDAP client will be disabled.

You can check to see if the ldap_cachemgr daemon is running by executing the following command:

```
# ps -ef | grep slapd  
root 25367 25353 0 15:35:19 pts/1 0:00 grep slapd
```

Following are two methods for determining if the LDAP client is online.

- Use the svcs command.

```
# svcs \*ldap\*  
STATE STIME FMRI  
disabled Aug_24 svc:/network/ldap/client:default
```

or

```
# svcs -l network/ldap/client:default
fmri svc:/network/ldap/client:default
enabled true
state online
next_state none
restarter svc:/system/svc/restarter:default
contract_id 1598
dependency require_all/none file://localhost/var/ldap/ldap_client_file (-)
dependency require_all/none svc:/network/initial (online)
dependency require_all/none svc:/system/filesystem/minimal (online)
```

- Pass the **-g** option to **ldap_cachemgr**

This option provides more extensive status information, which is useful when you diagnose a problem.

```
# /usr/lib/ldap/ldap_cachemgr -g
cachemgr configuration:
server debug level 0
server log file "/var/ldap/cachemgr.log"
number of calls to ldapcachemgr 19
cachemgr cache data statistics:
Configuration refresh information:
Previous refresh time: 2009/04/16 18:33:28
Next refresh time: 2009/04/16 18:43:28
Server information:
Previous refresh time: 2009/04/16 18:33:28
Next refresh time: 2009/04/16 18:36:08
server: 192.168.0.0, status: UP
server: 192.168.0.1, status: ERROR
error message: Can't connect to the LDAP server
Cache data information:
Maximum cache entries: 256
Number of cache entries: 2
```



Note – For more information on configuring a LDAP name service, refer to the System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP) at:

<http://docs.sun.com/app/docs/doc/816-4556>

Verifying Basic Client-Server Communication

The best way to show that your client is talking to the LDAP server is with the `ldaplist` command. Using `ldaplist` with no arguments dumps all the containers on the server. This works as long as the containers exist, and do not have to be populated. See the `ldaplist(1)` man page for more information.

If the first step works, you can try `ldaplist passwd username` or `ldaplist hosts hostname` but if they contain lots of data you might want to pick a less populated service, or pipe them to head or more.

Configuring JumpStart Installation Using the Solaris 10 Operating System

Objectives

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

- Describe the JumpStart configurations
- Implement a basic JumpStart server for SPARC® and x86/x64 clients
- Describe booting x86/x64 systems using the Preboot Execution Environment (PXE)
- Set up a DHCP server to support x86/x64 JumpStart clients
- Set up JumpStart software configuration alternatives
- Set up JumpStart to create a ZFS mirrored root pool
- Troubleshoot JumpStart configurations

Introducing JumpStart Configurations

JumpStart is an automatic installation process available in the Solaris 10 OS. JumpStart enables you to install the Solaris OS automatically and configure it differently, depending on the characteristics of client systems. JumpStart uses these identifying characteristics to select the correct configuration for each client system.

This module focuses first on the services and procedures required to create a JumpStart server for SPARC JumpStart clients. Creating a JumpStart server for x86/x64 JumpStart clients is described later in this module.

Purpose of JumpStart

System administrators who need to install multiple systems with similar configurations can use JumpStart to automate the installation process. JumpStart eliminates the need for operator intervention during the installation process.

The advantages of using JumpStart include the following:

- It enables system administrators to avoid the lengthy question-and-answer session that is part of the interactive installation process.
- It enables system administrators to install different types of systems easily.
- It allows automatic installation of the Solaris 10 OS and unbundled software.
- It simplifies administration tasks when widely used applications must be updated frequently.

JumpStart provides considerable time savings when multiple or ongoing installations are required for networked computing environments.

Four main services support the software installation process using JumpStart:

- Boot services
- Identification services
- Configuration services
- Installation services

Configuring JumpStart programs requires setting up these services on one or more networked servers. You can configure a single server to provide all four services for JumpStart, or you can configure the services separately on different servers.

Figure 12-1 Shows a typical JumpStart configuration.

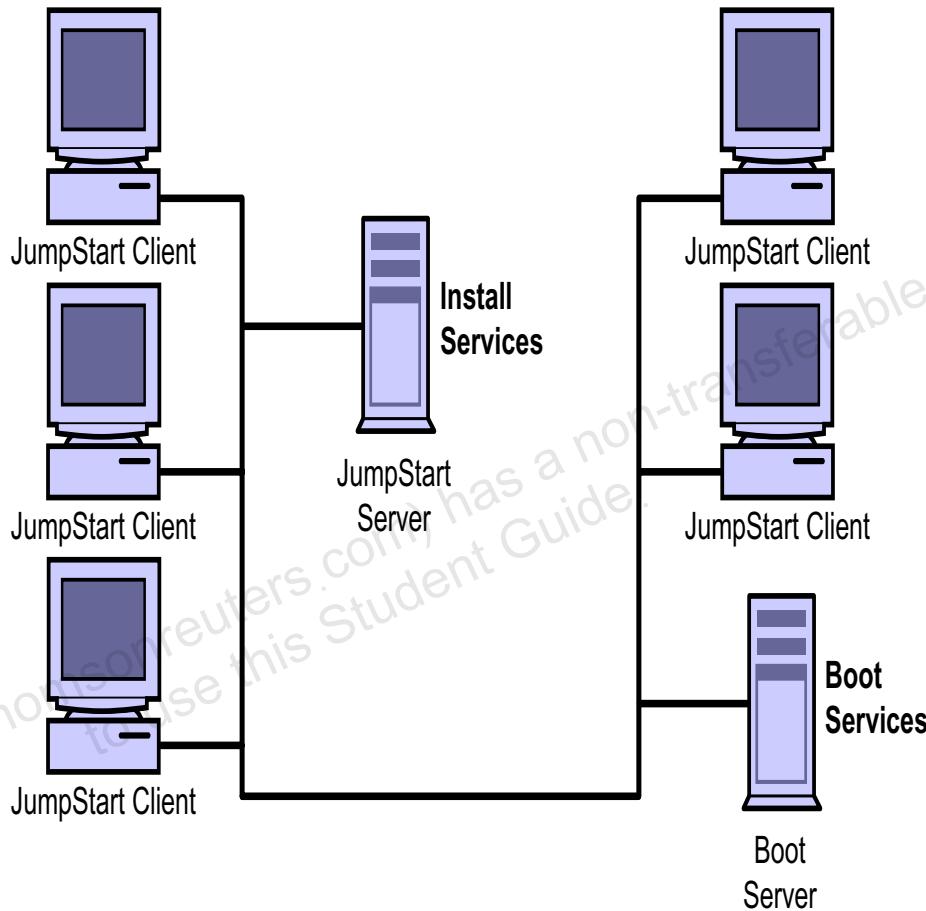


Figure 12-1 JumpStart Server Component Services

Boot Services

To boot a SPARC JumpStart client using the network, clients require support from a server that can respond to their Reverse Address Resolution Protocol (RARP), Trivial File Transfer Protocol (TFTP), and BOOTPARAMS requests. A system that provides these services is called a boot server. You can configure a boot server to provide any of the other required JumpStart services, or to only provide boot services.

If other servers provide identification, configuration, and installation services, the boot server identifies those servers for the JumpStart client. To support client RARP requests, the boot server must reside on the same subnet as the client, but the servers that provide these other services can reside on other network segments.



Note – For a comparison of SPARC and x86/x64 network boot processes associated with JumpStart installations, see the Sun BluePrint article called *Configuring JumpStart™ Servers to Provision Sun™ x86-64 Systems*, available through this URL:

<http://www.sun.com/blueprints/0205/819-1692.pdf>

In the most simple JumpStart configuration, for SPARC client boot operations to proceed, the following must be properly configured on the boot server:

- The /etc/ethers file
- The /etc/inet/hosts file
- The /tftpboot directory
- The /etc/bootparams file
- The /etc/dfs/dfstab file
- The TFTP service in SMF/INETD

The /etc/ethers and /etc/inet/hosts files configure the boot server to support RARP requests from SPARC JumpStart clients.



Note – In alternative JumpStart configurations, and for x86/x64 clients to boot, DHCP services on a boot server can supply boot and identification information that JumpStart clients require.

For each SPARC JumpStart client that the boot server supports, the /tftpboot directory must contain a symbolic link that points to a network bootstrap program. The inetd daemon must be configured to start the in.tftpd daemon on demand.

The boot server provides access to a boot image (a root (/) file system) that all JumpStart clients on the subnet use during the network boot process. The /etc/bootparams file lists the location of this root (/) file system and the locations of other directories that the JumpStart client requires. The /etc/dfs/dfstab file is used to configure JumpStart servers to share the directories that they provide through NFS.

You can configure boot services using the add_install_client script. The add_install_client script allows you to specify all of the information required in the files that support boot services. This script also creates the required files in the /tftpboot directory and appropriately modifies the inetd service configuration to support tftp requests.

Identification Services

JumpStart clients require support from a server to automatically get the answers to system identification questions that the client systems issue. The identification service is often provided by a boot server, but the service can be provided by any network server configured to provide identification.

JumpStart clients can obtain identification information from different sources, including:

- The /etc/inet/hosts file on the boot server
- The sysidcfg file,
- A name service such as:
 - LDAP (Lightweight Directory Access Protocol)
 - NIS (Network Information Service)
 - NIS+ (Network Information Service Plus)
- DHCP configuration information

You can use a combination of these sources to answer the client's identification requests. Identification information provided in a sysidcfg file overrides information provided by other sources.

Configuring a server to provide identification services is, for the most part, a manual process. You must manually edit the sysidcfg file, and share the directory where it resides. During the installation process, JumpStart clients use the Network File System (NFS) service to mount the directory that contains the sysidcfg file.

If you use a name service, configuring identification services involves updating the name service in the appropriate way. This may involve editing the source files and running commands to update the name service.

You can establish and manage DHCP configuration information on a boot server using the /usr/sadm/admin/bin/dhcpmgr GUI utility.

If the JumpStart client cannot obtain a response from a server for any identification item, the client interrupts the automatic identification process and asks for the information.

Listing Identification Items and Their Sources

Table 12-1 lists the identification items that JumpStart clients using SPARC® technology require, and also lists the sources in the Solaris 10 Operating System that can provide the information. In earlier releases of the Solaris Operating System, the list of items and usable sources sometimes differed.

Table 12-1 JumpStart Client Identification Items

| Identification Item | Configurable With the <code>sysidcfg</code> File? | Configurable With a Name Service? |
|--|--|--|
| Name service | Yes | Yes |
| Domain name | Yes | No |
| Name server | Yes | No |
| Network interface | Yes | No |
| Host name | Yes | Yes |
| IP address | Yes | Yes |
| Netmask | Yes | Yes |
| Dynamic Host Configuration Protocol (DHCP) | Yes | No |
| Internet Protocol Version 6 (IPv6) | Yes | No |
| Default router | Yes | No |
| Root password | Yes | No |
| Security policy | Yes | No |

Table 12-1 JumpStart Client Identification Items (Continued)

| Identification Item | Configurable With the sysidcfg File? | Configurable With a Name Service? |
|----------------------------------|---|---|
| Locale | Yes | Yes if NIS or NIS+, No if DNS or Lightweight Directory Access Protocol (LDAP) |
| Terminal Type | Yes | No |
| Time zone | Yes | Yes |
| Date and time | Yes | Yes |
| Power management (auto shutdown) | No | No |
| NFSv4 Domain | Yes | No |

For more information, refer to the Solaris 10 Release and Installation Collection online at <http://docs.sun.com>

Configuration Services

JumpStart clients require support from a server to obtain answers for system configuration questions that they issue. A system that provides this service is called a configuration server.

A configuration server provides information that specifies how the Solaris Operating System installation proceeds on the JumpStart client. Configuration information can include:

- Installation type
- System type
- Disk partitioning and file system specifications
- Software group selection
- Software package additions or deletions

On the configuration server, files known as *profile* files store client configuration information. A file called *rules.ok* on the configuration server allows JumpStart clients to select an appropriate profile file.

Associating a Configuration With a Client

A configuration server shares a directory, for example the /export/config directory, that minimally contains the files shown in Table 12-2.

Table 12-2 Files in the /export/config Directory

| File | Description |
|-----------------------------------|---|
| The rules file | <p>The rules file associates classes of clients with specific installation profiles. Classes in the rules file are identified using predefined keywords that include:</p> <ul style="list-style-type: none"> • hostname • arch • domainname • memsize • model <p>Clients select a profile by matching their own characteristics with an entry in the rules file.</p> |
| The profile (class) files | The profile files specify how the installation is to proceed and what software is to be installed. A separate profile file can exist for each class of JumpStart client on your network. |
| The check script | Run the check script after creating the rules and profile files. The check script verifies the syntax in the rules and profile files. If there are no syntax errors, the check script creates the rules.ok file. |
| The rules.ok file | The check script creates the rules.ok file from the rules file. The JumpStart installation procedure reads the rules.ok file during the automatic installation process (the rules file is not read). |
| Optional begin and finish scripts | The JumpStart client uses begin and finish scripts to perform preinstallation and postinstallation tasks. You can use these scripts to further customize the installation process, such as configuring power management on the JumpStart client. The begin and finish scripts are located in the configuration directory hierarchy shared by the configuration server. |

Installation Services

JumpStart clients require support from a server to find an image of the Solaris OS to install. A system that provides this service is called an install server. An install server shares a Solaris OS image from a CD-ROM, DVD, or local disk. JumpStart clients use the NFS service to mount the installation image during the installation process.

Sources of the Operating System Image

An install server provides the Solaris Operating System image by sharing one of the following:

- The Solaris 10 OS Software DVD
- The Solaris 10 OS Software 1 CD-ROM
- A spooled image of the Solaris 10 Operating System obtained from either the CD-ROM or DVD media
- A Flash installation image

CD-ROM or DVD

An install server can provide installation services by sharing either the Solaris 10 OS Software DVD or the Solaris 10 OS Software 1 CD-ROM.

The Solaris 10 OS Software DVD and the Solaris 10 OS Software 1 CD-ROM both contain a boot image and an installation image. Sharing either of these supports both boot services and installation services.

The installation image found on the Solaris 10 OS Software 1 CD-ROM only supports installing the Core (SUNWCreq) and Reduced Networking (SUNWCMreq) software groups. The Solaris 10 OS Software 2, 3, and 4 CD-ROMs contain the remainder of the installation image, but there is no support for changing CD-ROMs in the middle of a JumpStart installation procedure.

The Spooled Image

An install server can provide installation services by sharing a spooled image on a local disk. When you spool the Solaris Operating System image from CD-ROM or DVD, the result is a directory that contains the boot image and the installation image:

| | |
|------------------------|--|
| The boot image | JumpStart clients can boot from the root (/) file system contained in the boot image. For example, if you spool the Solaris 10 OS into a directory called /export/install, the boot image would be located in the /export/install/Solaris_10/Tools/Boot directory. |
| The installation image | JumpStart clients install the Solaris Operating System from the installation image. For example, if you spool the Solaris 10 Operating System into a directory called /export/install, the installation image would be located in the /export/install/Solaris_10/Product directory. |

The `setup_install_server` script enables you to spool the boot and installation images from the Solaris OS 1 CD-ROM or from the DVD.

The `add_to_install_server` script enables you to spool additional installation image data from CD-ROMs 2, 3, and 4.

The `setup_install_server` script with the `-b` option enables you to spool only the boot image from the Solaris OS 1 CD-ROM or from the DVD. The script supports creating a boot image on a boot server. The boot server would then be configured to direct the JumpStart client to a separate install server for the installation image.

A Flash Install Image

Flash installation is significantly faster than the current JumpStart installation or a network installation method. Flash allows detailed customization of the Solaris Operating System, hardware configuration, and third-party software packages prior to creation of the clones. In addition, Flash installation can provide enterprise-level disaster recovery when necessary.

Implementing a Basic JumpStart Server

A JumpStart server configuration includes:

- A single server that provides boot, identification, configuration, and installation services
- Boot and installation services provided by the Solaris 10 OS boot and installation images spooled to the local disk of the server
- Identification services provided by files on the server and a sysidcfg file, with no name service in place
- Configuration services provided by a rules file that contains an entry for a single JumpStart client, and a profile file that installs the entire Solaris 10 OS distribution into a set of slices on the JumpStart client

The following tasks are required to configure a single JumpStart server to provide basic software installation services using JumpStart:

1. Spool the operating system image.
2. Edit the sysidcfg file.
3. Edit the rules and profile files.
4. Run the check script.
5. Run the add_install_client script.
6. Boot the client.

Spooling the Operating System Image

Spooling the Solaris OS boot and installation image to disk is the most common method of supplying boot and installation services to JumpStart clients. You can spool the boot image and installation image to different servers. This section describes how one server provides both boot and installation services.

When you use the Solaris 10 CD-ROM source media, you must use the setup_install_server script to spool the Solaris 10 OS image from the Solaris 10 OS Software 1 CD-ROM, and then use the add_to_install_server script to spool the Solaris 10 OS image from the remaining CD-ROMs.

The Solaris 10 OS Software CD-ROM provides the boot image and the required portion of the installation image to install the Core (SUNWCreq) and Minimal Network (SUNCmreq) software groups. The remaining CD-ROMs provide the rest of the installation image, containing the data required to install the Minimal Core Metacluster (SUNWCmreq), End User (SUNWCuser), Developer (SUNWCprog), Entire Distribution (SUNWCall), and the Entire Distribution with OEM Support software group (SUNWCXall).

When you use the Solaris 10 DVD source media, you use the `setup_install_server` script to spool the entire Solaris 10 OS boot image and complete installation image to disk. All of the software associated with all software groups is included on the DVD source media.

When the spooling procedure is complete, the server has the data available to support boot and installation services for JumpStart clients. The spooled image also contains the `add_install_client` script that lets you configure boot and installation support for specific JumpStart clients.

To spool the Solaris 10 OS boot and installation images to a local disk, complete the following steps:

1. Create a directory with at least five Gbytes of space available to hold the Solaris OS image. Typically, the `/export/install` directory is used.
`# mkdir /export/install`
2. Insert the Solaris 10 OS DVD in the DVD drive or the Solaris 10 OS Software 1 CD-ROM in the CD-ROM drive. Allow the `vold` daemon to automatically mount the media.
3. Change the directory to the location of the `setup_install_server` script.
`# cd /cdrom/cdrom0/Solaris_10/Tools`

Note – Solaris releases for SPARC prior to Solaris 10 5/09 will use the path: `/cdrom/cdrom0/s0/Solaris_10/Tools`
Newer releases do not have the `s0` slice identifier.

4. Run the `setup_install_server` script to copy the Solaris 10 OS boot and installation images to the local disk (this process can take about an hour for CD-ROM media, and about two hours for DVD media).
`# ./setup_install_server /export/install`



Note – No specific requirement exists to use the /export/install directory to spool the Solaris image. If you have more than one Solaris image to spool to the same JumpStart server (for example, one for SPARC-based systems, and another for x86/x64-based systems) each requires its own directory. In this case, you may choose to spool the SPARC image to /export/install/S10_sparc, and the x86/x64 image to /export/install/S10_x86. You must specify the correct spool directory when it is required by JumpStart-related configuration commands.

5. When the `setup_install_server` script finishes, change directory to root (/), and eject the CD-ROM or DVD.

```
# cd /
# eject cdrom
```

6. If you are using CD-ROM media, insert the Solaris 10 OS Software 2 CD-ROM in the CD-ROM drive, and allow the `vold` daemon to automatically mount it.
 - a. Change the directory to the location of the `add_to_install_server` script.
 - b. Run the `add_to_install_server` script to copy the remainder of the installation image to the local disk (this process can take about 20 minutes).
7. `# cd /cdrom/cdrom0/Solaris_10/Tools`
- c. When `add_to_install_server` finishes, change the directory to root (/), and eject the CD-ROM.

```
# ./add_to_install_server /export/install
```

7. Repeat step 6 for the remaining CD-ROMs.

Note – The same procedure is used if the Language CD-ROM is required.



Editing the sysidcfg File

JumpStart clients use information in the sysidcfg file to answer system identification questions. If the JumpStart client cannot obtain a response for an identification question, the client interrupts the automatic identification process and asks for the information.

To provide complete identification services in the absence of a name service, the JumpStart server must provide information in the sysidcfg file that answers the following questions:

- What netmask will the client use?
- Will the client be configured to use IPv6 networking?
- What is the Internet Protocol (IP) address of the default router?
- What security policy will the client implement?
- What name service will the client use?
- What time zone will the client use?
- What system locale (region/country) will the client use?
- What system will provide the initial time-of-day information?
- What is the root user's password?

The sysidcfg file can contain:

- Identification information that all JumpStart clients can use
- Information that is client-specific

Locating the sysidcfg File

The sysidcfg file cannot be given any other name. For example, you would create a generic sysidcfg file in the /export/config directory on a JumpStart server. If you require sysidcfg files that contain client-specific information, each one must exist in a separate directory. For example, you could place the sysidcfg file for a client called client1 in the /export/config/client1 directory.

JumpStart clients learn of the location of the sysidcfg file from the BOOTPARAMS information that they obtain from the boot server. When you run the add_install_client script on the boot server, use the -p option, and specify the server and path where the sysidcfg file is stored. The following command indicates that the sysidcfg file that client1 uses is found on the server, server1 in the /export/config directory.

```
# ./add_install_client -c server1:/export/config -p  
server1:/export/config client1 sun4u
```

The server, server1, must share the /export/config directory using the NFS service before the client can mount it.



Note – Other options to the add_install_client command are discussed later in this module.

Constructing the sysidcfg File

The sysidcfg file lets you specify many different identification items. Entries in the sysidcfg file must conform to the following rules:

- Independent keywords can be listed in any order.
- Keywords are not case sensitive.
- Keyword values can be optionally enclosed in single (') or double (") quotation marks.
- Dependent keyword values must be enclosed in curly braces ({}) to tie them to their associated independent keyword.
- Only the first instance of a keyword is valid. If a keyword is specified more than once, only the first keyword specified is used.

Table 12-3 lists the keywords and arguments used to construct the sysidcfg file.

Table 12-3 Keywords and Arguments Used to Construct the sysidcfg File

| Keywords | Arguments |
|--|---|
| name_service {domain_name} | <p>name_service=NIS, NIS+, DNS, LDAP, OTHER, NONE</p> <p>Options for NIS and NIS+: {domain_name=domain_name name_server=hostname(ip_address)}</p> |
| | <p>Options for DNS: {domain_name=domain_name name_server=ip_address, ip_address, ip_address (three maximum) search=domain_name, domain_name, domain_name, domain_name, domain_name, domain_name (six maximum, the total length is less than or equal to 250 characters)}</p> |
| | <p>Options for LDAP: {domain_name=domain_name profile=profile_name profile_server=ip_address}</p> |
| network_interface, hostname, ip_address, netmask | <p>network_interface=primary or value (eg: bge0) {primary hostname=hostname ip_address=ip_address netmask=netmask protocol_ipv6=yes/no}</p> <p>If DHCP <i>is</i> used, specify: {dhcp protocol_ipv6=yes/no}</p> <p>If DHCP <i>is not</i> used, specify: {hostname=host_name default_route=ip_address ip_address=ip_address netmask=netmask protocol_ipv6=yes/no}</p> |

Table 12-3 Keywords and Arguments Used to Construct the sysidcfg File (Continued)

| Keywords | Arguments |
|-----------------|--|
| nfs_domain | To enables the NFSv4 domain to be derived dynamically, at run time, based on naming service configuration: nfs_domain= <i>dynamic</i> Or the value can be a fully qualified domain name, as per RFC1033 and RFC1035 recommendations: nfs4_domain=example.com |
| root_password | root_password= <i>root_password</i> (encrypted password from /etc/shadow) |
| security_policy | security_policy=kerberos, NONE Options for kerberos: {default_realm= <i>FQDN</i> admin_server= <i>FQDN</i> kdc= <i>FQDN1, FQDN2, FQDN3</i> } where <i>FQDN</i> is a fully qualified domain name. 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 the /usr/lib/locale file) |
| terminal | terminal= <i>terminal_type</i> (entry from the /usr/share/lib/terminfo database) for the installation. |
| timezone | timezone= <i>timezone</i> (entry from /usr/share/lib/zoneinfo file) |
| timeserver | timeserver=localhost, <i>hostname</i> , or <i>ip_addr</i> |

Implementing a Basic JumpStart Server

To configure a generic sysidcfg file on a JumpStart server, complete the following steps:

1. Create a directory to hold the sysidcfg file. Typically the /export/config directory holds the sysidcfg file.

```
# mkdir /export/config
```

2. Change the directory to /export/config, and create a file called sysidcfg using a text editor.

```
# cd /export/config  
# vi sysidcfg
```

3. In the sysidcfg file, add the following lines. Substitute values that are appropriate for your systems, location, and network.

```
network_interface=bge0 { primary  
                      protocol_ipv6=no  
                      netmask=netmask_value  
                      default_route=router_IP}  
  
security_policy=none  
name_service=none  
timezone=timezone  
system_locale=locale  
timeserver=timeserver_IP  
root_password=Hx23475vABDDM  
nfs4_domain=dynamic
```

- a. For the *netmask_value*, enter the correct netmask for your network.
- b. For the *router_IP* value, enter the IP address of the system that will act as your default router, or *none* if no router is to be specified.
- c. For the *timezone* value, enter the correct time zone for your location. Time zones are listed in the directory structure below the /usr/share/lib/zoneinfo directory. For example, the US/Mountain time zone refers to the /usr/share/lib/zoneinfo/US/Mountain directory.
- d. For the *locale* value, enter the correct system locale for your location. Locales are listed in the /usr/lib/locale directory.
- e. For the *timeserver_IP* value, enter the IP address of the system that provides the time-of-day to the JumpStart client. If you specify localhost as the time server, the system's time is assumed to be correct and the installation procedure does not prompt for the date and time.

- f. For the nfs4_domain value, enter dynamic to enable the NFSv4 domain to be derived dynamically, at run time, based on naming service configuration. This eliminates the need to run a finish script to set this setting as was done in earlier releases of Solaris 10.
- g. Save the sysidcfg file, and exit your edit session.

The following example shows entries in a sysidcfg file for a JumpStart client with a single bge0 network interface:

```
network_interface=bge0 { primary_protocol_ipv6=no  
                         netmask=255.255.255.0  
                         default_route=192.10.10.100 }  
  
security_policy=none  
name_service=none  
timezone=US/Mountain  
system_locale=en_US  
timeserver=192.10.10.100  
root_password=Hx23475vABDDM  
nfs4_domain=dynamic
```

Editing the rules and Profile Files

In order to provide configuration services, the JumpStart server provides a file called `rules.ok` that allows the JumpStart client to select a profile file. To create a `rules.ok` file, you edit a file called `rules`, and run a script called `check`. The `check` script creates the `rules.ok` file from the information you place in the `rules` file.

The `rules.ok` file enables groups of clients with the same characteristics to be grouped together as a *class*. Sometimes profile files are also referred to as class files. A profile file must be named to match the profile file name you declare in the `rules.ok` file. The terms *profile file* or *class file* are used to generically identify these files in this description.

The profile file must contain all the information normally provided during interactive installation about the disk partitioning and software selections for the JumpStart client. If the JumpStart client cannot obtain a response from a server for any configuration item, the client interrupts the automatic configuration process and asks for the information.

Each entry in the `rules.ok` file lists one or more identifying characteristics that JumpStart clients can match. When a client finds an entry in `rules.ok` that it matches, it uses the profile file associated with that entry. Clients use only the first entry in the `rules.ok` file that they match.

If a JumpStart client checks all the entries in `rules.ok` but does not find a match, the client begins an interactive configuration session.

The rules File Syntax

Entries in the `rules` file conform to the following syntax:

```
[!] match_key match_value [&& [!] match_key match_value]* \  
begin profile finish
```

where:

| | |
|------------------|---|
| <i>match_key</i> | A predefined keyword that describes an attribute of the system being installed. The keyword can be: <code>any</code> , <code>hostname</code> , <code>model</code> , <code>arch</code> , <code>installed</code> , <code>network</code> , <code>domainname</code> , <code>karch</code> , <code>totaldisk</code> , <code>disksize</code> , or <code>memsize</code> . |
|------------------|---|

| | |
|--------------------|--|
| <i>match_value</i> | The value (or range of values) selected by the system administrator for the <i>match_key</i> . You can use multiple keywords in a rule. Join multiple keywords with the logical AND symbol, (&&). You can use the logical NOT symbol (!) in front of a keyword to express negation. In other words, to express that the install client's value for <i>match_key</i> does not equal the <i>match_value</i> specified in the rule. |
| <i>begin</i> | The name of a begin script. This is a Bourne Shell script to be run before the installation is started. Use a (-) to indicate that no begin script runs. |
| <i>profile</i> | The name of the profile (class) file. |
| <i>finish</i> | The name of a finish script. This Bourne Shell script runs after the installation is completed. Use a (-) to indicate that no finish script runs. |

The example:

```
hostname client1 - profile1 -
```

causes a JumpStart client called *client1* to use a profile file called *profile1*. The dash (-) characters before and after the *profile1* file name indicate that the *client1* system does not run a begin or a finish script, respectively.

To configure a simple rules and profile file on a JumpStart server, complete the following steps:

1. Create a directory to hold the rules file if this directory does not already exist. Usually, the /export/config directory holds the rules file.

```
# mkdir /export/config
```

2. Change the directory to /export/config, and create a file called rules using a text editor.

```
# cd /export/config
```

```
# vi rules
```

3. In the rules file, add the following line. For *client_name*, substitute the name of your JumpStart client.

```
hostname client_name - profile1 -
```

4. Save the rules file, and exit your edit session.

5. Create a profile file called `profile1` by using a text editor.

```
# vi profile1
```

Add the following lines to the `profile1` file:

```
install_type    initial_install
system_type     standalone
partitioning    explicit
filesys         cxtxdxs1 512      swap
filesys         cxtxdxs0 free     /
cluster          SUNWCXall
```

- a. For `cxtxdxs0`, enter the correct designation for slice 0 on the JumpStart client's boot disk.
 - b. For `cxtxdxs1`, enter the correct designation for slice 1 on the JumpStart client's boot disk.
6. Save the `profile1` file, and exit your edit session.

For example, a simple profile file can contain the following information:

```
install_type    initial_install
system_type     standalone
partitioning    explicit
filesys         c0t0d0s0  free     /
filesys         c0t0d0s1  512      swap
cluster          SUNWCXall
package          SUNWman delete
```

This profile file declares that the JumpStart client performs an initial installation as a standalone system, uses partitioning that allocates 512 Mbytes to the swap area, allocates the remainder of the disk space to the root (/) file system, the client installs the Entire Distribution with OEM support software group, and then removes the man pages.

Running the check Script

Before a JumpStart client can use a configuration provided by a JumpStart server, you must run the check script to produce the file called `rules.ok`. The check script validates the syntax of the `rules` file and its associated profile files. If the validation completes successfully, the check script creates the `rules.ok` file which the JumpStart clients require.

This procedure assumes that the rules and profile file that you intend to use exist in the /export/config directory, and that the Solaris 10 OS has been spooled below the /export/install directory. To run the check script on a JumpStart server, complete the following steps:

1. Change the directory to the location of the check script.

```
# cd /export/install/Solaris_10/Misc/jumpstart_sample
```

2. Copy the check script to the /export/config directory.

```
# cp check /export/config
```

3. Change the directory to /export/config, and run the check script.

```
# cd /export/config
```

```
# ./check
```

Validating rules...

Validating profile profile2...

The custom JumpStart configuration is ok.

```
#
```

4. If the check script reports an error, edit the rules or profile file to correct the problem indicated. In the following example, the profile2 file contains a spelling error. For the example, the misspelling of the keyword, fileys, causes the check script to report the following output:

Validating rules...

Validating profile profile2...

Error in file "profile2", line 4

 fileys c0t0d0s0 free /

ERROR: Invalid keyword

5. Once the rules or profile file have been edited to correct any errors, run the check script again.

```
# cd /export/config
```

```
# ./check
```

Validating rules...

Validating profile profile1...

The custom JumpStart configuration is ok.

```
#
```

Running the add_install_client Script

The `add_install_client` script configures the boot server to provide the network boot services that JumpStart clients require. Options to the `add_install_client` script also let you specify what servers and what directories offer identification, configuration, and installation services. The options and arguments to `add_install_client` that you use must reflect the JumpStart server configuration decisions you have made.

Before you run the `add_install_client` script, edit the `/etc/inet/hosts` and `/etc/ethers` files on the boot server, and add a JumpStart client entry to each file. The following example shows how an entry for `client1` in the `/etc/inet/hosts` file appears:

```
192.10.10.4    client1
```

An entry for `client1` in `/etc/ethers` could appear as follows:

```
8:0:20:1c:88:5b client1
```

Note – The `add_install_client` script must be run from the directory where the installation image or boot image resides.

The `add_install_client` script options and arguments must match how you have configured the services on the servers that you intend to use. In the following example, one server provides all the services for JumpStart. Run the `add_install_client` script only on the server that provides the boot image.

You must run the `add_install_client` script once for each JumpStart client.

For this basic JumpStart configuration procedure, the `add_install_client` script requires that you specify the following information:

- The server and path where the rules and profile files are located (the `-c` option)
- The server and path where the `sysidcfg` file is located (the `-p` option)
- The installation server
- The name of the client
- The kernel architecture of the client

The following procedure assumes that the Solaris 10 OS boot and installation images have been spooled below the /export/install directory, that the rules, profile, and sysidcfg files you intend to use exist in the /export/config directory, and that you are registering a SPARC JumpStart client. To run the add_install_client script on a JumpStart server, complete the following steps:

1. Edit the /etc/inet/hosts file, and add an entry for the JumpStart client.
2. Edit the /etc/ethers file, and add an entry for the JumpStart client.
3. Change the directory to the location of the add_install_client script on the server.

```
# cd /export/install/Solaris_10/Tools
```

The following example supplies the required information for a SPARC client called client1:

```
# ./add_install_client -c server1:/export/config -p
server1:/export/config client1 sun4u
saving original /etc/dfs/dfstab in /etc/dfs/dfstab.orig
Adding "share -F nfs -o ro,anon=0 /export/install" to /etc/dfs/dfstab
making /tftpboot
enabling tftp in /etc/inetd.conf
starting rarpd
starting bootparamd
starting nfssd's
starting nfs mountd
updating /etc/bootparams
copying inetboot to /tftpboot
#
```

The add_install_client script automatically makes the changes required to support RARP, TFTP, the bootparams file, and NFS requests from the client, but it only causes the server to share the installation directory. Sharing the installation directory allows the JumpStart client to mount a root (/) file system during the network boot process, and to gain access to the installation image.



Note – The following example shows that for the client to mount the configuration directory from the server, you must manually edit the /etc/dfs/dfstab file and add an entry to share the configuration directory: share -o ro /export/config

Implementing a Basic JumpStart Server

This line in the /etc/dfs/dfstab file would share the /export/config directory as a read-only directory.

```
share -o ro /export/config
```

- Run the svcs command to check that NFS services are enabled.

```
# svcs -a |grep nfs
STATE          STIME      FMRI
disabled       14:56:34  svc:/network/nfs/mapid:default
disabled       14:56:34  svc:/network/nfs/cbd:default
disabled       14:56:36  svc:/network/nfs/server:default
online         14:56:56  svc:/network/nfs/status:default
online         14:56:57  svc:/network/nfs/nlockmgr:default
online         14:57:13  svc:/network/nfs/client:default
online         14:57:13  svc:/network/nfs/rquota:ticlts
online         14:57:13  svc:/network/nfs/rquota:udp
```

- Use the svcadm command to enable the NFS services if required:

```
# svcadm enable network/nfs/server:default
```

- Check that the NFS service is online.

```
# svcs -a |grep nfs
STATE          STIME      FMRI
disabled       14:56:34  svc:/network/nfs/cbd:default
online         14:57:13  svc:/network/nfs/client:default
online         16:01:13  svc:/network/nfs/status:default
online         16:01:13  svc:/network/nfs/nlockmgr:default
online         16:01:14  svc:/network/nfs/mapid:default
online         16:01:14  svc:/network/nfs/rquota:ticlts
online         16:01:15  svc:/network/nfs/server:default
online         16:01:15  svc:/network/nfs/rquota:udp
bash-2.05b#
```

- Verify that the /export/config and /export/install directories are currently shared.

```
# share
-
/export/install    ro,anon=0    ""
-
/export/config     ro      ""
```

Booting a SPARC JumpStart Client

After the JumpStart server has been configured to provide all of the required services, you can initiate the installation process on the SPARC JumpStart client.

To boot the SPARC JumpStart client, perform the following steps:

1. Bring the JumpStart client to run state 0.

```
# init 0
```

2. Boot the client to initiate the software installation using JumpStart. Use the **nowin** option to use the text-only installation to allow viewing all errors that may occur.

```
ok boot net - install nowin
```

Introducing the SPARC JumpStart Client Boot Sequence

To understand the services that a boot server provides, it is useful to know how a SPARC JumpStart client boots using the network, as shown in Figure 12-2.

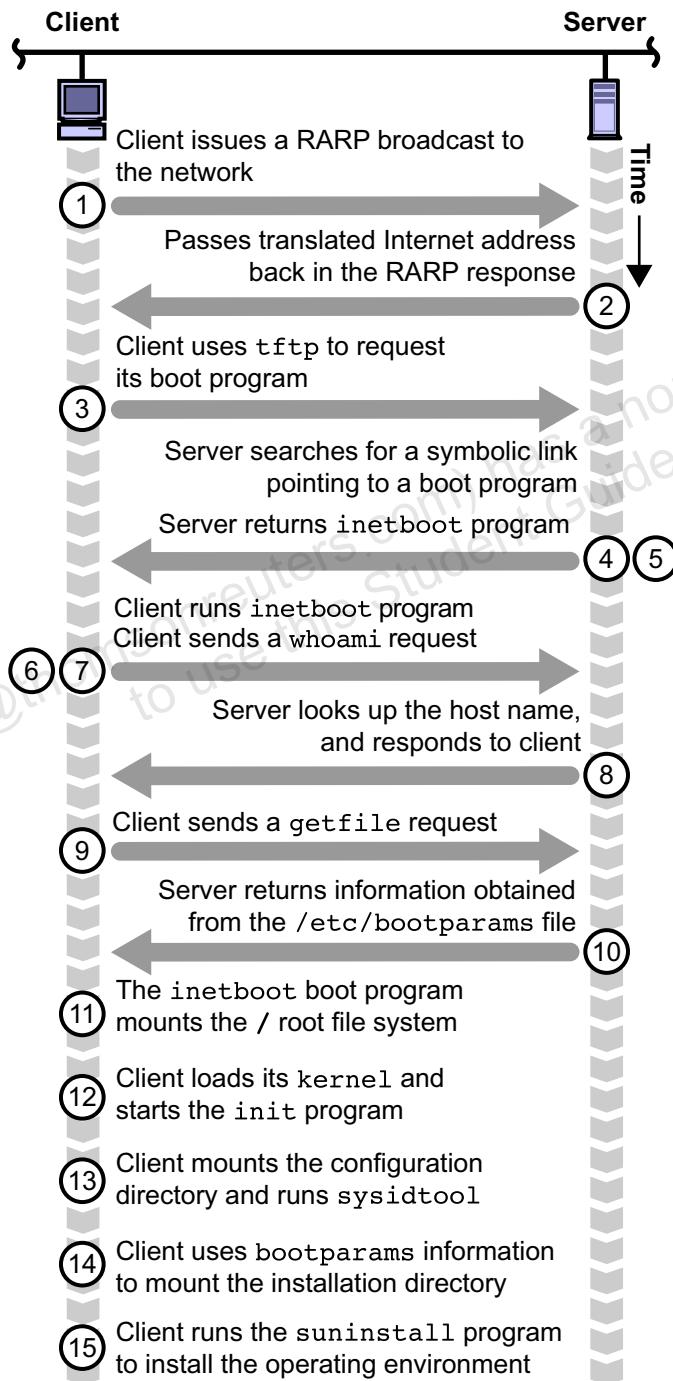


Figure 12-2 The JumpStart Boot Process

Figure 12-2 on page 12-28 shows the JumpStart client boot process. The following steps describe how a SPARC JumpStart client boots from a boot server, and starts the installation process:

1. When a SPARC JumpStart client boots, the boot PROM broadcasts a RARP request to the local subnet.
2. The `in.rarpd` daemon on the boot server processes the client's RARP request by:
 - a. Looking up the client's Ethernet address and host name in the `/etc/ethers` file
 - b. Checking for a corresponding host name in the `/etc/hosts` file
 - c. Returning the associated IP address to the client
3. The client's boot programmable read-only memory (PROM) sends a TFTP request for a network bootstrap program.
4. The `in.tftpd` daemon on the boot server processes the client's TFTP request. The daemon searches the `/tftpboot` directory for a file with a hexadecimal representation of the client's IP address. The hexadecimal representation is the name of the file. This file is a symbolic link that points to a network bootstrap program.
5. The `in.tftpd` daemon on the boot server returns the network bootstrap program to the JumpStart client.
6. The JumpStart client runs the network bootstrap program.
7. The network bootstrap program issues a `whoami` request to discover the JumpStart client's host name.
8. The `rpc.bootparamd` daemon on the boot server looks up the client's host name, and returns it to the client.
9. The network bootstrap program issues a `getfile` request to obtain the location of the root (/) file system.
10. The server responds with the location of the root (/) file system, obtained from the appropriate source:
 - The `/etc/bootparams` file.
 - A name service such as LDAP, NIS+, NIS.
11. After the client obtains its boot parameters, the network bootstrap program mounts the root (/) file system from the boot server.
12. The client loads its kernel and starts the `init` program. When the JumpStart client finishes booting, it attempts to find configuration information.

13. The client searches for the configuration server using BOOTPARAMS information. The client mounts the configuration directory, and runs the sysidtool daemon.
14. The client uses BOOTPARAMS information to locate and mount the Solaris Operating System installation image.
15. The client runs the suninstall program and installs the Solaris Operating System.

For boot operations to continue, the following files and directories must be properly configured on the boot server:

- The /etc/ethers file
- The /etc/inet/hosts file
- The /tftpboot directory
- The /etc/bootparams file
- The /etc/dfs/dfstab file

Note – For a comparison of SPARC and x86/x64 network boot processes associated with JumpStart installations, see the Sun BluePrint article called *Configuring JumpStart™ Servers to Provision Sun™ x86-64 Systems*, available through this URL:

<http://www.sun.com/blueprints/0205/819-1692.pdf>

The /etc/ethers and /etc/inet/hosts Files

A SPARC JumpStart client initially obtains its IP address through a RARP request while it boots. To obtain the RARP request, an entry for the client must exist in the /etc/ethers and /etc/inet/hosts files on the boot server.

Generally, you configure this information by editing these files manually, and by updating the name service, if one is in place. With this information available in either the /etc/ethers and /etc/inet/hosts files or in a name service, such as NIS or NIS+ on a boot server, the JumpStart client should be able to obtain the IP address and host name it needs to continue the boot process.



The /tftpboot Directory

SPARC JumpStart clients retrieve a network bootstrap program from the /tftpboot directory when they issue requests to the in.tftpd daemon running on the boot server. The in.tftpd daemon uses a symbolic link that is a hexadecimal representation of the client's IP address. This symbolic link locates a network bootstrap program to return to the /tftpboot directory. Different network bootstrap programs exist for different Solaris Operating System releases and client architectures.

In the following example, the symbolic link called C00A0A04 points to the network bootstrap program called inetboot.SUN4U.Solaris_10-1.

```
# cd /tftpboot
# ls -l
total 280
lrwxrwxrwx  1 root      other            26 Nov 110 17:31 C00A0A04 ->
inetboot.SUN4U.Solaris_10-1
```

The add_install_client script creates the required files in the /tftpboot directory when you run it to configure boot support for a JumpStart client. The platform group argument that you specify to the add_install_client script selects the bootstrap program appropriate for the client's kernel architecture. Running the add_install_client script from a Solaris 10 OS image automatically selects a bootstrap program specific to the Solaris 10 OS.



Note – Use the bc utility for a quick conversion from IP numbers to hexadecimal numbers. Run the bc utility, and press the Return key. Then enter obase=16. Enter each of the IP fields, one at a time, to get the hexadecimal conversion. Thus, 192 = C0, 10 = 0A, 10 = 0A, and 4 = 04. Putting it all together, the resultant hexadecimal IP number is C00A0A04. Press <Control-D> to exit the bc utility.

Describing the /etc/bootparams File

SPARC JumpStart clients retrieve information from the network when they issue requests to the `rpc.bootparamd` daemon that runs on the boot server. The `rpc.bootparamd` daemon references either:

- The `/etc/bootparams` file
- A naming service such as LDAP, NIS+, NIS

and returns the information to the client. The client system uses this information to mount the directories that it requires using the NFS service.

The `add_install_client` script updates the `/etc/bootparams` file when you run it to configure boot support for a JumpStart client. The `/etc/bootparams` file contains one entry for each JumpStart client that the boot server supports. Each entry lists the servers and directories that provide boot, identification, configuration, and installation services.

The options and arguments that you specify when you run the `add_install_client` script determine the content of the `/etc/bootparams` file. The following example describes an example entry in the `/etc/bootparams` file for a JumpStart client named `client1`:

```
client1
root=server1:/export/install/Solaris_10/Tools/Boot
install=server1:/export/install
boottype=:in
sysid_config=server1:/export/config
install_config=server1:/export/config
rootopts=:rsize=32768
```

The `add_install_client` command that creates the `/etc/bootparams` entry in the following example is:

```
# cd /export/install/Solaris_10/Tools
# ./add_install_client -c server1:/export/config -p
server1:/export/config client1 sun4u
```

Table 12-4 describes the example entries in the /etc/bootparams file.

Table 12-4 Entries in the /etc/bootparams File

| Entry | Definition |
|--|---|
| client1 | Specifies the JumpStart client name. |
| root=server1:/export/install/Solaris_10/Tools/Boot | Lists the boot server name and directory where the root (/) file system is found. This path is derived from the server and directory where you run the add_install_client script. |
| install=server1:/export/install | The server name and directory where the Solaris Operating System installation image is found. Unless you use the -s option, this path is derived from the server and directory where you run the add_install_client script. |
| boottype=:in | Indicates that client1 is an install client. This entry is the default client type created by the add_install_client script. |
| sysid_config=server1:/export/config | Lists the server name and directory where the sysidcfg file is found. This path is taken from the -p option and argument to the add_install_client script. |
| install_config=server1:/export/config | Lists the server name and directory where the rules and profile files are found. This path is taken from the -c option and argument to the add_install_client script. |
| rootopts=:rsize=32768 | Lists the mount options for the root (/) file system and the NFS read size. |

The /etc/dfs/dfstab File

JumpStart clients require access to directories that servers make available using NFS. Placing an entry for a directory in the /etc/dfs/dfstab file on a server lets the server automatically share the directory when it boots. The add_install_client script creates only one entry in the /etc/dfs/dfstab file on the boot server. This entry shares the location of the boot and installation images. For example:

```
share -F nfs -o ro,anon=0 /export/install
```

The ro and anon=0 options for the share directory in this example let JumpStart clients mount the directory as read-only and retain their root user privileges for the mount.

You must share any other directory that JumpStart clients require with the server that provides it. Generally, you must manually edit the /etc/dfs/dfstab file to create entries for these directories. For example, if a separate server provides JumpStart configuration information, the /etc/dfs/dfstab file on that server must contain an entry for it:

```
share -o ro /export/config
```

Before a JumpStart client can boot and obtain all of the NFS resources it requires, every directory listed as an argument to the add_install_client script must be shared by the server on which it resides.

Exercise 1: Configuring a JumpStart Server to Support One SPARC JumpStart Client

In this exercise, you configure a JumpStart server to support one SPARC JumpStart client.

Preparation

This exercise requires that you work with a lab partner. You must decide which lab system is to be configured as the JumpStart server and which lab system is to be configured as the JumpStart client.

After you decide which lab system has the JumpStart server role, you must attach and mount a Solaris 10 operating system DVD ISO image on that system. The ISO images are located in the /opt/ses/dvd directory. Choose an ISO image appropriate for your lab system architecture. For example:

```
# uname -a
SunOS host41 5.10 Generic_141444-09 sun4u sparc SUNW,UltraAX-i2
# ls /opt/ses/dvd
sol-10-u8-ga-sparc-dvd.iso sol-10-u8-ga-x86-dvd.iso
# lofiadm -a /opt/ses/dvd/sol-10-u8-ga-sparc-dvd.iso /dev/lofi/1
# mount -F hsfs -o ro /dev/lofi/1 /mnt
# ls /mnt
boot/ License/ Copyright platform/ installer* Solaris_10/
JDS-THIRDPARTYLICENSEREADME
```

To unmount and detach the ISO image, perform these steps:

```
# umount /mnt
# lofiadm -d /dev/lofi/1
```

This procedure assumes that the SPARC JumpStart client will use ARP/RARP services to obtain its initial IP address and host name information.

Task Summary

In this exercise, you perform the following tasks:

- Spool the SPARC Solaris boot and installation images from an ISO image file to disk.

Exercise 1: Configuring a JumpStart Server to Support One SPARC JumpStart Client

- Configure the /etc/ethers, /etc/inet/hosts, and /etc/netmasks files on the JumpStart server.
- Configure the rules, and profile files, and create a finish script in the /export/config directory.
- Configure the sysidcfg file in the /export/config directory.
- Configure NFS on the JumpStart server to share the /export/config directory.
- Run the add_install_client script to register one SPARC JumpStart client with the JumpStart server
- Boot and install the SPARC JumpStart client.

Tasks

Creating A Directory For the Install Image

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on the JumpStart server system.
2. Change directory to the /mnt/Solaris_10/Tools directory.

```
# cd /mnt/Solaris_10/Tools
```

3. Create a directory named /export/install/S10_sparc.

Each Solaris image that you spool to disk on the JumpStart server will require its own parent directory. Using the S10_sparc directory below /export/install for a SPARC Solaris image allows you to spool other Solaris images, for example, an x86/x64 image, to a different subdirectory below /export/install.

```
# mkdir /export/install/S10_sparc
```

4. Run the setup_install_server command to spool the boot and install images from the Solaris 10 Software ISO image to the /export/install/S10_sparc directory.

Note – This process takes between 1 and 2 hours typically, depending on the speed of the DVD-ROM drive. It only should take minutes if you have used the ISO image file.



```
# ./setup_install_server /export/install/S10_sparc
```

Verifying target directory...

Exercise 1: Configuring a JumpStart Server to Support One SPARC JumpStart

```
Calculating the required disk space for the Solaris_10 product
Calculating space required for the installation boot image
Copying the CD image to disk...
Copying Install Boot Image hierarchy...
Copying /boot netboot hierarchy...
Install Server setup complete
#
```

5. On the Sun Secure Global Desktop, click the icon for opening a console session on the JumpStart client system.
6. On your client system determine the system MAC address. This can be done on SPARC based systems at the ok prompt [run level 0] by running the banner command and looking at the ethernet address entry.

ok banner

7. Edit the /etc/ethers file, and add the MAC address entry using the information from the previous step for the JumpStart client, for example:

```
0:3:ba:f2:e5:5    client1
```

8. Edit the /etc/hosts file, and add an entry for the JumpStart client, if one does not already exist. Add the timehost alias to the JumpStart server's entry, for example:

```
192.10.200.1      server1  loghost   timehost
192.10.200.100    client1
```

9. Edit or check the /etc/netmasks file to be certain that it contains the network number and subnet mask for your network, for example:

```
192.10.200.0 255.255.255.0
```

10. Create the /export/config directory.

mkdir /export/config

11. If your JumpStart server is a ISO image, change directory to /ISO/Solaris_10/Misc/jumpstart_sample

cd /ISO/Solaris_10/Misc/jumpstart_sample

12. If your JumpStart server using the installation DVD and is a SPARC-based system, change directory to /cdrom/cdrom0/s0/Solaris_10/Misc/jumpstart_sample

cd /cdrom/cdrom0/s0/Solaris_10/Misc/jumpstart_sample

13. If your JumpStart server using the installation DVD and is an x86/x64-based system, change directory to /cdrom/cdrom0/s0/Solaris_10/Misc/jumpstart_sample.

cd /cdrom/cdrom0/s0/Solaris_10/Misc/jumpstart_sample

Exercise 1: Configuring a JumpStart Server to Support One SPARC JumpStart Client

14. Copy the content of the `jumpstart_sample` directory to the `/export/config` directory. This step places sample configuration files in the `/export/config` directory, which you use to set up the JumpStart server.

```
# cp -r * /export/config
```

15. Change the directory to `/export/config`. Move the `rules` file to `rules.orig`.

```
# cd /export/config
```

```
# mv rules rules.orig
```

16. Create a new file called `rules` that contains the following entry. Enter the name of your JumpStart client instead of `client1`:

```
hostname client1 - host_class finish_script
```

17. Edit the `/export/config/host_class` file so that it specifies an initial install; a standalone system type; explicit partitioning; the Entire Distribution software cluster; and partitions for root (/), swap, and /usr. Use partition sizes and device names appropriate for the JumpStart client system; for example:

```
install_type      initial_install
system_type       standalone
partitioning      explicit
cluster           SUNWCall
filesys           c0t0d0s0 10000 /
filesys           c0t0d0s1 512 swap
filesys           c0t0d0s6 free /usr
```

18. In the `/export/config` directory, create a file called `finish_script` that contains the following lines.

```
#!/bin/sh
/usr/bin/touch /a/noautoshutdown
```

These commands configure the JumpStart client to avoid using the autoshutdown power-saving feature.

19. Change the permissions on `finish_script` to 755.

```
# chmod 755 finish_script
```

20. 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 after the `check` program completes successfully.

```
# ./check
```

21. In the `/export/config` directory, create a file called `sysidcfg` that contains the following lines. The string `pVKN72yW0kCMS` is a 13-character encrypted string for the password `cangetin`. You could replace this string with a different encrypted password string by

Exercise 1: Configuring a JumpStart Server to Support One SPARC JumpStart

copying one from your own /etc/shadow file. Use the netmask appropriate to your network, and specify the correct Ethernet interface, timezone, and locale for your system.

```
network_interface=bge0 { primary protocol_ipv6=no
                           netmask=255.255.255.0
                           default_route=none }

name_service=none
timezone=US/Mountain
system_locale=C
timeserver=localhost
security_policy=none
root_password=pVKN72yW0kCMS
nfs4_domain=dynamic
```

22. Edit the /etc/dfs/dfstab file to add an entry for the /export/config directory as follows:

```
share -o ro /export/config
```

23. Run the svcs command to see if the NFS server service is online.

```
# svcs -a |grep nfs
STATE          STIME      FMRI
disabled       14:56:34  svc:/network/nfs/mapid:default
disabled       14:56:34  svc:/network/nfs/cbd:default
disabled       14:56:36  svc:/network/nfs/server:default
online         14:56:56  svc:/network/nfs/status:default
online         14:56:57  svc:/network/nfs/nlockmgr:default
online         14:57:13  svc:/network/nfs/client:default
online         14:57:13  svc:/network/nfs/rquota:ticlts
online         14:57:13  svc:/network/nfs/rquota:udp
```

24. If the NFS server service is disabled, enable it using the svcadm command.

```
# svcadm enable network/nfs/server:default
```

25. Check that the NFS server service is now online.

```
# svcs -a |grep nfs
STATE          STIME      FMRI
disabled       14:56:34  svc:/network/nfs/cbd:default
online         14:57:13  svc:/network/nfs/client:default
online         16:01:13  svc:/network/nfs/status:default
online         16:01:13  svc:/network/nfs/nlockmgr:default
online         16:01:14  svc:/network/nfs/mapid:default
online         16:01:14  svc:/network/nfs/rquota:ticlts
online         16:01:15  svc:/network/nfs/server:default
```

Exercise 1: Configuring a JumpStart Server to Support One SPARC JumpStart Client

online 16:01:15 svc:/network/nfs/rquota:udp

26. With NFS server service running, run the shareall command, then check if the /export/config directory is now shared.

```
# shareall  
# share  
- /export/config ro ""  
#
```

27. Change directory to /export/install/S10_sparc/Solaris_10/Tools.

```
# cd /export/install/S10_sparc/Solaris_10/Tools
```

28. 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 system using a spooled Solaris installation image below the /export/install/S10_sparc directory. Replace *server1* with the name of your JumpStart server, *client1* with the name of your JumpStart client, and *sun4x* with the client architecture, for example sun4u, for the type of client system that you are using.

```
# ./add_install_client -c server1:/export/config -p  
server1:/export/config client1 sun4x  
saving original /etc/dfs/dfstab in /etc/dfs/dfstab.orig  
Adding "share -F nfs -o ro,anon=0 /export/install/S10_sparc" to  
/etc/dfs/dfstab  
making /tftpboot  
enabling tftp in /etc/inetd.conf  
Converting /etc/inetd.conf  
enabling network/tftp/udp6 service  
enabling network/rarp service  
enabling network/rpc/bootparams service  
updating /etc/bootparams  
copying boot file to /tftpboot/inetboot.SUN4U.Solaris_10-1  
#
```

29. In a second terminal window execute the snoop command to watch the JumpStart installation progress. For example, the interface is a "bge" device and the client's ip address is configured to be 192.168.1.201

```
# snoop -d bge0 192.168.1.201
```

30. Use this command to boot the SPARC JumpStart client and initiate the installation.

```
ok boot net - install nowin
```

31. After the JumpStart has successfully completed, login as root using the password cangetin and verify that the OS is the release you installed and that the disk partitioning is as you configured:

```
# cat /etc/release
```

```
Solaris 10 10/09 s10s_u8wos_08a SPARC
Copyright 2009 Sun Microsystems, Inc. All Rights Reserved.
Use is subject to license terms.
Assembled 16 September 2009
```

```
# df -h
```

| Filesystem | size | used | avail | capacity | Mounted on |
|--------------------------|-------------|-------------|-------------|------------|-------------------|
| /dev/dsk/c0t0d0s0 | 6.7G | 910M | 5.8G | 14% | / |
| /devices | 0K | 0K | 0K | 0% | /devices |
| ctfs | 0K | 0K | 0K | 0% | /system/contract |
| proc | 0K | 0K | 0K | 0% | /proc |
| mnttab | 0K | 0K | 0K | 0% | /etc/mnttab |
| swap | 722M | 1.5M | 721M | 1% | /etc/svc/volatile |
| objfs | 0K | 0K | 0K | 0% | /system/object |
| sharefs | 0K | 0K | 0K | 0% | /etc/dfs/sharetab |
| /dev/dsk/c0t0d0s6 | 6.8G | 3.2G | 3.6G | 47% | /usr |
| fd | 0K | 0K | 0K | 0% | /dev/fd |
| swap | 721M | 312K | 721M | 1% | /tmp |
| swap | 721M | 48K | 721M | 1% | /var/run |
| # | | | | | |

32. Let your instructor know you have completed this exercise.

Booting and Installing x86/x64 Systems Over the Network With PXE

This section provides an overview of the Preboot Execution Environment (PXE).

What is PXE?

PXE network boot is a “direct” network boot. No boot media is required on the client system. With PXE, you can install an x86/x64-based client over the network by using DHCP.

PXE network boot is available only for devices that implement the Intel Preboot Execution Environment specification. To determine if your system supports PXE network boot, see your hardware manufacturer’s documentation.

Guidelines for Booting With PXE

To boot over the network by using PXE, you need the following systems.

- An install server
- A DHCP server
- An x86 client that supports PXE

When you are preparing to use PXE to install a client over the network, consider the following issues.

- Set up only one DHCP server on the subnet that includes the client system that you want to install. The PXE network boot does not work properly over subnets that include multiple DHCP servers.
- Some early versions of PXE firmware have a variety of shortcomings. If you experience difficulty with a particular PXE adapter, obtain firmware upgrade information from the adapter manufacturer’s web site. Refer to the `elx1(7D)` and `iprb(7D)` man pages for more information.

Establishing DHCP Services for JumpStart Clients

The Dynamic Host Configuration Protocol (DHCP) enables host systems in a TCP/IP network to be configured automatically for the network as they boot. DHCP uses a client and server mechanism. Servers store and manage configuration information for clients, and provide that information on a client's request. The information includes the client's IP address and information about network services available to the client.

You can use DHCP in conjunction with JumpStart to install the Solaris OS on client systems on your network. All SPARC-based systems that are supported by the Solaris OS and x86-based systems that meet the hardware requirements for running the Solaris OS can use this feature.

SPARC-based clients may make use of RARP or DHCP to supply the identity information they require to boot and begin the system identification and installation process, but x86/x64 clients that use the Pre-boot Execution Environment (PXE) only use DHCP for their configuration.

For this reason, you must configure a DHCP server to support boot and identification operations of x86/x64-based JumpStart clients. The same boot server may provide ARP/RARP services for SPARC clients and DHCP services for x86/x64 clients, or, both SPARC and x86/x64 clients could use DHCP.

This section focuses on configuring DHCP services to support x86/x64 JumpStart clients. More information about configuring DHCP to support both x86/x64 and SPARC JumpStart clients is found in the Solaris 10 05/09 Installation Guide: Network-Based Installations, available through this URL:

<http://docs.sun.com/app/docs/doc/819-6395?q=819-6395>

Configuring DHCP Services on a JumpStart Server

Configuring DHCP services to support JumpStart clients involves two main tasks, given that you have not yet established DHCP services for other purposes.

First, you must plan how you want to configure your DHCP server, and configure the server to provide the basic DHCP services you require.

Second, you must add DHCP macros to your configuration, and define macro options, to support the JumpStart clients that you register with the JumpStart server. The macros and options you add relate to the files created in the /tftpboot directory by the `add_install_client -d` script.

Planning a DHCP Configuration to Support JumpStart Installations

DHCP configuration tasks involve making decisions about the following configuration options:



Note – The exercise associated with this section of the course provides an example procedure that shows how to configure basic DHCP services.

- Selecting a server for DHCP

The host you select must meet specific requirements, including being accessible to all the networks that have clients that plan to use DHCP, either directly on the network or through a BOOTP relay agent.

In the example procedure to set up DHCP to support JumpStart clients, the JumpStart server will provide the required DHCP services.

- Choosing the DHCP Data Store

You can choose to store the DHCP data in text files, binary files, or the NIS+ directory service. In the example procedure, you choose to store DHCP data in text files.

- Setting a Lease Policy

A lease specifies the amount of time the DHCP server permits a DHCP client to use a particular IP address. During the initial server configuration, you must specify a site-wide lease policy.

In the example procedure, you use the default lease policy of one day. JumpStart clients use the IP address and host name information provided by the DHCP server to populate their /etc/inet/hosts files, and so continue to use the identity initially provided by DHCP, without continuing to use the DHCP service after the JumpStart installation completes.

- Selecting a router address or router discovery

When you configure a DHCP server, you must provide DHCP clients with router addresses in one of two ways. One way is to provide specific IP addresses for routers. However, the preferred method is to specify that clients should find routers with the router discovery protocol.

In the example procedure, you elect to use the router discovery protocol.

- Specifying which addresses that the server should manage

You must determine how many IP addresses you want the DHCP server to manage, and what those addresses are. In the example procedure, you specify that the DHCP server will supply only one IP address, which the single JumpStart client will use.

- Deciding if the server should automatically generate host names for clients.

The DHCP management tools can generate a client name to associate with each IP address. When using DHCP to provide identification information to JumpStart clients, however, you typically would assign specific host names to those clients. Depending on how you configure your rules file on your JumpStart server, the client's host name may be required to locate a profile used to configure the client.

In the example procedure, you associate a specific client name with the single IP address that the DHCP server will provide. Your rules file in this procedure uses that client name to locate the profile that the client will use.

- Determine what configuration macro to assign to clients.

This task is described in the next section, and the macro you create for a client depends on the output that the `add_install_client -d` script displays.

Tasks associated with planning your DHCP configuration are described in Chapter 13 of the Solaris System Administration Guide: IP Services, available through this URL:

<http://docs.sun.com/app/docs/doc/816-4554/6maoq01r7?a=view>

The `/usr/sadm/admin/bin/dhcpmgr` utility provides a GUI interface that facilitates setting up DHCP services.

Creating DHCP Macros and Options for x86/x64 JumpStart Clients

For an x86/x64 system that uses the Pre-boot Execution Environment (PXE) to boot through the network, you must create an appropriate DHCP macro on the DHCP server that is configured to support that client.

When you use the `add_install_client` script with the `-d` option on the JumpStart server, the script creates files in the `/tftpboot` directory required to support client boot operations. The `add_install_client` script reports DHCP macro configuration information to standard output. You use this macro configuration information to configure a macro on the DHCP server, so the JumpStart client can find its files in the `/tftpboot` directory.

When you register x86/x64 (PXE) clients with a JumpStart server, always use the `add_install_client` script with the `-d` option.

This `add_install_client` example configures files in `/tftpboot` for one x86/x64 client, and reports the DHCP macro name and the boot options you must use to properly configure DHCP services for this client.

```
# ./add_install_client -d -e 0:e0:81:5b:a6:e -s sys-
06:/export/install/S10_x86 -c sys-06:/export/config -p sys-
06:/export/config/client2 i86pc
Adding "share -F nfs -o ro,anon=0 /export/install/S10_x86" to
/etc/dfs/dfstab
copying boot file to /tftpboot/pxegrub.I86PC.Solaris_10-1
```

If not already configured, enable PXE boot by creating a macro named `0100E0815BA60E` with:

```
Boot server IP (BootSrvA) : 10.1.1.6
Boot file      (BootFile) : 0100E0815BA60E
```

```
#
```

In this example, the DHCP macro name `0100E0815BA60E` and two options, `BootSrvA` and `BootFile` are identified by the `add_install_client` script. The `BootSrvA` and `BootFile` options specify the IP address of the boot server, and the file that the PXE client will use to boot through the network.

Both `BootSrvA` and `BootFile` are standard DHCP macro options.

In the `/tftpboot` directory on the JumpStart server, the `add_install_client` script creates this list of files to support this client:

```
# cd /tftpboot

# ls -l
total 268
lrwxrwxrwx 1 root      root          26 Jun 21 12:36 0100E0815BA60E ->
pxegrub.I86PC.Solaris_10-1
dr-xr-xr-x  3 root      root          512 Jun 17 18:10 I86PC.Solaris_10-1
-rw-r--r--  1 root      root          339 Jun 21 17:46 menu.lst.0100E0815BA60E
lrwxrwxrwx  1 root      root          26 Jun 21 12:36 nbp.0100E0815BA60E ->
pxegrub.I86PC.Solaris_10-1
-rwxr-xr-x  1 root      root          119312 Jun 21 12:36 pxegrub.I86PC.Solaris_10-1
-rw-r--r--  1 root      root          177 Jun 21 12:36 rm.0100E0815BA60E
#
```

The string 0100E0815BA60E in these file names matches the MAC address of the x86/x64 client, with the 01 prefix added, and was generated in these names because of the -e 0:e0:81:5b:a6:e option used with the add_install_client script.

The symbolic link named 0100E0815BA60E matches the BootFile option you specify in the DHCP macro called 0100E0815BA60E. This symbolic link points to the pxegrub.I86PC.Solaris_10-1 file, which is the network boot program that this x86/x64 client will use.

The file named menu.lst.0100E0815BA60E contains the GRUB menu information that the client uses to continue the boot process once it loads the pxegrub.I86PC.Solaris_10-1 boot program. As constructed by add_install_client, the menu.lst.0100E0815BA60E file contains this information:

```
# more menu.lst.0100E0815BA60E
default=0
timeout=30
title Solaris_10 Jumpstart
    kernel /I86PC.Solaris_10-1/multiboot kernel/unix -B
install_config=10.1.1.6:/export/config,sysid_c
onfig=10.1.1.6:/export/config/client2,install_media=10.1.1.6:/export/inst
all/S10_x86,install_boot=10.1.1.6
:/export/install/S10_x86/boot
    module /I86PC.Solaris_10-1/x86.miniroot
```

Note that the kernel command in this file lists the locations of the NFS shared resources that the client requires for the JumpStart installation to proceed. These NFS resources were specified as arguments to the add_install_client script.

To complete the configuration of the menu.1st.*0100E0815BA60E* file, you must edit it and add the `- install dhcp` option at the location indicated by the bold text in the example below:

```
# vi menu.1st.0100E0815BA60E
default=0
timeout=30
title Solaris_10 Jumpstart
    kernel /I86PC.Solaris_10-1/multiboot kernel/unix - install dhcp
-B install_config=10.1.1.6:/export/config
,sysid_config=10.1.1.6:/export/config/client2,install_media=10.1.1.6:/exp
ort/install/S10_x86,install_boot=10.1.1.
6:/export/install/S10_x86/boot
        module /I86PC.Solaris_10-1/x86.miniroot
#
```

Using the macro information provided by the `add_install_client` script, you can configure the required DHCP macro using the `dhcpmgr` utility. The example procedure provided in the exercise for this section combines the general DHCP configuration steps with the specific steps required to create an example macro, using the following information from `add_install_client`:

If not already configured, enable PXE boot by creating a macro named `0100E0815BA60E` with:

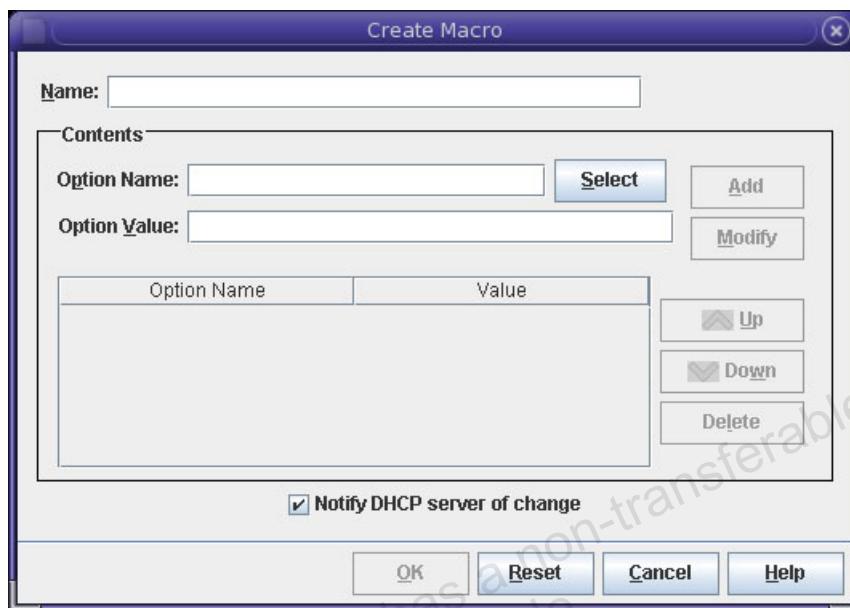
```
Boot server IP (BootSrvA) : 10.1.1.6
Boot file      (BootFile) : 0100E0815BA60E
```

To configure a macro using the `dhcpmgr` utility, perform the following steps. These steps assume an appropriate DHCP configuration already exists on the server used, and that an IP address entry has been assigned to the JumpStart client.

1. Run the `dhcpmgr` utility.

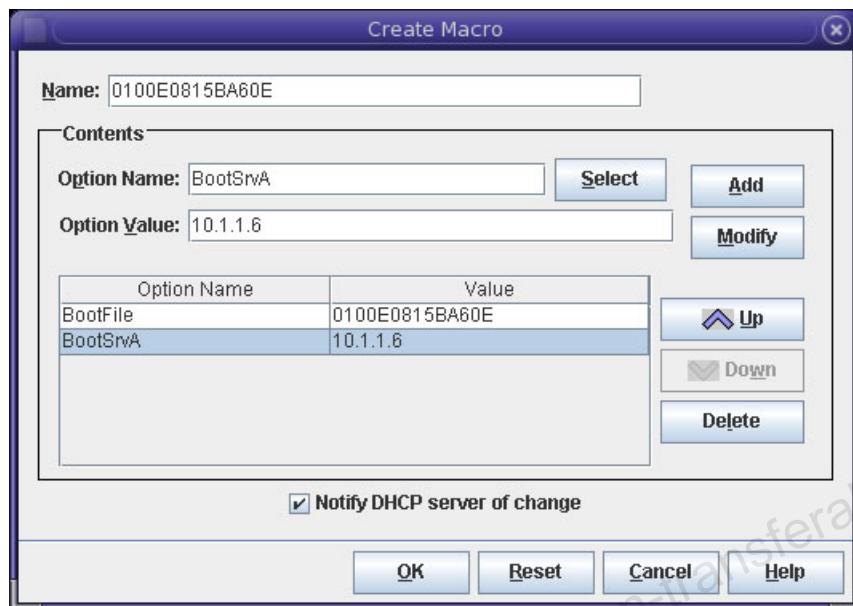
```
# /usr/sadm/admin/bin/dhcpmgr&
```

2. Select the Macros tab on the DHCP Manager main panel. Select Edit > Create to create the macro that your JumpStart client requires. The Create Macro panel displays.



3. Perform the following steps in the Create Macro dialog box:
- In the Name field, enter the macro name displayed by the `add_install_client` command when you ran it.
 - Enter `BootFile` into the Option Name field. In the Option Value field, enter the `BootFile` value displayed by the `add_install_client` script. The `BootFile` value will match the macro name.
 - Click Add.
 - Enter `BootSrvA` into the Option Name field. In the Option Value field, enter the `BootSrvA` value displayed by the `add_install_client` script. This is the IP address of the boot server.
 - Click Add.

f. Click OK when finished.



4. Select File > Exit to exit the dhcpcmgr utility.

You can use the dhtadm -P command to view the DHCP macros that are currently defined, and verify the macro information you supplied to dhcpcmgr. For example:

```
# dhtadm -P
Name          Type           Value
=====
0100E0815BA60E    Macro        :BootFile="0100E0815BA60E":BootSrvA=10.1.1.6:
10.0.0.0        Macro
:Subnet=255.255.255.0:RDiscvyF=1:Broadcst=10.0.0.255:
sys-06          Macro
:Include=Locale:Timeserv=10.1.1.6:LeaseTim=86400:LeaseNeg:
Locale         Macro        :UTCoffst=-25200:
#
```

Booting an x86/x64 JumpStart Client Using DHCP

To boot an X86/x64 JumpStart client from the network, power-on the system and press the F12 key as the system begins its boot process. On an x86/x64 system with PXE boot enabled, this will initiate the boot process from the network.

Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart Client (Demonstration)

In this exercise, you follow along as your instructor demonstrates how to configure a JumpStart server to support one x86/x64 JumpStart client.

Preparation

This exercise provides instructions that are compatible with a JumpStart server that has already been set up to support SPARC JumpStart clients, as described in the previous exercise, or a system that has not yet been configured to provide JumpStart services.

This procedure requires Solaris OS installation DVD media for x86/x64-based systems.

Task Summary

Follow along as your instructor performs these tasks:

- Spool the x86/x64 Solaris boot and installation images from DVD media to disk.
- Configure the rules, and profile files, and create a finish script in the /export/config directory.
- Configure the sysidcfg file in the /export/config/client2 directory.
- Configure NFS on the JumpStart server to share the /export/config directory.
- Run the add_install_client script to register one x86/x64 JumpStart client with the JumpStart server.
- Use the dhcpcmgr utility to set up DHCP services and establish DHCP macros that the x86/x64 client requires.
- Boot and install the x86/x64 JumpStart client.

Tasks

Complete the following steps:

1. On the JumpStart server, log in as `root`. Open a terminal window,
2. Insert the x64/x86 Solaris 10 OS Software DVD into the DVD drive.
3. If your server is a SPARC-based system, change directory to `/cdrom/cdrom0/s0/Solaris_10/Tools`.

`cd /cdrom/cdrom0/s0/Solaris_10/Tools`

4. If your server is an x86/x64-based system, change directory to `/cdrom/cdrom0/Solaris_10/Tools`.

`cd /cdrom/cdrom0/s0/Solaris_10/Tools`

Note – Solaris releases for SPARC prior to Solaris 10 5/09 will use the path: `/cdrom/cdrom0/s0/Solaris_10/Tools`
Newer releases do not have the s0 slice identifier.

5. Create a directory named `/export/install/S10_x86`. Each Solaris image that you spool to disk on the JumpStart server will require its own parent directory. Using the `S10_x86` directory below `/export/install` for an x86/x64 Solaris image allows you to spool other Solaris images, for example, a SPARC image, to a different subdirectory below `/export/install`.

`mkdir /export/install/S10_x86`

6. Run the `setup_install_server` command to spool the boot and install images from the Solaris 10 Software DVD to the `/export/install/S10_x86` directory.

Note – This process takes between 1.5 and 2 hours typically, depending on the speed of the DVD drive.

`./setup_install_server /export/install/S10_x86`

Verifying target directory...

Calculating the required disk space for the Solaris_10 product

Calculating space required for the installation boot image

Copying the CD image to disk...

Copying Install Boot Image hierarchy...

Install Server setup complete

#

Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart

7. Edit or check the /etc/netmasks file to be certain that it contains the network number and subnet mask for your network, for example:

```
192.10.200.0 255.255.255.0
```

8. Create the /export/config directory.

```
# mkdir /export/config
```

9. If your server is a SPARC-based system, change directory to /cdrom/cdrom0/Solaris_10/Misc/jumpstart_sample.

```
# cd /cdrom/cdrom0/s0/Solaris_10/Misc/jumpstart_sample
```

10. Copy the content of the jumpstart_sample directory to the /export/config directory. This step places sample configuration files, used by JumpStart, in the /export/config directory, which you use to set up the JumpStart server.

Skip this step if your server already has the sample JumpStart files in /export/config.

```
# cp -r * /export/config
```

11. Change the directory to /export/config. If you copied the JumpStart sample files to /export/config in the previous step, move the rules file to rules.orig.

```
# cd /export/config
```

```
# mv rules rules.orig
```

12. Add the following entry to an existing rules file, or create a new rules file that contains the following entry. Enter the name of your JumpStart client instead of client2:

```
hostname client2 - host_class2 finish_script
```

13. Create a file called /export/config/host_class2 that specifies an initial install; a standalone system type; one Solaris fdisk partition that uses the entire disk, explicit partitioning; the Entire Distribution software cluster; and partitions for root (/), swap, and /usr. Use partition sizes and device names appropriate for the JumpStart client system.

Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart Client

For example:

```
install_type    initial_install
system_type     standalone
fdisk           all solaris all
partitioning    explicit
cluster          SUNWCall
filesys          rootdisk.s0 10000 /
filesys          rootdisk.s1 512 swap
filesys          rootdisk.s6 free /usr
```



Note – The rootdisk.sx statements in this file identify slices on the system's default boot disk. The fdisk entry specifies that all disks will use one Solaris fdisk partition that uses the entire disk. Slices you specify will exist within the Solaris fdisk partitions on the disks you list.

14. If it does not already exist, in the /export/config directory, create a file called `finish_script` that contains the following lines.

```
#!/bin/sh
/usr/bin/touch /a/noautoshutdown
```

These commands configure the JumpStart client to avoid using the autoshutdown power-saving feature, and prevents the client from displaying questions regarding the NFSv4 domain when it boots. This file provides a simple example of a finish script.

15. Change the permissions on `finish_script` to 755.

```
# chmod 755 finish_script
```

16. Run the /export/config/check program, and correct any problems in the rules or profile files that it reports. Verify that the `rules.ok` file exists after the check program completes successfully.

```
# ./check
```

17. Create a directory named for this JumpStart client to hold its sysidcfg file, for example, /export/config/client2. A JumpStart client must use its own separate sysidcfg file if the sysidcfg file contains information specific to that client.

```
# mkdir /export/config/client2
```

Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart

18. In the `/export/config/client2` directory, create a file called `sysidcfg` that contains the following lines. The string `pVKN72yW0kCMs` is a 13-character encrypted string for the password cangetin. You could replace this string with a different encrypted password string by copying one from your own `/etc/shadow` file. Use the netmask appropriate to your network, and specify the correct Ethernet interface, timezone, and locale for your system.

```
network_interface=nge0 { primary protocol_ipv6=no
                           netmask=255.255.255.0
                           default_route=none }

name_service=none
timezone=US/Mountain
system_locale=C
timeserver=localhost
security_policy=none
root_password=pVKN72yW0kCMs
nfs4_domain=dynamic
```

19. Determine if the `/export/config` directory is currently shared by your JumpStart server.

```
# share
-
      /export/config    ro    ""
(output omitted)
#
```

20. If the `/export/config` directory is not already shared by your JumpStart server, perform the following steps:

- Edit the `/etc/dfs/dfstab` file to add an entry for the `/export/config` directory as follows:

```
share -o ro /export/config
```

- Run the `svcs` command to see if the NFS server service is online.

```
# svcs -a |grep nfs
STATE          STIME      FMRI
disabled       14:56:34  svc:/network/nfs/mapid:default
disabled       14:56:34  svc:/network/nfs/cbd:default
disabled       14:56:36  svc:/network/nfs/server:default
online         14:56:56  svc:/network/nfs/status:default
online         14:56:57  svc:/network/nfs/nlockmgr:default
online         14:57:13  svc:/network/nfs/client:default
online         14:57:13  svc:/network/nfs/rquota:ticlts
online         14:57:13  svc:/network/nfs/rquota:udp
```

Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart Client

- c. If the NFS server service is disabled, enable it using the `svcadm` command.

```
# svcadm enable network/nfs/server:default
```

- d. Check that the NFS server service is now online.

```
# svcs -a |grep nfs
```

| STATE | STIME | FMRI |
|----------|----------|-----------------------------------|
| disabled | 14:56:34 | svc:/network/nfs/cbd:default |
| online | 14:57:13 | svc:/network/nfs/client:default |
| online | 16:01:13 | svc:/network/nfs/status:default |
| online | 16:01:13 | svc:/network/nfs/nlockmgr:default |
| online | 16:01:14 | svc:/network/nfs/mapid:default |
| online | 16:01:14 | svc:/network/nfs/rquota:ticlts |
| online | 16:01:15 | svc:/network/nfs/server:default |
| online | 16:01:15 | svc:/network/nfs/rquota:udp |

- e. With NFS server service running, run the `shareall` command, then check if the `/export/config` directory is now shared.

```
# shareall
```

```
# share
```

```
-          /export/config    ro    ""
```

21. Change directory to
`/export/install/S10_x86/Solaris_10/Tools`.

```
# cd /export/install/S10_x86/Solaris_10/Tools
```

22. 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 a spooled x86/x64 Solaris installation image below the `/export/install/S10_x86` directory. Be certain to:

- Replace `server1` with the name of your JumpStart server.
- Specify the correct MAC address of your JumpStart client, and specify the `i86pc` client architecture.
- For the argument for the `-p` option, specify the correct directory used to hold the client's `sysidcfg` file.

Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart

The -d option specifies that this is a DHCP client.

```
# ./add_install_client -d -e 0:e0:81:5b:a6:e -s sys-  
06:/export/install/S10_x86 -c sys-06:/export/config -p sys-  
06:/export/config/client2 i86pc
```

Adding "share -F nfs -o ro,anon=0 /export/install/S10_x86" to
/etc/dfs/dfstab
copying boot file to /tftpboot/pxegrub.I86PC.Solaris_10-1

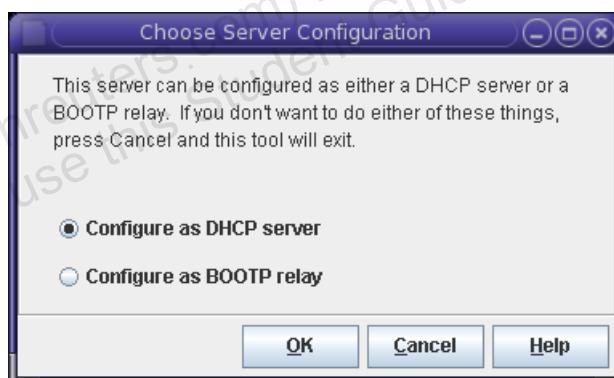
If not already configured, enable PXE boot by creating
a macro named 0100E0815BA60E with:

```
Boot server IP (BootSrvA) : 10.1.1.6  
Boot file      (BootFile) : 0100E0815BA60E  
#
```

23. To begin configuring DHCP services on your JumpStart server, run the dhcpcmgr utility. The Choose Server Configuration panel displays.

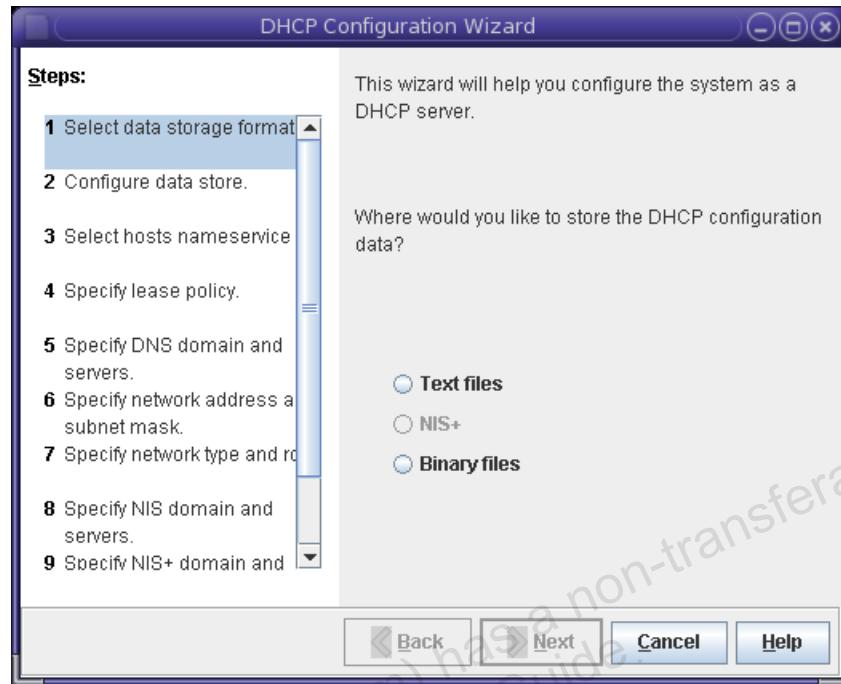
```
# /usr/sadm/admin/bin/dhcpcmgr &
```

24. Select the Configure as DHCP server button, and click OK. The DHCP Configuration Wizard displays.

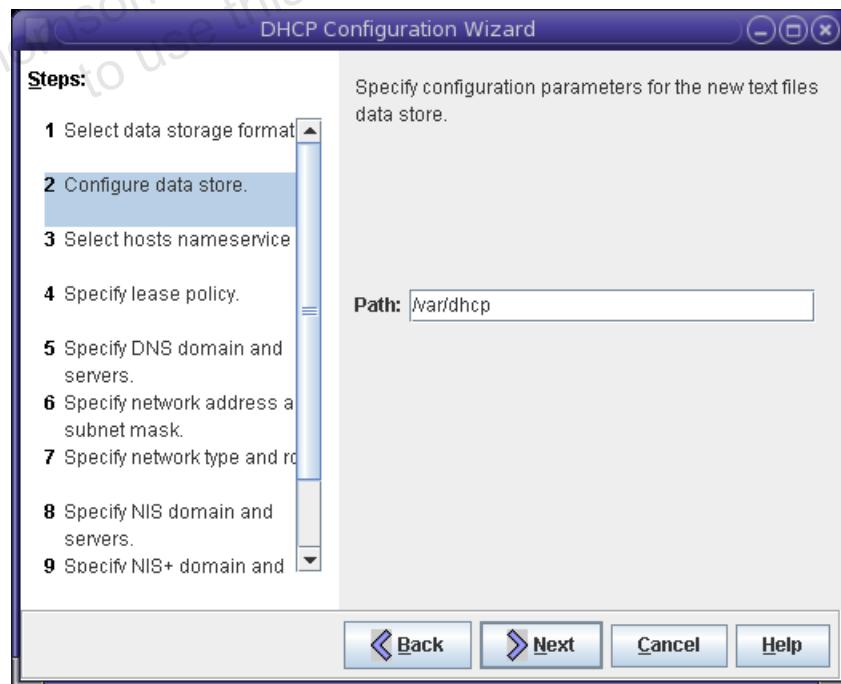


Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart Client

25. Select the Text files item, and click Next. The path dialog box displays.

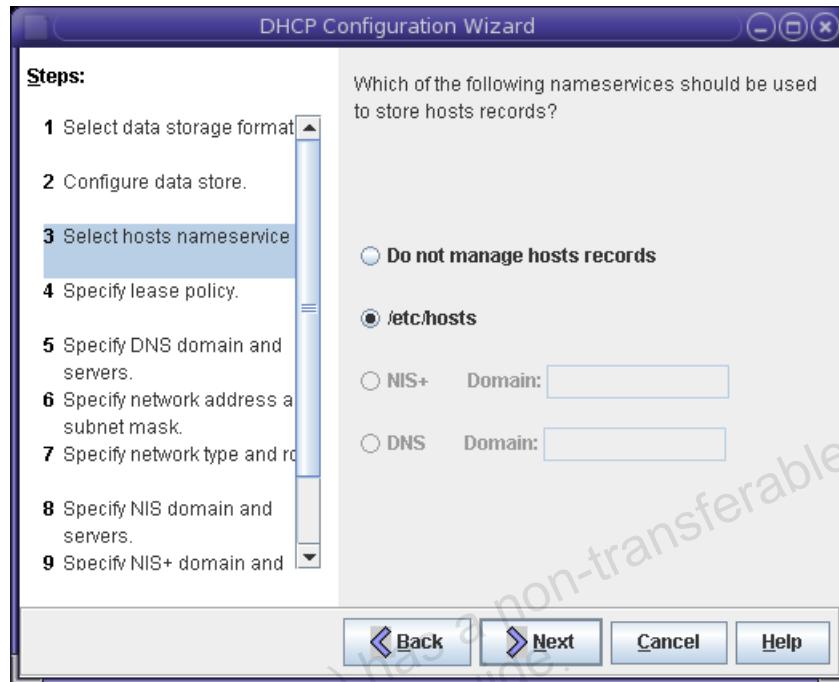


26. Accept the default storage location of /var/dhcp, and click Next. The hosts name service panel displays.

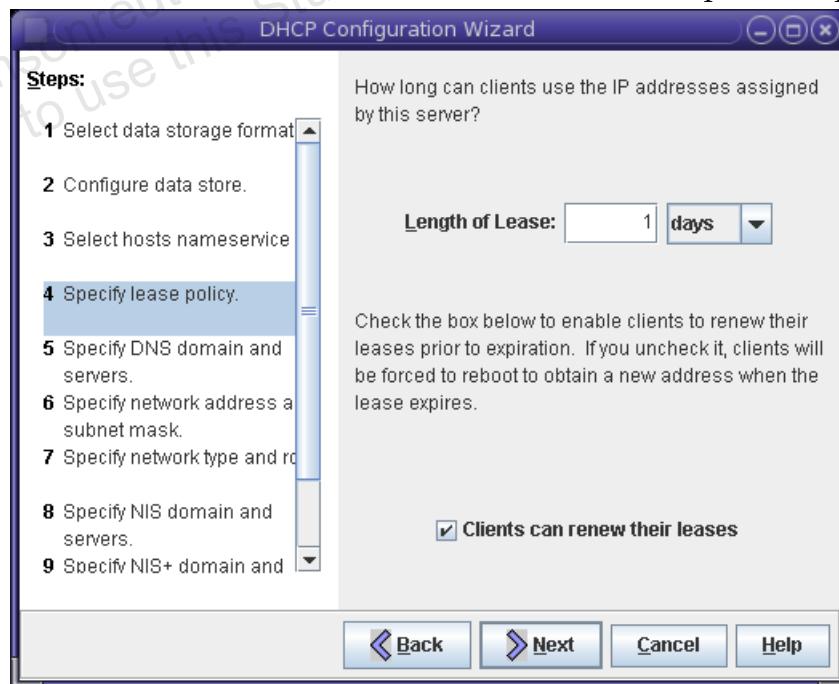


Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart

27. Select the /etc/hosts item, and click Next. The lease information panel displays.

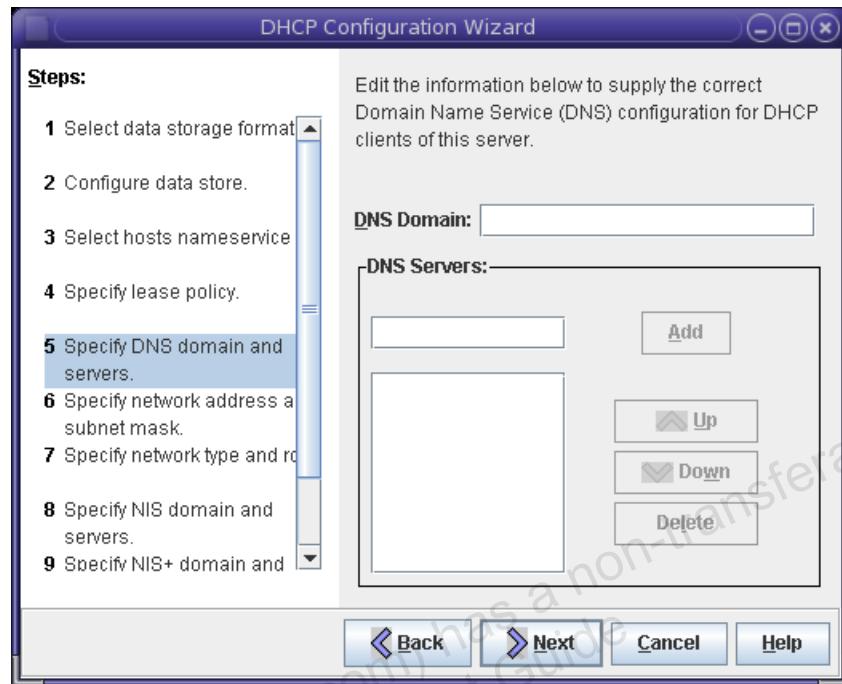


28. Use the default lease length value of 1 day, and allow clients to renew their leases. Click Next. The DNS domain panel displays.

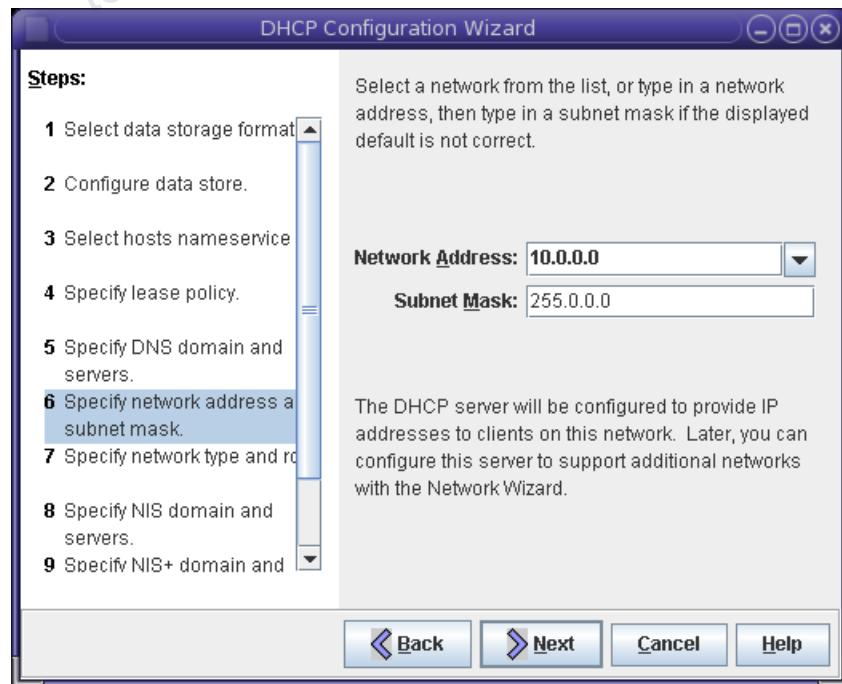


Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart Client

29. In this example, no DNS configuration is required. Click Next. The network address and subnet mask panel displays.

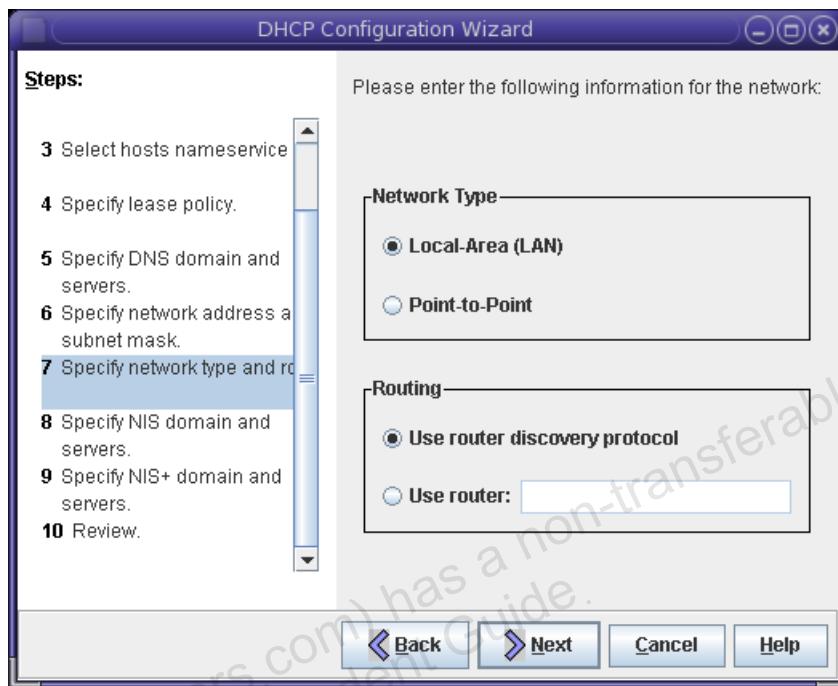


30. Verify that the network address and subnet mask match the values that are correct for your network. Click Next. The network type and routing panel displays

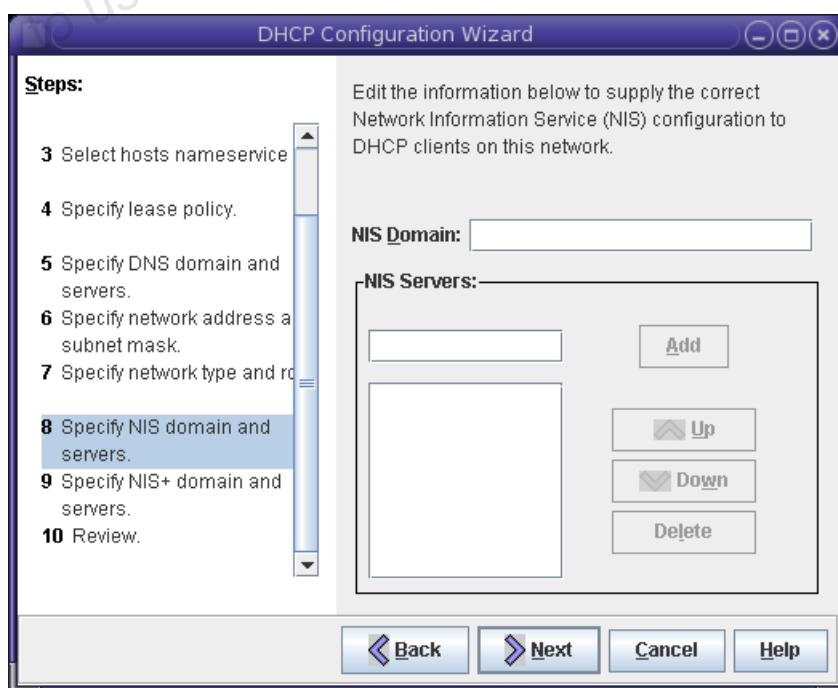


Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart

31. In this example, select the Local-Area network type, and choose to use the router discovery protocol. Click Next. The NIS configuration panel displays.

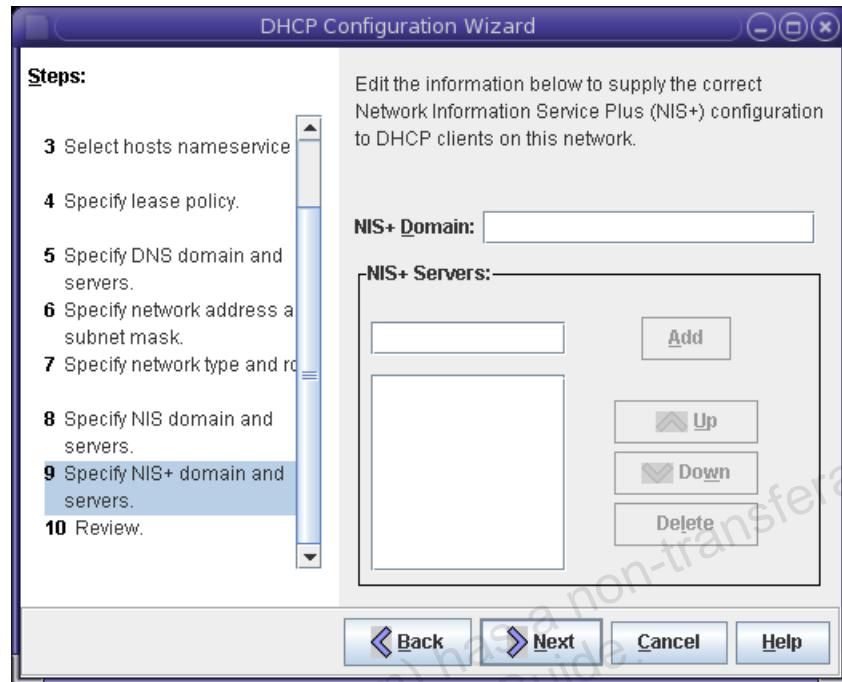


32. For this example, no NIS configuration is required. Click Next. The NIS+ configuration panel displays.

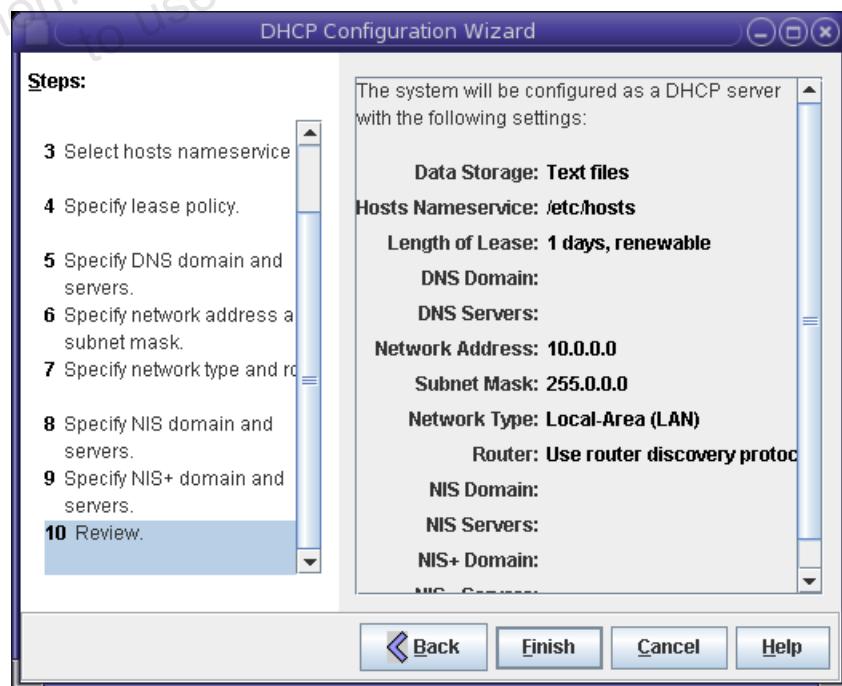


Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart Client

33. For this example, no NIS+ configuration is required. Click Next. The DHCP review panel displays.



34. Verify the settings you have chosen, and use the Back button to go back and correct any setting you want to change. When your configuration is correct, click Finish.

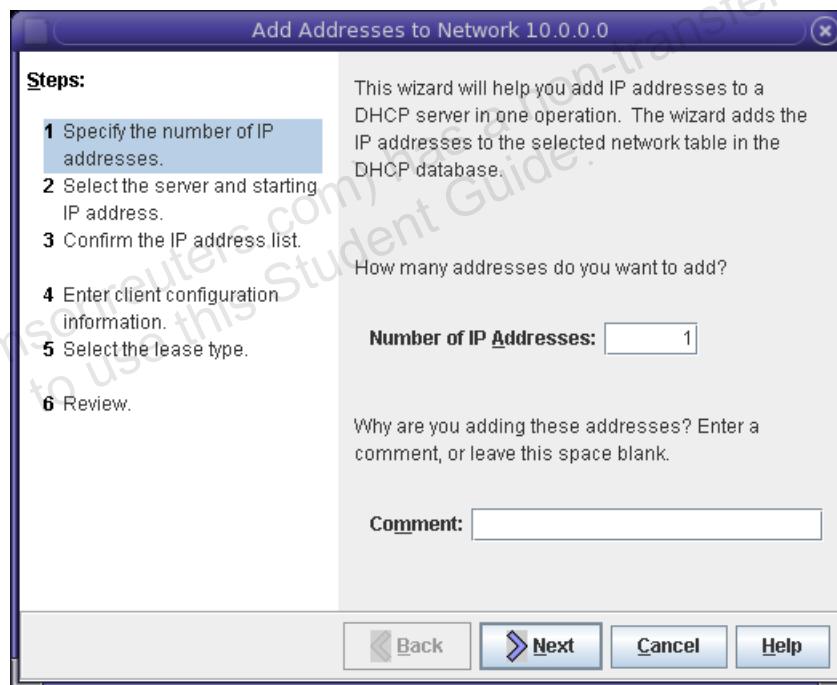


Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart

35. The configuration wizard terminates, and a dialog asks if you want to run the Address Wizard. Click Yes to configure DHCP addresses. The Add Addresses panel displays.

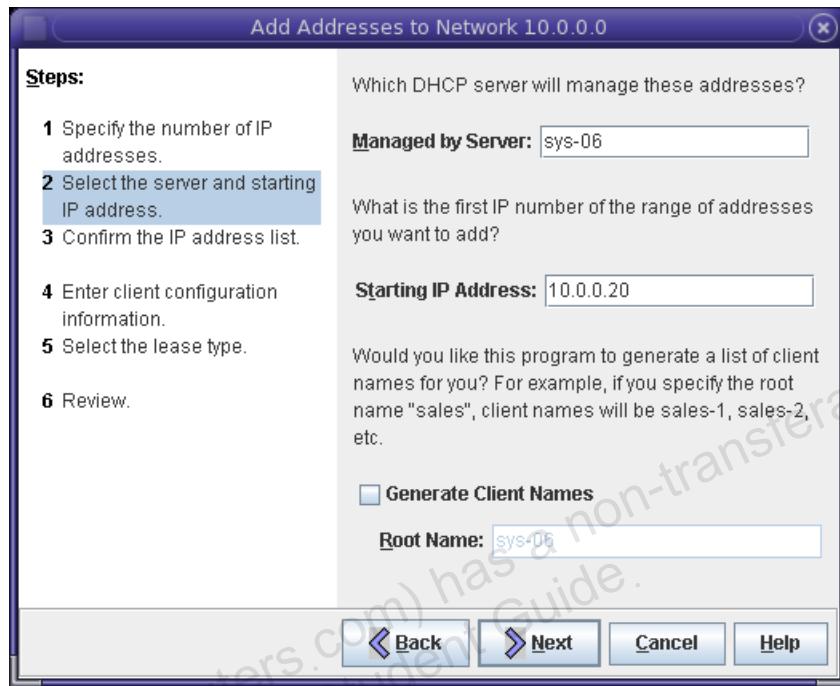


36. Set the number of IP addresses to 1 to support one client. Click Next. The server identification and starting IP address panel displays.

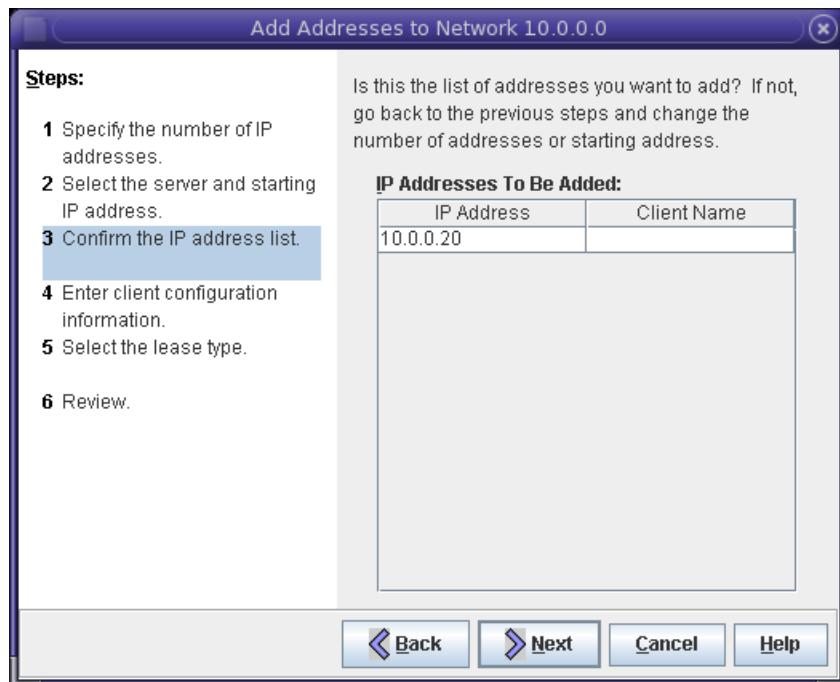


Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart Client

37. Verify that the name of your JumpStart server is listed in the Managed by Server field, and enter the starting IP address you wish to use. Do not choose to generate client names. Click Next. The confirm address list panel displays.

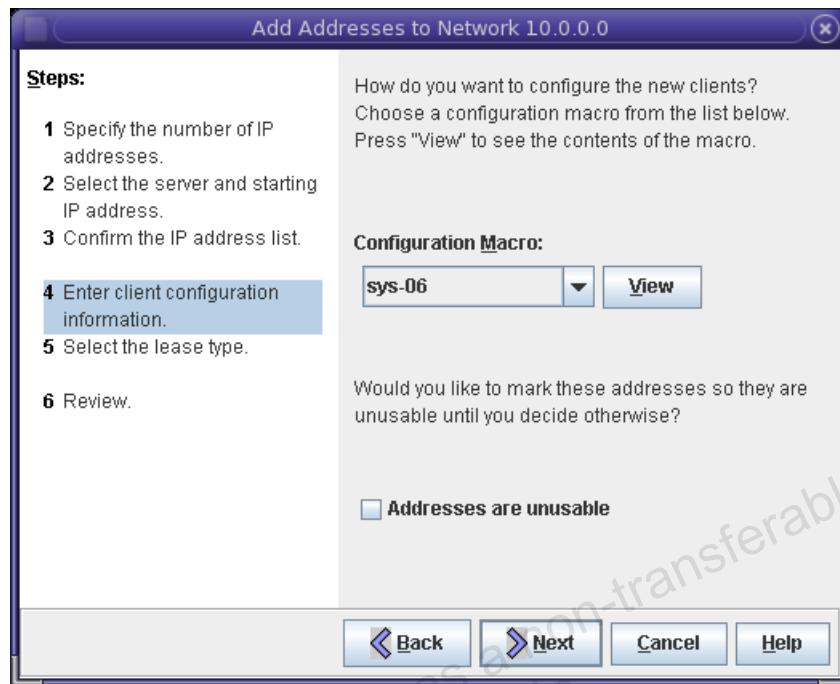


38. Verify that the list of IP addresses is correct. Use the Back button to make changes. When the address list is correct, click Next. The configuration macro panel displays.

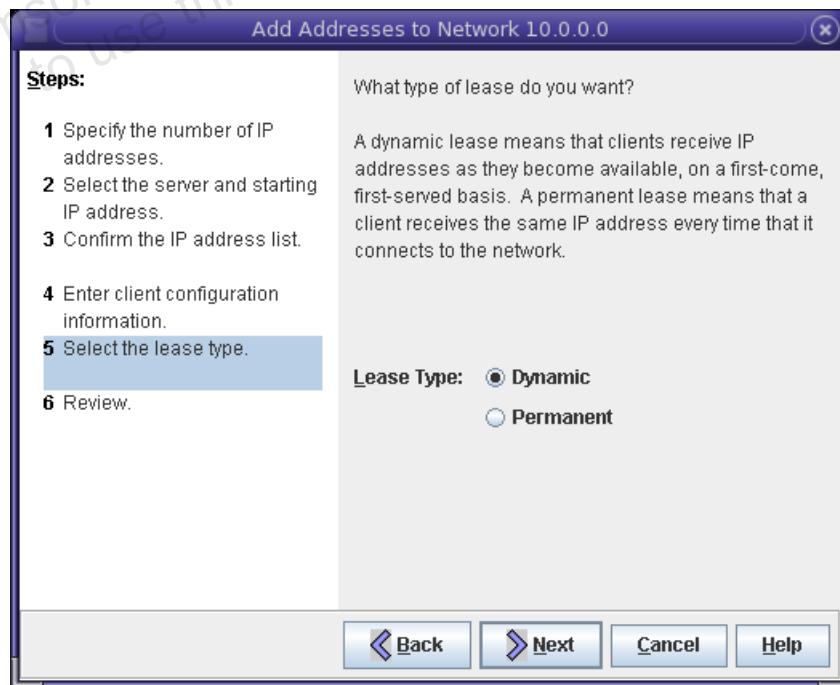


Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart

39. Click Next. The lease type panel displays.

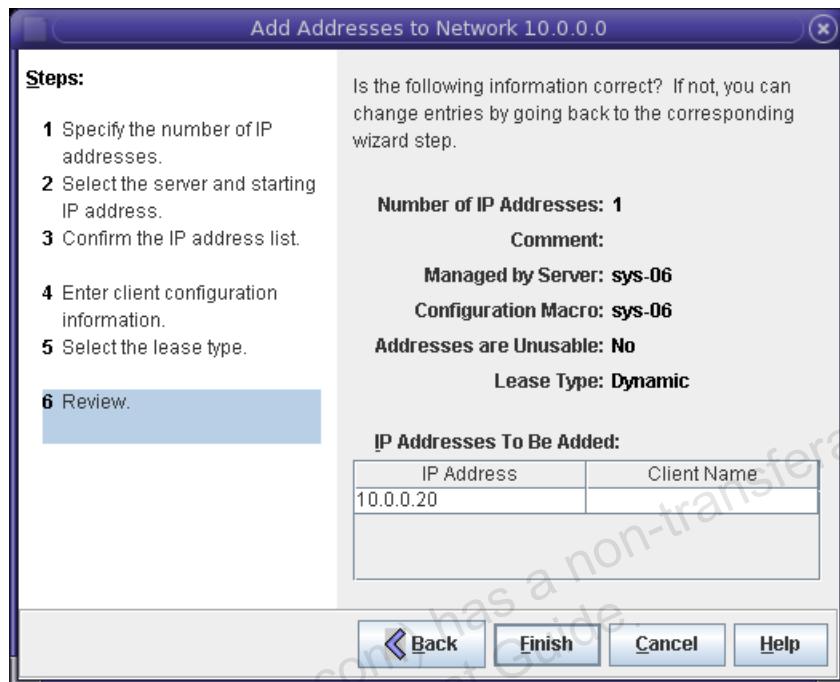


40. Select the Dynamic lease type, and click Next. The Review panel displays.

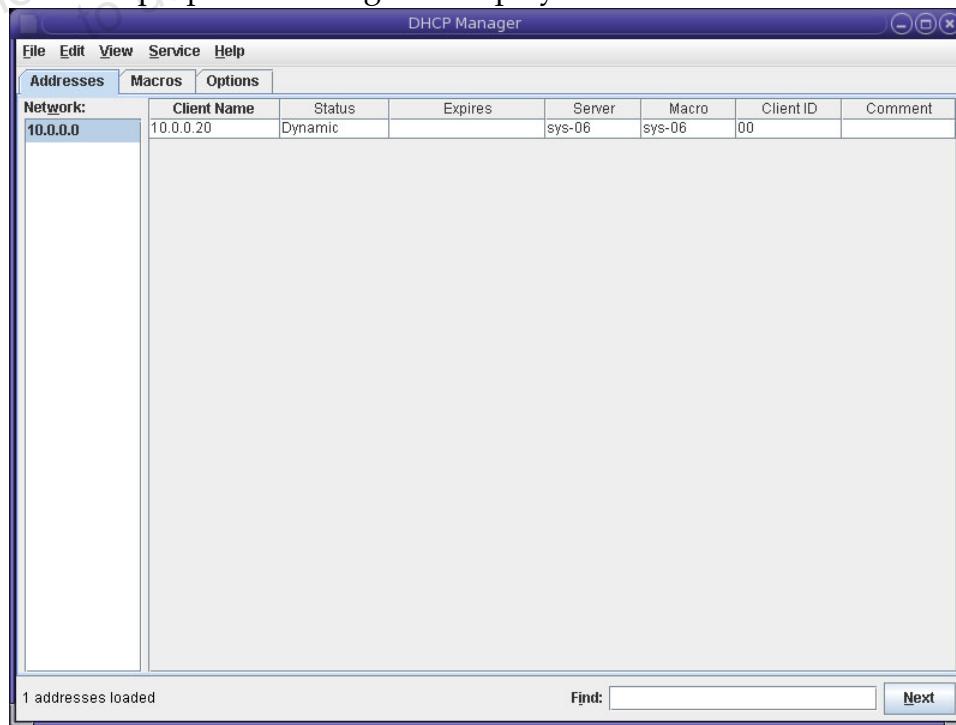


Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart Client

41. Verify that the settings match the configuration you want to use, and click Finish. The DHCP manager main panel now lists an entry for the network address you specified. No client name is listed.

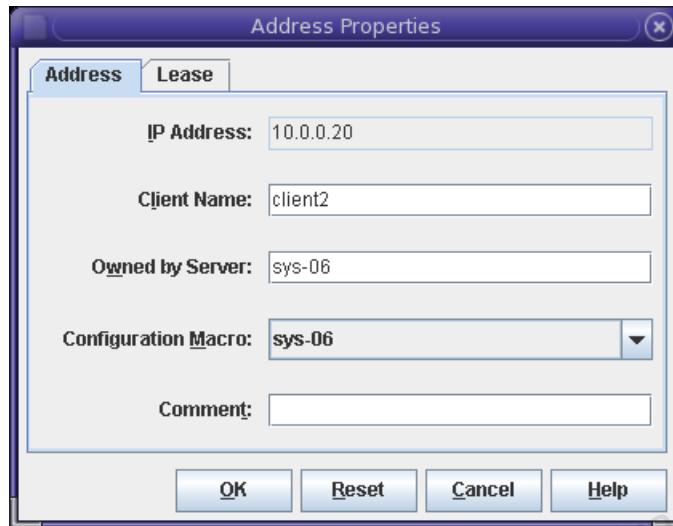


42. Select the new address entry, and then select Edit > Properties. The Address properties dialog box displays.

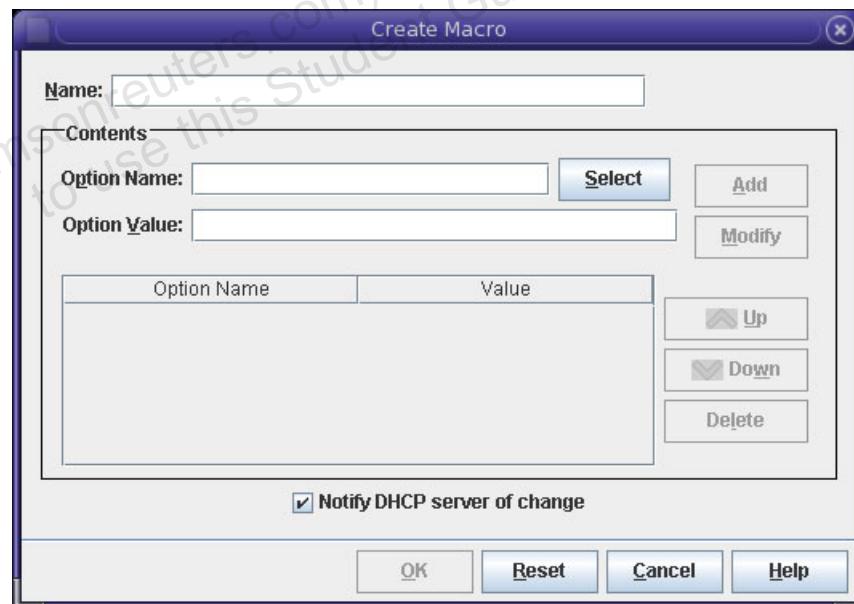


Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart

43. Enter the name of your JumpStart client in the Client Name field (client2 in this example), and click OK.

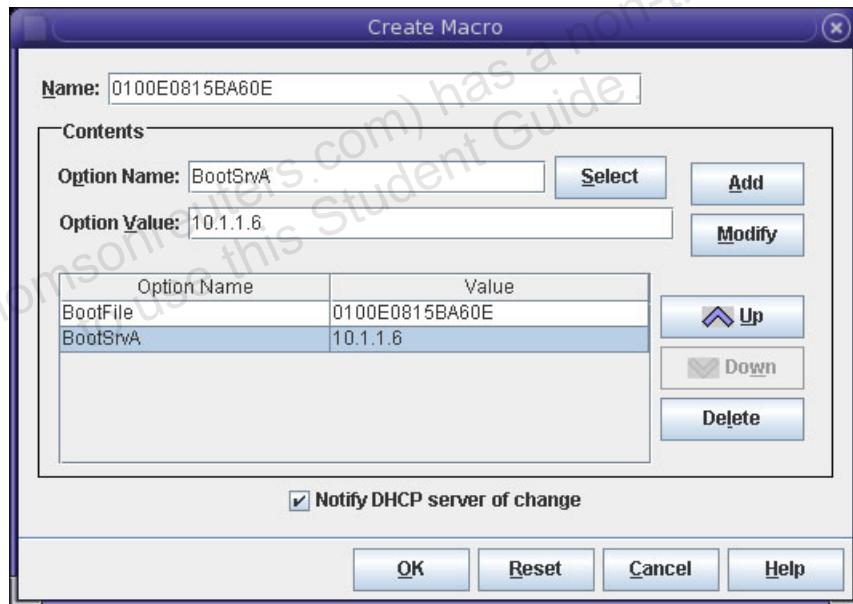


44. Select the Macros tab on the DHCP Manager main panel. Select Edit > Create to create the macro that your JumpStart client requires. The Create Macro panel displays.



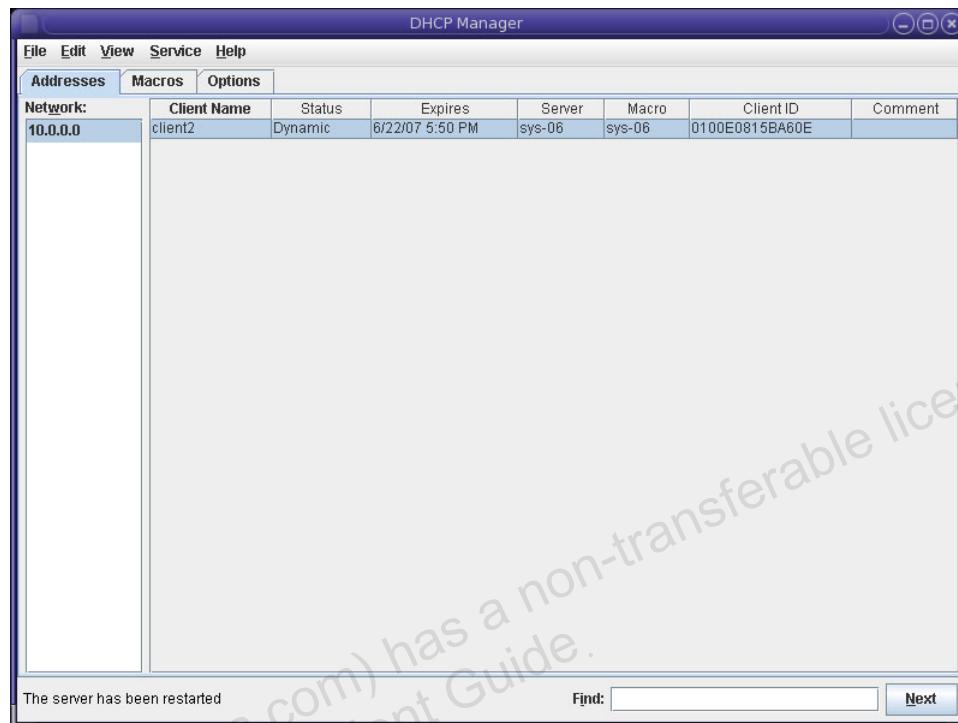
Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart Client

45. Perform the following steps in the Create Macro dialog box:
- In the Name field, enter the macro name displayed by the add_install_client command when you ran it.
 - Enter BootFile into the Option Name field. In the Option Value field, enter the BootFile value displayed by the add_install_client script. The BootFile value will match the macro name.
 - Click Add.
 - Enter BootSrvA into the Option Name field. In the Option Value field, enter the BootSrvA value displayed by the add_install_client script. This is the IP address of the boot server.
 - Click Add.
 - Click OK when finished.



Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart

46. When you are finished with these steps, `dhcpmgr` displays an entry for the address you specified similar to this:



47. Select File > Exit to exit from the `dhcpmgr` utility.
48. Change directory to `/tftpboot`, and list the `menu.1st.xxx` file for your JumpStart Client. The last segment of the `menu.1st.xxx` file name reflects the MAC address of your JumpStart client.

```
# cd /tftpboot
# ls -l *menu.1st*
-rw-r--r-- 1 root      root          324 Jun 21 12:36
menu.1st.0100E0815BA60E
#
```

Exercise 2: Configuring a JumpStart Server to Support One x86/x64 JumpStart Client

49. Edit the menu.1st.xxx file for your JumpStart client, and change the line that reads:

```
kernel /I86PC.Solaris_10-1/multiboot kernel/unix -B  
install_config=10.1.1.6:/export/config,sysid_c  
onfig=10.1.1.6:/export/config/client2,install_media=10.1.1.6:/export/inst  
all/S10_x86,install_boot=10.1.1.6  
:/export/install/S10_x86/boot
```

so it reads:

```
kernel /I86PC.Solaris_10-1/multiboot kernel/unix - install dhcp -B  
install_config=10.1.1.6:/export/config,sysid_c  
onfig=10.1.1.6:/export/config/client2,install_media=10.1.1.6:/export/inst  
all/S10_x86,install_boot=10.1.1.6  
:/export/install/S10_x86/boot
```

The text in bold type identifies the text to add to this line.

50. Boot the X86/x64 JumpStart client from the network. To do this, power-on the system and press the F12 key as the system begins its boot process. On an x86/x64 system with PXE boot enabled, this will initiate the boot process from the network.
51. Let your instructor know you have completed this exercise.

Setting Up JumpStart Software Configuration Alternatives

JumpStart supports a range of alternative server and client configurations. Depending on your network configuration, available server resources, and the client configurations that you want, you can:

- Set up all JumpStart services on a single server
- Configure one server per subnet to provide boot services separately from the other JumpStart services
- Configure boot, identification, configuration, and installation services on separate servers
- Configure begin scripts and finish scripts to further customize software installation on JumpStart clients
- Configure a name service to provide identification information

The flexibility in server and client configuration lets you build JumpStart services to meet your specific software installation needs.

Setting Up a Boot-Only Server for SPARC Clients

Network configuration considerations or limits on server resources might require that you create JumpStart boot-only servers. A boot server responds to RARP, TFTP, and BOOTPARAMS requests from SPARC JumpStart clients and provides a boot image using the NFS service.



Note – SPARC JumpStart clients may also use DHCP instead of RARP to obtain identification information and begin the boot process. This section focuses on RARP-based procedures.

In the BOOTPARAMS information that the boot server offers, it identifies identification, configuration, and installation services.

Two main configuration steps are required to create a JumpStart boot server:

- Running the `setup_install_server` script with the `-b` option to spool a boot image from CD-ROM or DVD
- Running the `add_install_client` script with options and arguments that show a list of servers and the identification, configuration, and installation services that they provide

It is also possible to provide boot services from a shared CD-ROM or DVD, but this is not the most common or practical configuration, and can be a security issue.

Subnet Restrictions

SPARC JumpStart clients broadcast RARP requests when they attempt to boot from the network. Broadcast network traffic is normally not forwarded to networks other than the one where the broadcast traffic originated. This situation requires that a JumpStart boot server exist on the same subnet to which JumpStart clients are directly connected.

The initial network requests for boot-related services are the only JumpStart client requests that are limited by these subnet restrictions. Identification services can be provided by a sysidcfg file made available to the client by using NFS or by binding the JumpStart client to a name service in use. Configuration and installation services are also made available using the NFS service. The NFS service and name services generally allow for network traffic to route among subnets, but the services that depend on them can be provided by servers on different subnets from the one to which the client is directly attached.

Note – An alternative to this restriction would be to use a DHCP installation. More information on this topic can be found at <http://docs.sun.com/app/docs/doc/817-5504>, *Solaris 10 Installation Guide: Network-Based Installations*.

Often, a single server provides all of the JumpStart services. It might be necessary for various reasons to configure servers other than the boot server to respond to identification, configuration, or installation requests from JumpStart clients. In these cases, it is useful to create a boot server on the subnet where JumpStart clients reside.



Figure 12-3 shows a JumpStart network configuration with a separate boot server.

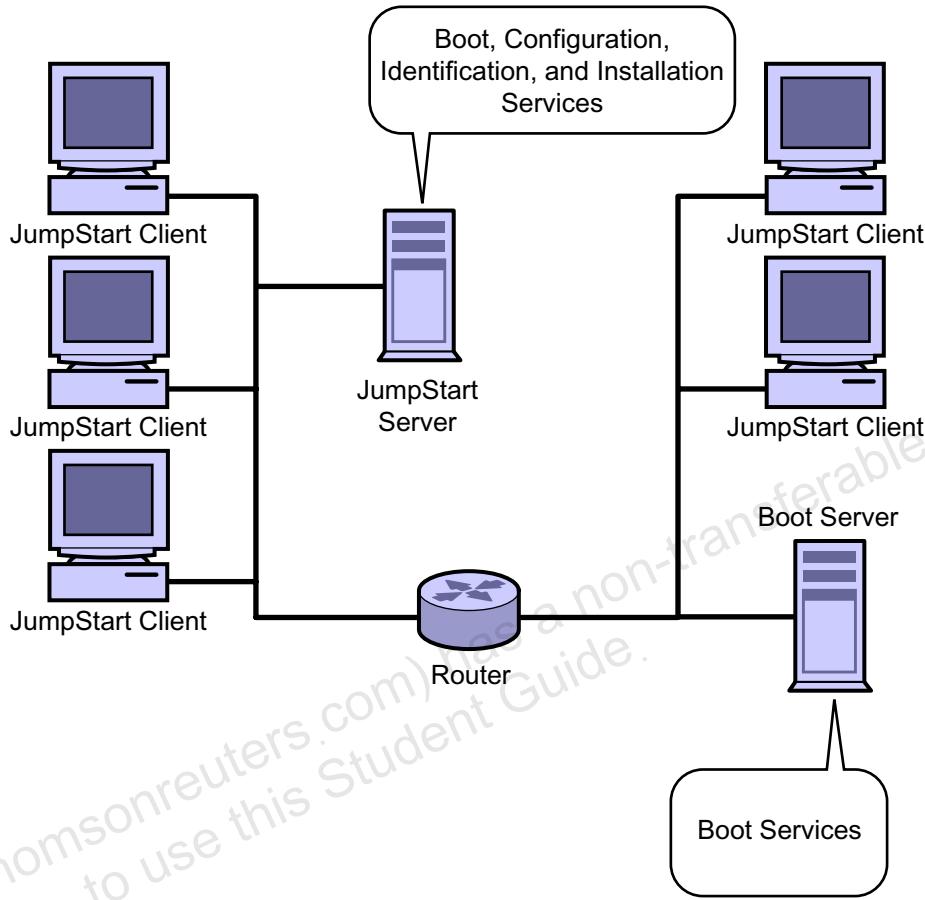


Figure 12-3 The JumpStart Boot Server

Executing the `setup_install_server` Script

To spool only the boot image from a Solaris 10 OS Software 1 CD-ROM or from the DVD, run the `setup_install_server` script with the `-b` option. In the Solaris 10 Operating System, the `setup_install_server` script spools a boot image that occupies about 400 Mbytes of disk space. All JumpStart clients that boot from this server use the same boot image.

Setting Up JumpStart Software Configuration Alternatives

To spool the Solaris 10 Operating System boot image to a local disk, complete the following steps on the system chosen as a boot server:

1. Create an empty directory with at least 400 Mbytes of space available to hold the Solaris Operating System boot image. The `/export/install` directory is usually used for this purpose.

```
# mkdir /export/install
```

2. Insert the Solaris 10 OS Software 1 CD-ROM in the CD-ROM drive, or the Solaris 10 OS DVD in the DVD drive. Allow the `vold` command to automatically mount the media.
3. Change the directory to the location of the `setup_install_server` script.

```
# cd /cdrom/cdrom0/s0/Solaris_10/Tools
```

Note – Solaris releases for SPARC prior to Solaris 10 5/09 will use the path: `/cdrom/cdrom0/s0/Solaris_10/Tools`
Newer releases do not have the s0 slice identifier.

4. Run the `setup_install_server` script with the `-b` option to copy the Solaris 10 Operating System boot image to the local disk. This process can take up to 30 minutes.

```
# ./setup_install_server -b /export/install
```

5. When `setup_install_server` finishes, change directory to root (`/`), and eject the CD-ROM or DVD.

```
# cd /  
# eject cdrom
```

Executing the `add_install_client` Script

The `add_install_client` script configures the boot server to offer the network boot services that JumpStart clients require. When you configure a boot-only server, you must specify options to the `add_install_client` script to indicate which servers and which directories provide identification, configuration, and installation services.

You must run the `add_install_client` script once for each JumpStart client.

Before you run the `add_install_client` script, update the `/etc/inet/hosts` and `/etc/ethers` information for the JumpStart client.

If a name service is not in use, edit the /etc/inet/hosts and /etc/ethers files on the boot server, and add an entry to each file for the JumpStart client. For example, an entry for client1 in the /etc/inet/hosts file could appear as follows:

```
192.10.10.4    client1
```

An entry for client1 in the /etc/ethers file could appear as follows:

```
8:0:20:1c:88:5b client1
```

If a name service is in use, you must edit the /etc/inet/hosts and /etc/ethers files on the appropriate name service server, and run the commands required to update the name service maps or tables.

The /etc/inet/hosts file on the boot server must also contain an entry for each server you specify when you run the add_install_client script.

The add_install_client script automatically makes the changes required for the boot server to support RARP, TFTP, BOOTPARAMS, and NFS requests from the client. The add_install_client script automatically causes the boot server to share the /export/install directory, if that is where the boot image is spooled. Sharing the /export/install directory lets the JumpStart client mount the boot image during the network boot process.

The following procedure assumes that the Solaris 10 OS boot image has been spooled below the /export/install directory on the boot server. It also assumes that the JumpStart Server has the sysidcfg file, rules.ok file, and class file located in the /export/config directory.

To run the add_install_client script on a boot server, complete the following steps:

1. Update the /etc/inet/hosts information to add an entry for the JumpStart client.
2. Update the /etc/ethers information to add an entry for the JumpStart client.
3. Change the directory to the location of the add_install_client script on the server.

```
# cd /export/install/Solaris_10/Tools
```

Setting Up JumpStart Software Configuration Alternatives

Run the `add_install_client` script, and specify server and client information as follows:

```
# ./add_install_client -c 192.168.1.201:/export/config -p  
192.168.1.200:/export/config -s 192.168.1.200:/export/install  
clientA sun4u  
saving original /etc/dfs/dfstab in /etc/dfs/dfstab.orig  
Adding "share -F nfs -o ro,anon=0 /export/install/Solaris_10/Tools/Boot"  
to /etc/dfs/dfstab  
making /tftpboot  
enabling tftp in /etc/inetd.conf  
starting rarpd  
starting bootparamd  
starting nfssd's  
starting nfs mountd  
updating /etc/bootparams  
copying inetboot to /tftpboot  
#
```

When you complete this procedure, and meet conditions on the other servers, you can initiate the installation process on a JumpStart client.

Setting Up Identification Service Alternatives

JumpStart clients can obtain the identification information that they require from different sources, including the `/etc/inet/hosts` file on a boot server, the `sysidcfg` file, or a name service, such as LDAP, NIS+, or NIS. Identification information provided in a `sysidcfg` file takes precedence over information provided by other sources.

Configuring `/etc/inet/hosts` and `/etc/ethers` Files

If a name service is not in use, a JumpStart client obtains its IP address and host name from the `/etc/inet/hosts` file found on the boot server.

If a name service is in use, the maps or tables that contain `/etc/inet/hosts` and `/etc/ethers` information must include entries for the JumpStart client.

Configuring the sysidcfg File

JumpStart clients use information in the sysidcfg file to answer identification questions. Information in this file replaces identification information available to the client from other sources. If the JumpStart client cannot obtain a response for an identification question, the client interrupts the automatic identification process and asks for the information.

The Solaris OS JumpStart clients require a sysidcfg file to answer identification questions that cannot be provided by default from a name service, including entries with information regarding:

- Default router (if not using router discovery)
- IPv6
- Kerberos configuration
- Naming service

The sysidcfg file allows you to specify nearly all of the identification information that a JumpStart client requires. The sysidcfg file can contain:

- Identification information that all JumpStart clients can use
- Information that is client-specific

If you supply client-specific information in the sysidcfg file, you must create a separate sysidcfg file for each client. You must name the file sysidcfg on each system. Therefore, if you specify client-specific information in the sysidcfg file, you must place each unique sysidcfg file in a separate directory.

Locating the sysidcfg File

Typically, you would create a generic sysidcfg file in the /export/config directory on a JumpStart server. The sysidcfg files that contain client-specific information must exist in separate directories. For example, the /export/config/client1/sysidcfg directory.

SPARC JumpStart clients learn of the location of the sysidcfg file from BOOTPARAMS information that they obtain from the boot server. When you run the add_install_client script on the boot server, use the -p option, and specify the server and path where the sysidcfg file is stored. The following command indicates that the sysidcfg file that client1 will use is found on the server, server1, in the /export/config directory.

```
# ./add_install_client -c server1:/export/config -p  
server1:/export/config client1 sun4u
```

The server, server1, must share the /export/config directory by using the NFS service before the client can mount it.

Constructing the sysidcfg File

The sysidcfg file lets you specify many different identification items. Entries in the sysidcfg file must conform to the following rules:

- Independent keywords can be listed in any order.
- Keywords are not case sensitive.
- Keyword values can be optionally enclosed in single (') or double (") quotation marks.
- Dependent keywords must be enclosed in curly braces ({}) to tie them to their associated independent keyword.
- For all keywords except the network_interface keyword, only the first instance of a keyword is valid. If a keyword is specified more than once, only the first keyword specified is used.

Examples of the sysidcfg File

The following is an example of the sysidcfg file configuring a single network interface:

```
network_interface=bge0 { primary protocol_ipv6=no  
                         netmask=255.255.255.0  
                         default_route=192.10.10.1}  
  
security_policy=none  
name_service=none  
timezone=US/Mountain  
system_locale=en_US  
timeserver=192.10.10.1  
root_password=Hx23475vABDDM  
nfs4_domain=dynamic
```



Note – The encrypted root_password entry in this example represents the password cangetin.

Setting Up JumpStart Software Configuration Alternatives

The following example shows a sysidcfg file which is used to configure multiple network interfaces. The capability to configure multiple network interfaces in the sysidcfg file was introduced in Solaris 9 (9/04).

```
network_interface=bge0 { primary hostname=sys01
                           ip_address=192.168.2.10
                           protocol_ipv6=no
                           netmask=255.255.255.0
                           default_route=192.168.2.1}

network_interface=qfe0 { hostname=sys01
                           ip_address=192.168.2.101
                           protocol_ipv6=no netmask=255.255.255.0
                           default_route=192.168.2.1}

network_interface=qfe1 { hostname=sys02
                           ip_address=192.168.2.111
                           protocol_ipv6=no netmask=255.255.255.0
                           default_route=192.168.2.1}

network_interface=qfe2 { dhcp protocol_ipv6=no }

network_interface=qfe3 { ip_address=192.168.2.121
                           protocol_ipv6=no netmask=255.255.255.0
                           default_route=192.10.10.1}

security_policy=none
name_service=none
timezone=US/Mountain
system_locale=en_US
timeserver=192.10.10.1
root_password=Hx23475vABDDM
nfs4_domain=dynamic
```

Setting Up Configuration Service Alternatives

You can customize how JumpStart clients load and configure the Solaris OS. Entries in the `rules.ok` and profile files establish the basic Solaris OS configuration that a JumpStart client uses. Begin and finish scripts further customize the software installation process.

Examples of `rules` File Entries

The following is an example of the `rules` file entries.

```
#  
# The first five rules listed here demonstrate specifics:  
#  
hostname  client1      -      host_class      set_root_pw  
hostname  client2      -      class_basic_user -  
network   192.43.34.0  && ! model 'SUNW,T5240' - class_net3 -  
model    'SUNW,T5240'  -      class_t5240     complete_t5240  
memsize  64-106       &&     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`. 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 any machine on network `192.43.34` that is not a Sun SPARC Enterprise T5240 Server system architecture. The class file is `class_net3`. This rule does not specify a begin or finish script.
- The fourth rule matches a machine that is a Sun SPARC Enterprise T5240 Server system architecture. The class file is `class_t5240`. There is a finish script called `complete_t5240`.
- The fifth rule matches a machine using SPARC architecture and with a memory size between 64 and 106 Mbytes. The class file is `class_prog_user`.
- The sixth rule matches any machine. The class file is `class_generic`. This rule does not specify a begin or finish script.

Begin Scripts

Begin scripts are Bourne scripts that JumpStart clients run *before* installing the Solaris OS. Begin scripts allow you to perform a variety of tasks on the JumpStart client. Typically, you would use a begin script to back up data from the client before proceeding with the Solaris OS installation.

The following example begin script causes the JumpStart client to copy its existing /etc/passwd and /etc/shadow files to a directory on an NFS server:

```
#!/bin/sh

HOSTNAME=`/bin/uname -n`
mount 192.10.10.100:/backup /mnt

if [ ! -d /mnt/${HOSTNAME} ]; then
    mkdir /mnt/${HOSTNAME}
fi

if [ -d /mnt/${HOSTNAME} ]; then
    mount /dev/dsk/c0t0d0s0 /a
    cp /a/etc/passwd /a/etc/shadow /mnt/${HOSTNAME}
    umount /a
fi

umount /mnt
```

This example script works only if the following conditions exist:

- The server using the IP address 192.10.10.100 shares the /backup directory in read-write mode and with the anon=0 option set
- The JumpStart client has a previously installed root (/) file system available as /dev/dsk/c0t0d0s0

This example script shows that a begin script can mount disk resources from other systems, mount resources from the client itself, and copy files between those mounted directories. File systems that exist on the client are available using their standard logical device names. NFS provides access to shared directories on the network. The mount points /a and /mnt are available in the root (/) file system when the JumpStart client mounts from the boot server.

For a client to use a begin script, the script must be associated with a rule that the client selects from the rules file. For example, the rule:

```
hostname client1 begin1 config1 -
```

would cause a JumpStart client called client1 to use the begin script called begin1.

Profile (Class) File

A profile file is a text file that determines how the Solaris Operating System installation proceeds on a JumpStart client. Profile files are sometimes called *class* files. Rules listed in the rules file allow classes of clients to select an appropriate profile file. Although you usually associate a different profile with every rule, you can use the same profile for multiple rules.

The following example shows that for a client to use a profile file, the profile must be associated with the rule the client selects from the rules file:

```
hostname client1 - config1 -
```

The rule.ok file would cause a JumpStart client called client1 to use the profile file called config1.

An entry in a profile file consists of one keyword and its associated parameters. Each keyword controls one element of the Solaris Operating System software installation. Each profile consists of multiple entries. Profile file names must match the names used in the rules file.

Keywords and Arguments

Table 12-5 lists the keywords and parameters used in a profile file to specify how the Solaris OS installation proceeds on the JumpStart client.

Table 12-5 Keywords and Arguments for Profile Files

| Keywords | Arguments |
|-----------------------------|---|
| install_type | initial_install upgrade flash_install flash_upgrade |
| system_type | standalone server |
| partitioning | default existing explicit |
| cluster <i>cluster_name</i> | add delete |
| package <i>package_name</i> | add delete |
| timeout | <i>minutes</i> |
| retry | <i>number</i> |
| usedisk | <i>disk_name</i> |
| dontuse | <i>disk_name</i> |
| locale | <i>locale_name</i> |
| num_clients | <i>number</i> |
| client_swap | <i>size</i> |
| client_arch | <i>kernel_architecture</i> |
| filesys | <i>device size file_system optional_parameters</i> |
| metadb | <i>slice [size in blocks] [number]</i> |
| patch | <i>patch_id_list patch_file patch_location</i> |
| archive_location | <i>retrieval_type location</i> |
| pool | <i>ZFS_root_pool_name</i> |
| bootenv | <i>installbe</i> |
| bename | <i>Live_Upgrade_boot_environment_name</i> |

The cluster keyword requires a parameter that lists name of the software group you want to install. Table 12-6 defines the software group names according to the common names used for them during the interactive installation routine.

Table 12-6 Possible Entries for the cluster Keyword

| Interactive Installation Name | software group Name |
|--------------------------------------|---------------------|
| Reduced Network | SUNWCrnet |
| Core | SUNWCreq |
| End User | SUNWCuser |
| Developer | SUNWCprog |
| Entire Distribution | SUNWCall |
| Entire Distribution Plus OEM Support | SUNWCXall |

Note – See the *Solaris™ 10 System Release and Installation Collection* for a description of the clusters and packages available on the Solaris 10 Software Distribution CD-ROMs.



Examples of Profile Files

Creating an Empty Boot Environment

In the following example, the profile indicates that the custom JumpStart program creates an empty boot environment. An empty boot environment contains no file systems and no copy from the current boot environment occurs. The boot environment can be populated later with a Solaris Flash archive and then activated.

```
# profile keywords          profile values
# -----
install_type                initial_install
system_type                 standalone
partitioning                explicit
filesys                     c0t0d0s0 auto /
filesys                     c0t3d0s1 auto swap
filesys                     any auto usr
cluster                      SUNWCall
bootenv createbe bename second_BE \
filesystem ./:/dev/dsk/c0t1d0s0:ufs \
filesystem -:/dev/dsk/c0t1d0s0:swap \
filesystem /export:shared:ufs
```

Creating RAID-1 Volumes When Installing a Solaris Flash Archive

In the following example, the profile indicates that the custom JumpStart program uses Solaris Volume Manager technology to create RAID-1 volumes (mirrors) for the root (/), swap, /usr and /export/home file systems. A Solaris Flash archive is installed on the boot environment.

```
# profile keywords          profile values
# -----
install_type                flash_install
arhcive_location            nfs
server:/export/home/export/flash.s10.SUNWCall
partitioning                explicit
filesys                     mirror:d10 c0t0d0s0 c0t1d0s0 4096 /
filesys                     mirror c0t0d0s1 2048 swap
filesys                     mirror:d30 c0t0d0s3 c0t1d0s3 4096 /usr
filesys                     mirror:d40 c0t0d0s4 c0t1d0s4 4096 /usr
filesys                     mirror:d50 c0t0d0s5 c0t1d0s5 free /export/home
metadb                      c0t1d0s7 size 8192 count 3
```

Finish Scripts

Finish scripts are Bourne scripts that JumpStart clients run *after* installing the Solaris Operating System but *before* they reboot. Finish scripts allow you to perform a variety of post-installation tasks on the JumpStart client, including:

- Setting the power-management configuration
- Retrieving backed-up data from a server on the network
- Copying selected files from a JumpStart server to the client

The following example finish script causes the JumpStart client to turn off automatic shutdown for power management, retrieve its backed-up /etc/passwd and /etc/shadow files from a directory on an NFS server, and copy a file from the configuration server to the JumpStart client.

```
#!/bin/sh

touch /a/noautoshutdown

HOSTNAME=`/bin/uname -n`

mount 192.10.10.100:/backup /mnt

if [ -d /mnt/${HOSTNAME} ]; then
    echo "Copying passwd and shadow..."
    cp /mnt/${HOSTNAME}/passwd /a/etc/passwd
    cp /mnt/${HOSTNAME}/shadow /a/etc/shadow
fi

umount /mnt

mkdir /a/labfiles

cp ${SI_CONFIG_DIR}/files/setup.tar /a/labfiles
```

This example script works if the following conditions exist:

- The server using the IP address 192.10.10.100 shares the /backup directory.
- The passwd and shadow files exist in the /backup/*client_name* directory on the server that shares it, where *client_name* is the host name of the JumpStart client.

- The configuration server has the file called `setup.tar` in the `files` directory. The `files` directory must exist in the directory that this server shares, and the client uses it as `${SI_CONFIG_DIR}`.

Typically `${SI_CONFIG_DIR}` refers to the `/export/config` directory on the configuration server. `${SI_CONFIG_DIR}` specifically refers to the directory associated with the `install_config` item that the client found in the `/etc/bootparams` file. The `${SI_CONFIG_DIR}` variable is one of several JumpStart software-specific variables that you can use in begin and finish scripts.

Note – For more information on JumpStart software variables available for use in begin and finish scripts, refer to the Solaris 10 Release and Installation Collection. In the Solaris 10 OS and earlier releases back to Solaris 2.5.1, JumpStart clients automatically mount all of their file systems below the `/a` directory, before the finish script runs. The client uses its boot image to construct the directory that it will use on reboot. The directory hierarchy is mounted under the `/a` directory in the boot image. This temporary mount point allows finish scripts to make changes to the client's directory hierarchy by prefixing the absolute path name of the files and directories to be modified, created, or deleted with the `/a`. This directory allows you to write finish scripts that copy files into the client's file systems without mounting them within the script.

The `touch /a/noautoshutdown` command is the only method available to automatically disable the power management feature on the JumpStart client. Without this file in the client's root (`/`) directory, the client asks power management configuration questions when it boots.

For a client to use a finish script, the script must be associated with the rule that the client selects from the `rules.ok` file. For example, consider the rule:

```
hostname client1 begin1 config1 finish1
```

This rule would cause a JumpStart client called `client1` to use the finish script called `finish1`.

Setting Up Installation Service Alternatives

In addition to the standard JumpStart installation configurations, you can create alternatives for installation.

The boot and installation services can be loaded from the following:

- Solaris 10 Installation DVD
- Solaris 10 Installation CD-ROMs
- ISO image mounted using the loopback filesystem
- Flash Archive

Using CD and DVD Sources

You can set up boot and installation services directly from the Solaris 10 OS Software DVD or from the Solaris 10 OS Software 1 CD-ROM. To do this, you must also configure identification and configuration services in the same manner as when you use a spooled Solaris OS image. You can also use the lofiadm command to mount the Solaris 10 ISO image and make it available to the installation service.

Using A DVD Or CD-ROM For The Installation Image

The Solaris 10 DVD contains an installation image that supports installing all software groups through the Entire Distribution with Operating System support.

The installation image found on the Solaris 10 OS Software 1 CD-ROM only supports installing the Core and Minimal Network software groups. The Solaris 10 OS Software 2, 3, and 4 CD-ROMs contain the remainder of the installation image, but there is no support for changing CD-ROMs in the middle of a JumpStart installation procedure.

To set up boot and installation services from DVD or CD-ROM, complete the following steps:

1. Insert the Solaris 10 OS Software DVD in the DVD drive or the Solaris 10 OS Software 1 CD-ROM in the CD-ROM drive. Allow the vold daemon to automatically mount the media.
2. Change the directory to the location of the `add_install_client` script.

```
# cd /cdrom/cdrom0/Solaris_10/Tools
```

3. Run the `add_install_client` script, and specify the server and client information as follows:

```
# ./add_install_client -c server:/config_path -p server:/sysid_path
client_name platform_group
```

- a. For the `server:/config_path` value, enter the name of the server and path where the rules and profile files are located.
- b. For the `server:/sysid_path` value, enter the name of the server and path where the sysidcfg file is located.
- c. For the `client_name` field, enter the name of the JumpStart client.
- d. For the `platform_group` field, enter the correct kernel architecture for the JumpStart client, for example, sun4u.

The `add_install_client` script automatically makes the changes required to support RARP, TFTP, BOOTPARAMS, and NFS requests from the client, but this script only causes the server to share the `/cdrom/sol_10_sparc/s0` directory. Sharing the `/cdrom/sol_10_sparc/s0` directory lets the JumpStart client to mount a root (/) file system during the network boot process and to gain access to the installation image.

You must manually configure the appropriate servers to share the other directories you name in the `add_install_client` command.

Using A Flash Source

You can also use a Flash source as an alternative installation service. The Flash installation feature lets you to create a single reference installation of the Solaris 10 OS on a master system. You can replicate the installation on other systems known as clones.

The Flash installation utilities are available as part of the Solaris OS. Before the Flash archive is created and deployed, you must decide how to integrate the installation process into your specific environment. Some items to consider are:

- Building support for custom hardware and driver configurations at installation time, which eliminates the need to re-create the archive in the future. The recommended installation for the required level of support on the master is Entire Distribution + OEM support.
- Selecting the name conventions for each archive in advance.

- Allocating the contents of each archive or customized multiple archives, including third-party software and package additions or deletions. At least one archive must contain the Solaris 10 OS files.
- Using the Solaris Installation.

There are certain advantages to using a Flash archive for the installation. These include:

- Reduction in installation time.
- Greater portability, as the Flash archive can be used on more than one system architecture.
- The ability to include third-party software in the installation source.

In comparison, the advantages of using the standard JumpStart suninstall method for the installation include:

- Able to be more selective with the installation options based on the architecture and system build of the JumpStart client.
- Layout of storage media can be more greatly controlled.

New Solaris 10 Keywords

Starting with the first release of Solaris 10, new keywords have been added to enhance the JumpStart process. These keywords can greatly reduce the installation time and eliminate in some cases the need for finish scripts.

Creating RAID-1 Volumes Using the Profile File

The `filesystem` and `mirror` keywords can be used in the profile file to create RAID-1 volumes on the client system.

The syntax of the profile `filesystem` keyword is:

```
filesystem [mirror[:name]] slice slice size file_system [mount_options]
```

The following example creates a mirror called `d12` consisting of two components, slice `c0t0d0s0` and `c1t3d0s0`. The size of the mirror is 850 Mbytes and is used as the mount point for the root file system.

```
filesystem mirror:d12 c0t0d0s0 c1t3d0s0 850 /
```

If a name is not provided for the mirror, one is automatically provided.

The `mirror` keyword causes one state database replica to be put on each slice in the mirror automatically.

The `metadb` Keyword

The administrator may choose to create additional metastate databases using the `metadb` keyword.

The `metadb` keyword allows the system administrator to specify the size and number of metastate databases that will reside on a mirrored slice.

The `metadb` keyword syntax is as follows:

```
metadb slice [size size] [count count]
```

An example of the syntax is:

```
metadb c1t0d0s0 size 8192 count 3
```



Note – The above example would not be considered best practice. It would be better to put the 3 metadb's on separate disks.

The `metadb` keyword defaults to 1 metastate database and uses a default size of 8192 blocks. Metastate databases can only be put on slices that will be part of a mirror or the JumpStart will fail. The `mirror` qualifier automatically puts one metastate database on each slice. Use the `metadb` keyword when the system administrator wants more than one metastate database per mirror slice.

Sample Profile File Using Mirroring

The following profile example creates RAID-1 volumes (mirrors) for the root (/), /usr, and /var file systems:

```
install_type    initial_install
cluster        SUNWCXall
filesys        mirror    c0t0d0s0    c1t3d0s0  850      /
filesys        mirror:d10   c0t0d0s3   c1t3d0s3  1000     /var
filesys        c0t0d0s1           512      swap
filesys        c1t3d0s1           512
filesys        c0t0d0s7    count  4
metadb         c1t3d0s7    count  4
filesys        mirror    c0t0d0s6    c1t3d0s6  5000     /usr
filesys        c0t0d0s7           free    /export/home
filesys        c1t3d0s7           free
```

The following list describes this example:

1. The installation type is an initial installation.
2. The Entire Distribution Plus OEM software cluster is to be installed.
3. The root (/) file system is created and mirrored on the slices `c0t0d0s0` `c1t3d0s0` and is 850 Mbytes in size. The resulting RAID volumes are automatically assigned names as none is specified.
4. The /var file system is created and mirrored on the slices `c0t0d0s3` and `c1t3d0s3`. The RAID-1 volume is called `d10`.
5. The swap slice is created on `c0t0d0s1` and is 512 Mbytes in size.
6. Slice `c1t3d0s1` is 512MB in size but is not allocated to any file system.

7. Four state database replicas are created on slice c0t0d0s4 and slice c1t3d0s4.
8. The /usr filesystem is created and mirrored on slices c0t0d0s6 and c1t3d0s6. The name of the RAID-1 volume is automatically assigned.
9. The /export/home file system is created on the remaining free space on disk c0t0d0s7.
10. Slice c1t3d0s7 is created on the remaining free space on c1t3d0 but is not allocated to any file system.

Installing Packages That Are Not Part of the Installation Media

The package keyword previously was only used to add or delete packages from the installation that were part of the installation media. In Solaris 10 the keyword has been enhanced to allow package installations that are not part of the installation media. Previously this was only possible by using a finish script.

Packages to be installed can be obtained from the following sources:

- NFS server
- HTTP server
- Local device
- Local file

The syntax for the entry in the profile varies depending on the location selected, as shown in Table 12-7.

Table 12-7 Package Syntax

| Package Source | Syntax example |
|----------------|---|
| NFS | package SUNWnew add nfs sys01:/var/spool/pkg/Solaris_10 or package SUNWnew add nfs://sys01/var/spool/pkg/Solaris_10 |
| HTTP | package SUNWnew add http://sys01/solaris10 or package SUNWnew add http://sys01/solaris10 proxy sys02:8080 |
| local_device | package SUNWnew add local_device c0t6d0s0 /solaris10/pkg ufs |
| local_file | package SUNWnew add local_file /solaris10/pkg |

Note – HTTP with packages requires a streams format package.



The keyword `timeout` is an optional keyword that allows the system administrator to specify the maximum length of time in minutes that is allowed to pass without receipt of data from the HTTP server. If the `timeout` value is exceeded, the connection closes, reopens, and resumes.

If the value is set to 0, the connection is not reopened. A time-out reconnection causes pkgadd to restart from the beginning and data that was received prior to the timeout is discarded.

The keyword proxy is an optional keyword that allows specification of a proxy host and proxy port. A proxy host retrieves a Solaris package from beyond a firewall. When you use the proxy keyword, you must specify a proxy port. If you do not use the proxy keyword and do not specify a port, port 80 is used.

The following examples show the optional keywords:

```
package SUNWnewpkg http://server1/solaris10 timeout 5
```

or

```
package SUNWnewpkg add http://server1/solaris10 proxy localserv:8080
```

You can install a package from a local device such as a diskette or a CD-ROM. You must specify the full device pathname. If the full pathname is not specified, /dev/dsk is added to the pathname. If you do not specify the filesystem type, ufs is tried first and hsfs is tried second.

The path is relative to the / (root) of the device specified. The following example shows the proper syntax:

```
package SUNWname add local_device device path file_system_type
```

For example, a package installation from the local CD-ROM with an HSFS file system uses the following entry in the JumpStart profile file:

```
package SUNWnewpkg add local_device c0t6d0s0 /solaris10/pkg
```

A package can be installed if it is part of the miniroot. The miniroot is a DVD, CD, or NFS mounted directory from which the system is booted. You can access any file that is part of the miniroot during the JumpStart installation. The following example shows the proper syntax for adding packages from a local file:

```
package SUNWname add local_file path
```

For example:

```
package SUNWnewpkg add local_file /solaris10/pkg
```

If you specify a location for a package in the profile file and do not specify a specific location for those packages following the package listed with the location, the subsequent packages are assumed to be in the same location as the first package. For example:

```
package SUNWnewpkg1 http://server1/var/spool/pkg timeout 5
package SUNWnewpkg2
package SUNWnewpkg3 http://server2/var/spool/pkg
package SUNWnewpkg4
package SUNWnewpkg5
package SUNWnewpkg6 nfs://server3/export
```

The use of keyword all in place of a package name indicates that all packages in the specified location should be added. For example:

```
package all http://server1/var/spool/pkg
```

Adding Patches Using the `patch` Keyword

The `patch` keyword has been introduced to allow patches to be installed during the JumpStart process. Previously patches had to be installed either manually or with a `finish` script. Patches can be obtained from the following sources:

- NFS server
- HTTP server
- Local device
- Local file

Table 12-8 Patch keyword syntax

| Source | Syntax Example |
|--------------|--|
| NFS | <pre>patch list_file nfs://sys01/solaris_10/patches patch 112345-06,122223-01 nfs sys01:/solaris_10/patches</pre> |
| HTTP | <pre>patch 112233-01,223344-04 http://sys01/solaris10/patches patch list_file http://sys01/solaris10/patches</pre> |
| local_device | <pre>patch 112233-01,223344-04 local_device c0t6d0s0 /solaris10/Patches patch list_file local_device c0t6d0s0 /solaris10/Patches</pre> |
| local_file | <pre>patch 112233-01,223344-04 local_file /solaris10/Patches patch list_file local_file /solaris10/Patches</pre> |

The `patch` keyword supports a list of comma separated patches or a file containing a list of patches as arguments.

Note – A patch stored on an HTTP server must be in JAR format.

The keywords `timeout` and `retry` also work with the `patch` keyword.



The keyword `retry` is an optional keyword whose argument `n` represents the maximum number of times the install process will attempt to mount the directory.

An example of each style entry in the profile file is:

```
patch patch-list-file nfs://server1/solaris10/patches retry 5
```

or

```
patch 112467-01,112765-02 nfs server1:/solaris10/patches
```

JumpStart Installation For A ZFS Root (/) File System

Beginning with the Solaris 10 10/08 release, you can use JumpStart to install a ZFS root file system. The profile must contain the `pool` keyword. The `pool` keyword installs a new root pool, and a new boot environment is created by default.

You can provide the name of the boot environment and you can create a separate `/var` dataset with existing `bootenv` and `installbe` keywords and the new `bename` and `dataset` options.

Some keywords that are allowed in a UFS-specific profile are not allowed in a ZFS specific profile, such as those specifying the creation of UFS mount points.

Note – For overall ZFS planning information, see Chapter 6, “ZFS Root File System Installation (Planning),” in *Solaris 10 05/09 Installation Guide: Planning for Installation and Upgrade*.

The following is an example entry in a JumpStart profile file to install a ZFS root file system:

```
pool pool1 auto auto auto mirror c0t0d0s0 c0t1d0s0
```

The keyword `pool` defines the name of the root pool. `auto` specifies the size of the disks automatically. The size is determined by the size of the specified disks.

The second `auto` configures the swap area to be automatically sized. The default size is 1/2 the size of physical memory, but no less than 512 Mbytes and no greater than 2 Gbytes. You can set the size outside this range by using the `size` option.

The third `auto` configures the dump device to be automatically sized. The mirrored configuration of disks has the `mirror` keyword and disk slices specified as `c0t0d0s0` and `c0t1d0s0`.

The following is another example specifying the size of the disk slice to be 80 Gbytes and the swap area and dump volumes to be 2 Gbytes in size:

```
pool newpool 80g 2g 2g mirror any any
```

The any options in the mirrored configuration finds any two available devices that are large enough to create a 80-Gbyte pool. If two such devices are not available, the install fails.



Note – See the JumpStart Keywords for a ZFS Root (/) File System (Reference) section in the Solaris 10 05/09 Installation Guide: Custom JumpStart and Advanced Installations.

ZFS Root Flash Install

New with the Solaris 10 10/09 release, a JumpStart profile can now be used to identify a Flash archive of a ZFS root pool. Prior to this release Flash archive are not supported when installing a ZFS root pool.

A Flash archive install with a ZFS root file system is achieved with a two-step process:

1. Generate a Flash archive that is used to install and boot a system with a ZFS root file system.
2. Perform a JumpStart installation of a system by using a ZFS Flash archive.

The ZFS Flash archive (flar) contains the entire pool hierarchy, except for the swap and dump volumes and any excluded datasets. The swap and dump volumes are created when the Flash archive is installed.

For example, to create a flar of the ZFS root pool

```
# flarcreate -n zfs10u8BE zfs10u8flar
Full Flash
Checking integrity...
Integrity OK.
Running precreation scripts...
Precreation scripts done.
Determining the size of the archive...
The archive will be approximately 4.94GB
Creating the archive...
Archive creation complete.
Running postcreation scripts...
Postcreation scripts done.
```

Running pre-exit scripts...
Pre-exit scripts done.

After the flar is created of a ZFS root pool on a Solaris 10 10/09 system, you can create a jumpstart profile on the installation server to be used for installing any system. For example, the following profile is used to install the S10U8 flar just created:

```
install_type flash_install
archive_location nfs system:/export/jump/zfs10u8flar
partitioning explicit
pool rpool auto auto mirror c0t1d0s0 c0t0d0s0
```

Note – For a complete installation method, refer to the new section in the *ZFS Administration Guide: Installing a ZFS Root File System (Flash Archive Installation)*.

Some dependencies and requirements for a ZFS root flash install include:

- The interactive option of a flar installation is not supported with the ZFS root file system, nor can a ZFS flar be used to install a ZFS Bootable Environment (BE) with Live Upgrade.
- ZFS flar installs are only supported on systems with the same architecture. For example, a flar created on a sun4u system will not work on a sun4v system.
- Only a full installation of a ZFS flar is supported. Using a differential, or a hybrid UFS/ZFS flar of a ZFS root file system is not supported.
- The ZFS flar can only be used to install a ZFS root file system, not UFS.
- Although the entire root pool, minus any explicitly excluded datasets, is archived and installed, only the ZFS BE that is booted when the archive is created is usable after the Flash archive is installed. However, pools that are archived with the `fclar` or `fclarcreate` command's `-R rootdir` option can be used to archive a root pool other than the one that is currently booted.
- The `fclarcreate` and `fclar` command options to include and exclude individual files are not supported in a ZFS flar, only excluding entire datasets from a ZFS flar is allowed.
- The `fclar info` command is not supported for a ZFS Flash archive and returns the following error:

```
# fclar info -l zfs10u8flar
ERROR: archive content listing not supported for zfs
archives.
```



Creating Additional Boot Environments Using Live Upgrade And Jumpstart

You can also add additional Live Upgrade boot environments during the JumpStart installation process using the bootenv and bename profile keywords. An example JumpStart profile file entry is as follows:

```
bootenv installbe bename solaris10_8
```

bootenv installbe changes the characteristics of the default boot environment that is created during the installation.

The bename names the new boot environment as solaris10_8.

Note – Live Upgrade is covered in detail in Module 14 of this course.



Jumpstart And Zones

Starting with the Solaris 10 1/06 release, when non-global zones are installed, you can use the custom JumpStart program to upgrade. For an automated JumpStart installation, you can upgrade or patch with any keyword that applies to an upgrade or patching. In releases prior to Solaris 10 08/07, only a limited number of keywords could be used. The time to upgrade or patch might be extensive, depending on the number of non-global zones that are installed.

Exercise 3: (Optional) Creating a ZFS Mirrored Root Pool

In this exercise, you will modify your JumpStart client profile file to include the new JumpStart keyword pool to create a ZFS pool.

- Determine which drive on the client system is the boot drive using the `df -h` command:

```
# df -h /
Filesystem           size   used  avail capacity  Mounted on
/dev/dsk/c0t0d0s0     5.8G   4.1G   1.7G    72%       /
```

- Use the `format -e` command [expert mode] to determine the logical device name of the second drive on the client system as shown:

```
# format -e
Searching for disks...done
```

AVAILABLE DISK SELECTIONS:

- 0. c0t0d0 <SUN72G cyl 14087 alt 2 hd 24 sec 424>
/pci@1d,700000/scsi@4/sd@0,0
- 1. **c0t1d0** <SUN72G cyl 14087 alt 2 hd 24 sec 424>
/pci@1d,700000/scsi@4/sd@1,0

Specify disk (enter its number): **1**

selecting c0t1d0
[disk formatted]

- While still in the `format -e` utility, determine that the second drive does not have an EFI label on it. If the "Specify Label type" lists a "1" then the disk has an EFI label and it must be removed and labeled with an SMI label as shown:

```
(output omitted)
format> label
[0] SMI Label
[1] EFI Label
Specify Label type[1]: 0
Warning: This disk has an EFI label. Changing to SMI label will erase all
current partitions.
Continue? yes
Auto configuration via format.dat [no]? no
Auto configuration via generic SCSI-2 [no]? no
```

Exercise 3: (Optional) Creating a ZFS Mirrored Root Pool

4. Continue to use the format -e utility to repartition the second disk by putting all of the free space on the disk into slice 0 by using the "All Free Hog" method as shown:

```
format> partition
(output omitted)
partition> print
Current partition table (default):
Total disk cylinders available: 14087 + 2 (reserved cylinders)
```

| Part | Tag | Flag | Cylinders | Size | Blocks |
|------|------------|------|------------|----------|-----------------------|
| 0 | root | wm | 0 - 25 | 129.19MB | (26/0/0) 264576 |
| 1 | swap | wu | 26 - 51 | 129.19MB | (26/0/0) 264576 |
| 2 | backup | wu | 0 - 14086 | 68.35GB | (14087/0/0) 143349312 |
| 3 | unassigned | wm | 0 | 0 | (0/0/0) 0 |
| 4 | unassigned | wm | 0 | 0 | (0/0/0) 0 |
| 5 | unassigned | wm | 0 | 0 | (0/0/0) 0 |
| 6 | usr | wm | 52 - 14086 | 68.10GB | (14035/0/0) 142820160 |
| 7 | unassigned | wm | 0 | 0 | (0/0/0) 0 |

```
partition> modify
Select partitioning base:
 0. Current partition table (default)
 1. All Free Hog
```

```
Choose base (enter number) [0]? 1
```

| Part | Tag | Flag | Cylinders | Size | Blocks |
|------|------------|------|-----------|---------|-----------------------|
| 0 | root | wm | 0 | 0 | (0/0/0) 0 |
| 1 | swap | wu | 0 | 0 | (0/0/0) 0 |
| 2 | backup | wu | 0 - 14086 | 68.35GB | (14087/0/0) 143349312 |
| 3 | unassigned | wm | 0 | 0 | (0/0/0) 0 |
| 4 | unassigned | wm | 0 | 0 | (0/0/0) 0 |
| 5 | unassigned | wm | 0 | 0 | (0/0/0) 0 |
| 6 | usr | wm | 0 | 0 | (0/0/0) 0 |
| 7 | unassigned | wm | 0 | 0 | (0/0/0) 0 |

Do you wish to continue creating a new partition table based on above table[yes]? **yes**

Free Hog partition[6]? **0**

```
Enter size of partition '1' [0b, 0c, 0.00mb, 0.00gb]: 0
Enter size of partition '3' [0b, 0c, 0.00mb, 0.00gb]: 0
Enter size of partition '4' [0b, 0c, 0.00mb, 0.00gb]: 0
Enter size of partition '5' [0b, 0c, 0.00mb, 0.00gb]: 0
Enter size of partition '6' [0b, 0c, 0.00mb, 0.00gb]: 0
Enter size of partition '7' [0b, 0c, 0.00mb, 0.00gb]: 0
```

Exercise 3: (Optional) Creating a ZFS Mirrored Root Pool

| Part | Tag | Flag | Cylinders | Size | Blocks |
|------|------------|------|-----------|---------|-----------------------|
| 0 | root | wm | 0 - 14086 | 68.35GB | (14087/0/0) 143349312 |
| 1 | swap | wu | 0 | 0 | (0/0/0) 0 |
| 2 | backup | wu | 0 - 14086 | 68.35GB | (14087/0/0) 143349312 |
| 3 | unassigned | wm | 0 | 0 | (0/0/0) 0 |
| 4 | unassigned | wm | 0 | 0 | (0/0/0) 0 |
| 5 | unassigned | wm | 0 | 0 | (0/0/0) 0 |
| 6 | usr | wm | 0 | 0 | (0/0/0) 0 |
| 7 | unassigned | wm | 0 | 0 | (0/0/0) 0 |

Okay to make this the current partition table[yes]? **yes**
Enter table name (remember quotes): "**new**"

Ready to label disk, continue? **yes**

```
partition> quit  
(output omitted)  
format> quit
```

5. Edit the profile file for the client on the JumpStart server to include the required JumpStart keywords to configure the ZFS root pool with both disks mirrored and to create a Live Upgrade boot environment:

```
# vi /export/config/host_class  
install_type initial_install  
system_type standalone  
pool zpool1 auto auto mirror c0t0d0s0 c0t1d0s0  
bootenv installbe bename sol-10u8
```

The previous profile configuration performs an initial installation specified with `install_type initial_install` in a new ZFS root pool, identified with pool `zpool1`, whose size is automatically sized with the `auto` keyword to the size of the specified disks.

The swap area and dump device are automatically sized with other `auto` keywords, in a mirrored configuration of disks (with the `mirror` keyword and disks specified as `c0t0d0s0` and `c0t1d0s0`).

Boot environment characteristics are set with the `bootenv` keyword to install a new BE [Boot Environment] with the keyword `installbe` and a bename named `sol-10u8` is created.

Note – More information about Live Upgrade is covered in the next module.



Exercise 3: (Optional) Creating a ZFS Mirrored Root Pool

6. If the client system is running, bring it down to the ok prompt and enter the JumpStart command:

```
# init 0
```

```
ok boot net - install nowin
```

(output omitted)

Searching for JumpStart directory...

Using rules.ok from 192.168.1.200:/export/config.

Checking rules.ok file...

Using profile: host_class

Using finish script: finish_script

Executing JumpStart preinstall phase...

Searching for SolStart directory...

Checking rules.ok file...

Using begin script: install_begin

Using finish script: patch_finish

Executing SolStart preinstall phase...

Executing begin script "install_begin"...

Begin script install_begin execution completed.

Processing profile

- Saving Boot Environment Configuration
- Selecting cluster (SUNWCall)
- Selecting all disks
- Configuring boot device
- Configuring / (c0t0d0s0)
- Configuring (c0t1d0s0)
- Automatically configuring disks for Solaris operating system

Verifying disk configuration

Verifying space allocation

- Total software size: 4157.69 Mbytes

Preparing system for Solaris install

Configuring disk (c0t0d0)

- Creating Solaris disk label (VTOC)

Configuring disk (c0t1d0)

- Creating Solaris disk label (VTOC)
- Creating pool zpool1
- Creating swap zvol for pool zpool1
- Creating dump zvol for pool zpool1

Creating and checking file systems

- Creating zpool1/ROOT/sol-10u8 dataset

Beginning Solaris software installation

Starting software installation

```
SUNWocfd.....done. 4157.35 Mbytes remaining.  
SUNWlucfg.....done. 4157.27 Mbytes remaining.  
SUNWcsu.....done. 4141.81 Mbytes remaining.  
SUNWcsr.....done. 4137.45 Mbytes remaining.  
SUNWcsl.....done. 4122.53 Mbytes remaining.  
SUNWcnetr.....done. 4122.46 Mbytes remaining.  
SUNWcar.u.....done. 4121.81 Mbytes remaining.  
SUNWjdmk-base.....done. 4120.56 Mbytes remaining.  
SUNWcakr.u.....done. 4096.19 Mbytes remaining.  
SUNWkvm.u.....done. 4093.83 Mbytes remaining.  
SUNWckr.....done. 4081.43 Mbytes remaining.
```

(output ommitted)

- When the system has finished the JumpStart, login and examine the ZFS root pool configuration. The `zpool list` command shows that the creation of the root ZFS pool has been created:

```
# zpool list  
NAME      SIZE    USED   AVAIL    CAP  HEALTH  ALTROOT  
zpool1  14.1G  5.15G  8.97G  36%  ONLINE  -
```

- The `zpool status -v` command will show you the configuration of your ZFS mirrored pool:

```
# zpool status -v  
pool: zpool1  
  state: ONLINE  
    scrub: none requested  
  config:  
  
    NAME        STATE     READ WRITE CKSUM  
    zpool1      ONLINE      0      0      0  
      mirror    ONLINE      0      0      0  
        c0t0d0s0  ONLINE      0      0      0  
        c0t1d0s0  ONLINE      0      0      0  
  
errors: No known data errors
```

Exercise 3: (Optional) Creating a ZFS Mirrored Root Pool

9. Execute the `zfs list` command and it will show you the ZFS file systems that were created based on your profile file configuration:

```
# zfs list
NAME          USED   AVAIL   REFER  MOUNTPOINT
zpool1        5.59G  8.31G  96.5K  /zpool1
zpool1(ROOT)  4.22G  8.31G  18K    legacy
zpool1(ROOT/sol-10u8) 4.22G  8.31G  4.22G  /
zpool1/dump   896M   8.31G  896M   -
zpool1/export 38K    8.31G  20K    /export
zpool1/export/home 18K    8.31G  18K    /export/home
zpool1/swap    512M   8.75G  59.0M  -
```

10. Execute the `df -h` command to see all of the other file systems that were created during the JumpStart process:

```
# df -h
Filesystem      size   used  avail capacity  Mounted on
zpool1(ROOT/sol-10u8) 14G    4.2G  8.3G   34%    /
/devices        0K    0K    0K    0%    /devices
ctfs           0K    0K    0K    0%    /system/contract
proc           0K    0K    0K    0%    /proc
mnttab         0K    0K    0K    0%    /etc/mnttab
swap           345M   464K  344M   1%    /etc/svc/volatile
objfs          0K    0K    0K    0%    /system/object
sharefs         0K    0K    0K    0%    /etc/dfs/sharetab
fd              0K    0K    0K    0%    /dev/fd
swap           345M   160K  344M   1%    /tmp
swap           344M   40K   344M   1%    /var/run
zpool1/export  14G   20K   8.3G   1%    /export
zpool1/export/home 14G   18K   8.3G   1%    /export/home
zpool1        14G   96K   8.3G   1%    /zpool1
```

11. Examine the swap space using the `swap -l` command:

```
# swap -l
swapfile        dev  swaplo blocks   free
/dev/zvol/dsk/zpool1/swap 256,1     16 1048560 1048560
```

12. Finally, execute the `lustatus` command to examine the Live Upgrade boot environment that was created during the JumpStart installation:

```
# lustatus
Boot Environment      Is      Active Active Can Copy
Name                  Complete Now    On Reboot Delete Status
-----
sol-10u8            yes     yes     yes    no    -
```

13. Let your instructor know you have completed this exercise.

Troubleshooting JumpStart

If any of the four main JumpStart services are improperly configured, the JumpStart clients can:

- Fail to boot
- Fail to find a Solaris OS image to load
- Ask questions interactively for configuration
- Fail to partition disks or create file systems, and fail to load the Operating System

Resolving Boot Problems

Problems in the JumpStart client boot process are usually associated with RARP, TFTP, or BOOTPARAMS configuration issues. If the client issues error messages or fails to proceed with the boot process, it usually means that one of these services is not properly configured.

Resolving RARP Problems

If the JumpStart client fails to boot and repeatedly issues the following message:

Timeout waiting for ARP/RARP packet

then the JumpStart client cannot obtain RARP services from a boot server. Check to make sure `in.rarpd` is running on the server. This message probably indicates that the `/etc/ethers` or `/etc/inet/hosts` file on the boot server is not correctly configured. To correct this problem, edit these files, and ensure that the MAC address and host name for the client in the `/etc/ethers` file, and that the IP address and host name for the client in the `/etc/inet/hosts` file are correct.

Other problems to check for that can cause this error message:

- Name service not updated to reflect new entries in the `/etc/ethers` or `/etc/inet/hosts` files
- Physical network connections

Enter the commands required to update the name service in use. Usually, the messages these commands issue will indicate whether an update for the `/etc/ethers` or `/etc/inet/hosts` files was successful.

Check all of the physical network connections between the client and the boot server to eliminate a potential source of the updating problem.

Resolving TFTP Problems

If the JumpStart client issues the following message once and stops booting:

Timeout waiting for ARP/RARP packet

this message indicates that the JumpStart server cannot obtain TFTP services from a boot server.

Usually, this error message indicates that there is no entry for the JumpStart client in the `/tftpboot` directory on the boot server. An easy way to solve this problem is to run the `rm_install_client` script and then the `add_install_client` script for this client. For example:

```
# cd /export/install/Solaris_10/Tools  
# ./rm_install_client client1
```

```
# ./add_install_client -c server1:/export/config -p
server1:/export/config client1 sun4u
```

Other problems to check for that can cause this message to appear:

- The incorrect platform group argument to the add_install_client script was used (For example, specifying sun4m for a sun4u system).
- The boot server is not configured to allow the in.tftpd daemon to run on demand.

If you specify the incorrect platform group for the client when you run the add_install_client script, the client might hang, or issue additional error messages and panic early in the boot process. To solve this problem, run the rm_install_client script and then the add_install_client script, and specify the correct platform group.

If the boot server is not configured to allow the in.tftpd daemon to run on demand, the client hangs. Usually, the add_install_client script automatically modifies the boot server to provide this service. To correct this problem, run the following commands to enable the TFTP service.

Check to see if the TFTP service is available:

```
# inetadm | grep tftp
```

If the command does produce any output, edit the /etc/inet/inetd.conf file and ensure the following line is present:

```
# vi /etc/inet/inetd.conf
# TFTP - tftp server (primarily used for booting)
tftp dgram    udp6    wait root /usr/sbin/in.tftpd in.tftpd -s /tftpboot
```

If the line is commented out, uncomment it.

Run the command to import the service into SMF:

```
# inetconv
```

Check that the tftp service is now available:

```
# inetadm | grep ftp
enabled    online          svc:/network/ftp:default
enabled    online          svc:/network/tftp/udp6:default
#
```

Resolving BOOTPARAMS Problems

If the JumpStart client obtains RARP and TFTP responses, but stops booting after displaying a numeric value, such as:

23e00

the JumpStart client is unable to obtain BOOTPARAMS information from a boot server. This value indicates that the client was able to load its network bootstrap program. If no information for the client exists in the /etc/bootparams file, or if the `rpc.bootparamd` daemon is not running, this portion of the boot process fails.

If no entry exists in the /etc/bootparams file for the JumpStart client, create an entry by running the `add_install_client` script that automatically starts the `rpc.bootparamd` daemon.

The SMF starts the `rpc.bootparamd` daemon when the boot server boots. Logic in the `/var svc/milestone/network/tftp-udp6.xml` file checks for the `/tftpboot` directory, and starts the `rpc.bootparamd` daemon if the directory exists. Check if the `rpc.bootparamd` daemon is running:

```
# pgrep -fl bootparamd
```

If the `rpc.bootparamd` process is not running, check whether the `/tftpboot` directory exists. If it exists, manually start the `rpc.bootparamd` process with the following commands:

```
# svcs -a | grep bootparams
disabled      14:12  svc:/network/rpc/bootparams:default
# svcadm enable network/rpc/bootparams:default
# svcs | grep bootparams
online        14:20:33 svc:/network/rpc/bootparams:default
#
```

Resolving Identification Problems

Problems in the JumpStart client identification process usually relate to identification information missing from the `sysidcfg` file or from a name service. If a JumpStart client cannot obtain a response from a server for any identification item, the client interrupts the automatic identification process and asks for the information. The client usually indicates what information is missing, but not necessarily from what source.

Resolving sysidcfg Problems

In the absence of a name service, if the JumpStart client interrupts the identification or installation process to obtain any of the following identification items, check the sysidcfg file on the JumpStart server, and correct the problem you find:

- Will the client be configured to use IPv6 networking?
- What netmask will the client use?
- What is the IP address of the default router?
- What security policy will the client implement?
- What name service will the client use?
- What time zone will the client use?
- What system locale will the client use?
- What system will provide the time-of-day information?
- What is the root user's password?

Resolving Name Service Problems

If you use a name service, and the JumpStart client interrupts the identification process to obtain identification items *other than* the following, check the corresponding map or table information in the name service, and correct the problem you find:

- Will the client implement IPv6 protocols?
- What is the IP address of the default router?
- What security policy will the client implement?
- What is the root log in password?

The previous items can only be provided using the sysidcfg file.

You can use the sysidcfg file to provide information that a name service could otherwise provide. You must verify the content of the sysidcfg file or any information that it provides. Information provided in the sysidcfg file overrides information in name services.

Resolving Configuration Problems

Problems in the JumpStart client configuration process usually relate to improperly configured rules or profile files. If a JumpStart client cannot obtain a response from a server for any configuration item, or if the configuration information it finds is incompatible with the client's hardware, it interrupts the automatic configuration process.

The information that the client requests usually indicates what is missing or improperly configured. Incompatible configuration information causes the client to display a panel that describes the problem.

Resolving rules File Problems

Sometimes the JumpStart client completes its identification tasks, but then issues the following messages:

```
Checking rules.ok file...
```

```
Warning: Could not find matching rule in rules.ok
```

```
Press the return key for an interactive Solaris install program...
```

These messages indicate that it cannot find an entry in the rules.ok file that it matches.

Usually this happens because administrators fail to run the check script to generate an up-to-date rules.ok file. To correct this problem, verify that the rules file contains an entry that matches the client, and then run the check script. For example:

```
# ./check
```

```
Checking validity of rules...
```

```
Checking validity of profile1 file...
```

```
The auto-install configuration is ok.
```

```
#
```

Resolving Profile (Class) File Problems

If the JumpStart client completes its identification tasks, but then displays an error message, such as:

```
ERROR: Field 2 - Disk is not valid on this system (c0t4d0s0)
```

it indicates that a configuration error exists in the profile file it has selected.

To correct this error, edit the profile file that the client uses, and correct the problem indicated.

Resolving Installation Problems

Problems in the JumpStart client installation process usually relate to NFS configuration problems. If a server fails to share a directory that a JumpStart client requires, the installation cannot proceed.

Resolving NFS Problems

If the JumpStart client obtains RARP and TFTP responses, but panics and displays an error message similar to the following:

```
panic - boot: Could not mount filesystem  
Program terminated  
ok
```

the client cannot mount the root (/) file system defined in the /etc/bootparams file.

To correct this problem, edit the /etc/dfs/dfstab file on the boot server to ensure that it contains an entry that shares the required directory structure. Check the /etc/bootparams file on the boot server to determine what directory to share. For example, the /etc/dfs/dfstab file could contain the following entry to share the /export/install directory:

```
share -F nfs -o ro,anon=0 /export/install
```

The -o ro,anon=0 options are required for the client to use the root (/) file system properly.

Run the following commands to stop and start the NFS daemons on the boot server:

```
# svcadm disable network/nfs/server:default  
# svcadm enable network/nfs/server:default
```

If the JumpStart client issues an error message that indicates that it cannot mount any directory it requires or automatically begins an interactive installation session, verify the configuration of the /etc/dfs/dfstab file on the servers that provide the directories that the client requires. Make any required change in the servers' /etc/dfs/dfstab files, and stop and restart the NFS server daemons on those servers.

Any directory listed in the `/etc/bootparams` file on the boot server must be shared by the server providing the directory.

Resolving Begin and Finish Script Problems

Begin and finish script problems can be the most troublesome of all issues related to JumpStart. Any error possible in a shell script is possible in one of these scripts. Debugging begin and finish scripts might involve multiple attempts at booting the JumpStart client, or otherwise performing trial runs of the scripts.

After writing begin or finish scripts, you must verify that these scripts are referenced in the appropriate rule in the `rules` file. You must also remember to run the `check` script to regenerate the `rules.ok` file.

Resolving Syntax Problems

If the JumpStart client boots, displays the GUI interface in one window, and then the window disappears after the begin script runs, a syntax error might exist in your begin script.

To check for this problem on the JumpStart client, open a terminal window, and examine the `/tmp begin.log` file. This file contains standard output and error messages that the begin script generates. Correct any error it reports in the begin script and try booting the client again.

The JumpStart client behaves similarly when it encounters errors in finish scripts. If the JumpStart client abruptly closes the window in which the finish script is running, it is probable that a syntax error exists in your finish script.

To check for this problem, after the JumpStart client reboots, examine the `/var/sadm/system/logs/finish.log` file. This file contains standard output and error messages that the finish script generates. Correct any error it reports in the finish script, and try booting the client again.

Identifying Log Files

JumpStart clients retain the following log files during the installation process:

```
/tmp	begin.log  
/tmp	finish.log  
/tmp/install_log  
/var/sadm/system/logs/sysidtool.log
```

These logs contain standard output and error messages from begin scripts, finish scripts, the Solaris OS software installation process, and the system identification process that the client performs.

JumpStart clients retain a corresponding set of log files after the installation process completes and the system reboots:

```
/var/sadm/system/logs/begin.log  
/var/sadm/system/logs/finish.log  
/var/sadm/system/logs/install_log  
/var/sadm/system/logs/sysidtool.log
```

Notes:

Performing Live Upgrade Using the Solaris 10 Operating System

Objectives

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

- Describe the benefits of using Live Upgrade
- Describe the Solaris Live Upgrade process
- Describe Solaris Live Upgrade requirements
- Identify the Solaris Live Upgrade commands
- Create an alternate boot environment cloned from a running system
- Create a differential flash archive in a Live Upgrade boot environment
- Modify the state of the new boot environment
- Extend a base boot environment with a differential flash archive
- Use Live Upgrade to patch a system
- Using JumpStart to implement a Live Upgrade environment

Introducing Solaris Live Upgrade

Solaris Live Upgrade provides a method of upgrading a system while the system continues to operate. While your current boot environment is running, you can duplicate the boot environment, then upgrade the duplicate. Alternatively, rather than upgrading, you can install a Solaris Flash archive on a boot environment. The original system configuration remains fully functional and unaffected by the upgrade or installation of an archive. When you are ready, you can activate the new boot environment by rebooting the system. If a failure occurs, you can quickly revert to the original boot environment with a simple reboot. This switch eliminates the normal downtime of the test and evaluation process.

Solaris Live Upgrade enables you to duplicate a boot environment without affecting the currently running system. You can then do the following:

- Upgrade a system.
- Change the current boot environment's disk configuration to different file system types, sizes, and layouts on the new boot environment.
- Maintain numerous boot environments with different images. For example, you can create one boot environment that contains current patches and create another boot environment that contains an Update release.
- Apply new patches to a new boot environment instead of to the current system environment.

Some understanding of basic system administration is necessary before using Solaris Live Upgrade including information about system administration tasks such as managing file systems, mounting, booting, and managing swap.

Note – The only limitation to upgrading involves a Solaris Flash archive. When you use a Solaris Flash archive to install, an archive that contains non-global zones are not properly installed on your system.

A Solaris system that is configured with Trusted Extensions requires extra steps to upgrade labeled zones. For information on this procedure, see “Upgrading a Trusted Extensions System That is Configured with Labeled Zones” under “Installation Enhancements” in *Solaris 10 8/07 Release Notes*.



Solaris Live Upgrade Process

The process of using Live Upgrade to upgrade a Solaris system includes the following general phases:

- Creating an alternate boot environment (ABE) by cloning a current Solaris OS instance. The source for this cloning could also be a flash archive.
- Changing the state of the system in the ABE for reasons including the following:
 - Upgrading to another OS release
 - Updating a release with patches or updates
- Activating the new boot environment (BE)
- Optionally falling back to the original BE.

Multiple Release Compatibility



Note – The only limitation to upgrading involves a Solaris Flash archive. When you use a Solaris Flash archive to install, an archive that contains non-global zones are not properly installed on your system.

A Solaris system that is configured with Trusted Extensions requires extra steps to upgrade labeled zones. For information on this procedure, see “Upgrading a Trusted Extensions System That is Configured with Labeled Zones” under “Installation Enhancements” in *Solaris 10 8/07 Release Notes*.

Ensure that you have the most recently updated patch list by consulting <http://sunsolve.sun.com>. Search for the infodoc 206844 on the SunSolve web site.



Caution – Correct operation of Solaris Live Upgrade requires that a limited set of patch revisions be installed for a particular OS version. Before installing or running Solaris Live Upgrade, you are required to install these patches.

The patches listed in infodoc 206844 are subject to change at any time. These patches potentially fix defects in Solaris Live Upgrade, as well as fix defects in components that Solaris Live Upgrade depends on. If you experience any difficulties with Solaris Live Upgrade, please check and make sure that you have the latest Solaris Live Upgrade patches installed.

Solaris Live Upgrade System Requirements

Solaris Live Upgrade is included in the Solaris software. You need to install the Solaris Live Upgrade packages on your current OS. The release of the Solaris Live Upgrade packages that you use must match the release of the OS you are upgrading to. For example, if your current OS is the Solaris 9 release and you want to upgrade to the Solaris 10 05/09 release, you need to install the Solaris Live Upgrade packages from the Solaris 10 05/09 release.

Table 13-1 lists releases that are supported by Solaris Live Upgrade.

Table 13-1 Supported Solaris Releases

| Your Current Release | Compatible Upgrade Release |
|----------------------|---|
| Solaris 8 OS | Solaris 8, 9, or any Solaris 10 release |
| Solaris 9 OS | Solaris 9 or any Solaris 10 release |
| Solaris 10 OS | Any Solaris 10 release |

Solaris Live Upgrade Disk Space Requirements

To estimate the file system size that is needed to create a boot environment, start the creation of a new boot environment. The size is calculated. You can then abort the process.

The disk on the new boot environment must be able to serve as a boot device. Some systems restrict which disks can serve as a boot device. Refer to your system's documentation to determine if any boot restrictions apply.

The disk might need to be prepared before you create the new boot environment. Check that the disk is formatted properly:

- Identify slices large enough to hold the file systems to be copied.
- Identify file systems that contain directories that you want to share between boot environments rather than copy. If you want a directory to be shared, you need to create a new boot environment with the directory put on its own slice. The directory is then a file system and can be shared with future boot environments.

Guidelines for Selecting Slices for File Systems

When you create file systems for a boot environment, the rules are identical to the rules for creating file systems for the Solaris OS. Solaris Live Upgrade cannot prevent you from creating invalid configurations for critical file systems. For example, you could type a lucreate command that would create separate file systems for root (/) and /kernel which is an invalid division of the root (/) file system.

Do not overlap slices when re-slicing disks. If this condition exists, the new boot environment appears to have been created, but when activated, the boot environment does not boot. The overlapping file systems might be corrupted.

For Solaris Live Upgrade to work properly, the `fstab` file on the active boot environment must have valid contents and must have an entry for the root (/) file system at the minimum.

Guidelines for Selecting a Slice for the root (/) File System

When you create an inactive boot environment, you need to identify a slice where the root (/) file system is to be copied. Use the following guidelines when you select a slice for the root (/) file system. The slice must comply with the following:

- Must be a slice from which the system can boot.
- Must meet the recommended minimum size.
- Can be on different physical disks or the same disk as the active root (/) file system.
- Can be a Veritas Volume Manager volume (VxVM). If VxVM volumes are configured on your current system, the lucreate command can create a new boot environment. When the data is copied to the new boot environment, the Veritas file system configuration is lost and a UFS file system is created on the new boot environment.

Guidelines for Selecting a Slice for a swap File System

These guidelines contain configuration recommendations and examples for a swap slice.

Configuring swap for the New Boot Environment

You can configure a swap slice in three ways by using the lucreate command with the -m option:

- If you do not specify a swap slice, the swap slices belonging to the current boot environment are configured for the new boot environment.
- If you specify one or more swap slices, these slices are the only swap slices that are used by the new boot environment. The two boot environments do not share any swap slices.
- You can specify to both share a swap slice and add a new slice for swap.

The following examples show the three ways of configuring swap. The current boot environment is configured with the root (/) file system on c0t0d0s0. The swap file system is on c0t0d0s1.

- In the following example, no swap slice is specified. The new boot environment contains the root (/) file system on c0t1d0s0. swap is shared between the current and new boot environment on c0t0d0s1.

```
# lucreate -n be2 -m :/dev/dsk/c0t1d0s0:ufs
```

- In the following example, a swap slice is specified. The new boot environment contains the root (/) file system on c0t1d0s0. A new swap file system is created on c0t1d0s1. No swap slice is shared between the current and new boot environment.

```
# lucreate -n be2 -m :/dev/dsk/c0t1d0s0:ufs -m -:/dev/dsk/c0t1d0s1:swap
```

- In the following example, a swap slice is added and another swap slice is shared between the two boot environments. The new boot environment contains the root (/) file system on c0t1d0s0. A new swap slice is created on c0t1d0s1. The swap slice on c0t0d0s1 is shared between the current and new boot environment.

```
# lucreate -n be2 -m :/dev/dsk/c0t1d0s0:ufs -m -:shared:swap -m -:/dev/dsk/c0t1d0s1:swap
```

Failed Boot Environment Creation if swap is in Use

A boot environment creation fails if the swap slice is being used by any boot environment except for the current boot environment. If the boot environment was created using the -s option, the alternate-source boot environment can use the swap slice, but not any other boot environment.

Guidelines for Selecting Slices for Shareable File Systems

Solaris Live Upgrade copies the entire contents of a slice to the designated new boot environment slice. You might want some large file systems on that slice to be shared between boot environments rather than copied to conserve space and copying time.

File systems that are critical to the OS such as root (/) and /var must be copied. File systems such as /home are not critical file systems and could be shared between boot environments. Shareable file systems must be user-defined file systems and on separate swap slices on both the active and new boot environments. You can reconfigure the disk several ways, depending your needs.

Installing Solaris Live Upgrade

You can install the Solaris Live Upgrade packages by using the following:

- The pkgadd command. The Solaris Live Upgrade packages are SUNWlur, SUNWluu (and starting with the Solaris 10 05/09 release) the SUNWlucfg, and these packages must be installed in that order.
- An installer on the Solaris Operating System DVD, the Solaris Software - 2 CD, or a net installation image.

Creating a Boot Environment

The process of creating a boot environment provides a method of copying critical file systems from an current boot environment to a new boot environment. The disk is reorganized if necessary, file systems are customized, and the critical file systems are copied to the new boot environment.

File System Types

Solaris Live Upgrade distinguishes between two file system types: critical file systems and shareable.

Critical file systems are required by the Solaris OS. These file systems are separate mount points in the /etc/vfstab file of the current and inactive boot environments. These file systems are always copied from the source to the inactive boot environment. Critical file systems are sometimes referred to as non-shareable. Examples are root (/), /usr, /var, or /opt.

Shareable file systems are user-defined files such as /export that contain the same mount point in the /etc/vfstab file in both the current and inactive boot environments. Therefore, updating shared files in the current boot environment also updates data in the inactive boot environment. When you create a new boot environment, shareable file systems are shared by default. But you can specify a destination slice and then the file systems are copied.

In addition, swap is a special shareable file system. Like a shareable file system, all swap slices are shared by default. But, if you specify a destination directory for swap, the swap slice is copied.

Copying File Systems

The process of creating a new boot environment begins by identifying an unused slice where a critical file system can be copied. If a slice is not available or a slice does not meet the minimum requirements, you need to format a new slice.

After the slice is defined, you can reconfigure the file systems on the new boot environment before the file systems are copied into the directories. You reconfigure file systems by splitting and merging them, which provides a simple way of editing the /etc/vfstab file to connect and disconnect file system directories. You can merge file systems into their parent directories by specifying the same mount point. You can also split file systems from their parent directories by specifying different mount points.

After file systems are configured on the inactive boot environment, you begin the automatic copy. Critical file systems are copied to the designated directories. Shareable file systems are not copied, but are shared. The exception is that you can designate some shareable file systems to be copied. When the file systems are copied from the current to the inactive boot environment, the files are directed to the new directories. The current boot environment is not changed in any way.

Live Upgrade Commands

The following Table 13-2 briefly describes the commands used with Live Upgrade.

Table 13-2 Live Upgrade Commands

| LU Command | Description |
|----------------------|--|
| lu | A deprecated curses-based menu interface for creating and administering boot environments. |
| luactivate | Designate the specified boot environment as the one to boot from in subsequent boots. |
| lucancel | Cancel a scheduled Live Upgrade operation. |
| lucompare | Compare the contents of two boot environments. |
| lucreate | Create a boot environment. |
| lucurr | Display the name of the currently booted boot environment. |
| ludelete | Delete a boot environment. |
| lufslist | List the file systems of a specified boot environment. |
| lumake | Re-create a boot environment based on the current boot environment. |
| lumount/ luumount | Mount/unmount file systems of a specified boot environment. |
| lurename | Rename a boot environment. |
| lustatus | For every boot environment, list whether a boot environment is active, active upon the next boot, in the midst of a copy operation, and if a copy operation is scheduled for it. |
| luupgrade | Modify a boot environment by installing flash archives, installing a complete OS, installing and/or deleting OS and application packages, or installing OS patches. |

Example Procedure: Creating A Base Master Flash Archive

The following example procedure illustrates how to create a base master flash archive. Application of a differential flash archive involves first applying a base master flash archive and then applying a differential archive.

1. Make a full flash archive of the currently running system for use as the base master flash archive.

```
# mkdir /xxx ; cd /xxx  
# flarcreate -S -c -n master_sys_env_1 master_sys_env_1.flar
```

This flash archive will not be used until later in this procedure. It will be used to initially install a client system after which a differential flash archive will be installed on that client to extend its installed state.

2. Check the administrative information stored in the flash archive.

```
# flar info master_sys_env_1.flar  
archive_id=bce4466c276e17fde18d0ebaccd44615  
files_archived_method=cpio  
creation_date=20060225212333  
creation_master=sys-01  
content_name=master_sys_env_1creation_node=sys-01  
creation_hardware_class=sun4u  
creation_platform=SUNW,UltraAX-i2  
creation_processor=sparc  
...  
files_compressed_method=compress  
content_architectures=sun4u  
type=FULL
```

Example Procedure: Cloning An Alternate Boot Environment From a Running System

In this example we will create an ABE cloned from a running system by using the `luupgrade` command to extend the base ABE with a differential flash archive.

In this part of the procedure a new boot environment (`sys_env_2`) will be cloned from the currently running boot environment (`sys_env_1`). Refer to Figure 13-1. The single root file system will be copied over. The swap and `/export/home` partitions will be part of each boot environment.

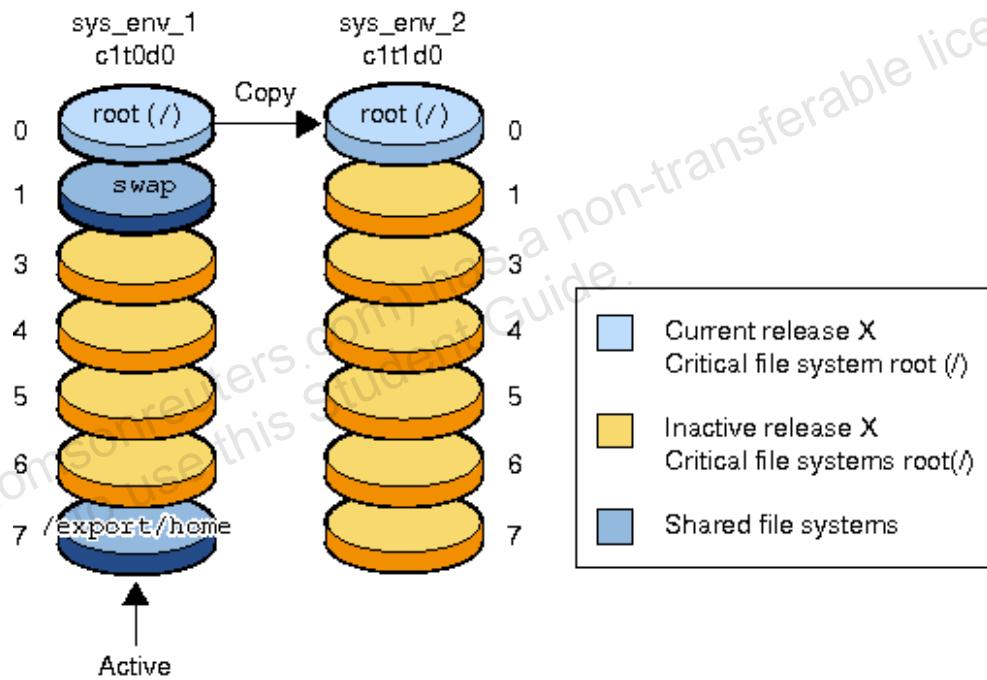


Figure 13-1 Cloning a New Boot Environment From a Running System

1. Prepare disk space for an alternate boot environment. By first examining the partitioning of disk 1, where the current boot environment is installed:

```
# prtvtoc /dev/rdsck/c1t0d0s2
* /dev/rdsck/c1t0d0s2 partition map
...
*                               First      Sector      Last
* Partition  Tag  Flags    Sector      Count     Sector   Mount Directory
          0    2    00    2097414  67963725  70061138   /
          1    3    01            0    2097414        2097413
          2    5    00            0   71127180  71127179
          4    0    00   70061139        8667   70069805
```

Example Procedure: Cloning An Alternate Boot Environment From a Running

```
5      0    00  70069806      8667  70078472  
7      8    00  70078473  1048707  71127179  /export/home
```

2. Examine the mounting of the current boot environment.

```
# mount  
...  
/ on /dev/dsk/c1t0d0s0 ...  
/export/home on /dev/dsk/c1t0d0s7 ...  
...
```

3. Partition the second disk to be identical to the first so that it can be used for the ABE.

```
# /usr/sbin/prvtvtoc /dev/rdsck/c1t0d0s2 | /usr/sbin/fmthard -s - \  
/dev/rdsck/c1t1d0s2
```

```
fmthard: New volume table of contents now in place.
```



Note – Having partitioning the same on both disks is a requirement for this example only. Live Upgrade can be used to implement partitioning changes. For example, if the original system has separate partitions and file systems for /, /usr and /var, the new environment can merge all of them into one partition and one file system.

4. Check that the partitioning on the second disk matches that of the first disk.

```
# prvtvtoc /dev/rdsck/c1t1d0s2  
* /dev/rdsck/c1t1d0s2 partition map  
*  
...  
  
*          First     Sector     Last  
* Partition Tag  Flags   Sector   Count   Sector Mount Directory  
  0        2    00  2097414  67963725  70061138  
  1        3    01          0  2097414  2097413  
  2        5    00          0  71127180  71127179  
  4        0    00  70061139      8667  70069805  
  5        0    00  70069806      8667  70078472  
  7        8    00  70078473  1048707  71127179
```

5. Create the alternative boot environment with these specifications:

- Name the current boot environment sys_env_1
- Name the new boot environment sys_env_2
- Arrange that /export/home will be shared between the environments

Example Procedure: Cloning An Alternate Boot Environment From a Running

- Match the file system - partition assignments for both environments

```
# lucreate -c "sys_env_1" -m /:/dev/dsk/c1t1d0s0:ufs -n "sys_env_2"
```

Discovering physical storage devices

Discovering logical storage devices

Cross referencing storage devices with boot environment configurations

Determining types of file systems supported

Validating file system requests

Preparing logical storage devices

Preparing physical storage devices

Configuring physical storage devices

Configuring logical storage devices

Analyzing system configuration.

No name for current boot environment.

Current boot environment is named <sys_env_1>.

Creating initial configuration for primary boot environment <sys_env_1>.

The device </dev/dsk/c1t0d0s0> is not a root device for any boot environment.

PBE configuration successful: PBE name <sys_env_1> PBE Boot Device </dev/dsk/c1t0d0s0>.

Comparing source boot environment <sys_env_1> file systems with the file system(s) you specified for the new boot environment. Determining which file systems should be in the new boot environment.

Updating boot environment description database on all BEs.

Searching /dev for possible boot environment filesystem devices

Updating system configuration files.

The device </dev/dsk/c1t1d0s0> is not a root device for any boot environment.

Creating configuration for boot environment <sys_env_2>.

Source boot environment is <sys_env_1>.

Creating boot environment <sys_env_2>.

Creating file systems on boot environment <sys_env_2>.

Creating <ufs> file system for </> on </dev/dsk/c1t1d0s0>.

Mounting file systems for boot environment <sys_env_2>.

Calculating required sizes of file systems for boot environment <sys_env_2>.

Populating file systems on boot environment <sys_env_2>.

Checking selection integrity.

Integrity check OK.

Populating contents of mount point </>.

Copying.

Creating shared file system mount points.

Creating compare databases for boot environment <sys_env_2>.

Creating compare database for file system </>.

Example Procedure: Cloning An Alternate Boot Environment From a Running

Updating compare databases on boot environment <sys_env_2>.

Making boot environment <sys_env_2> bootable.

Population of boot environment <sys_env_2> successful.

Creation of boot environment <sys_env_2> successful.

6. Examine both boot environments with the lufslist command.

```
# lufslist sys_env_1
```

```
boot environment name: sys_env_1
This boot environment is currently active.
This boot environment will be active on next system boot.
```

| Filesystem Options | fstype | device | size | Mounted on | Mount |
|--------------------|--------|-------------|--------------|------------|-------|
| /dev/dsk/c1t0d0s1 | swap | 1073875968 | - | | - |
| /dev/dsk/c1t0d0s0 | ufs | 34797427200 | / | | - |
| /dev/dsk/c1t0d0s7 | ufs | 536937984 | /export/home | | - |

```
# lufslist sys_env_2
```

```
boot environment name: sys_env_2
```

| Filesystem Options | fstype | device | size | Mounted on | Mount |
|--------------------|--------|-------------|--------------|------------|-------|
| /dev/dsk/c1t0d0s1 | swap | 1073875968 | - | | - |
| /dev/dsk/c1t1d0s0 | ufs | 34797427200 | / | | - |
| /dev/dsk/c1t0d0s7 | ufs | 536937984 | /export/home | | - |

Note that in the sys_env_2 environment listing, /export/home still shows on the first disk, c1t0d0. This is also true for swap. This is because both swap and /export/home are being shared between the two environments; they were not cloned to the new BE. Only the root file system shows on the second disk, c1t1d0s0. (When the source of the cloning contains separate file systems for /, /usr, /var, or /opt, these critical file systems are required for the new boot environment and therefore will be copied.)

Example Procedure: Cloning An Alternate Boot Environment From a Running

7. Use the `lulstatus` command to check the status of the boot environments.

```
# lufsstatus
```

| Boot Environment Name | Is Complete | Active Now | Active On Reboot | Can Delete | Copy Status |
|-----------------------|-------------|------------|------------------|------------|-------------|
| sys_env_1 | yes | yes | yes | no | - |
| sys_env_2 | yes | no | no | yes | - |

Note that `sys_env_1` is currently active and will be in effect on next system boot. The `sys_env_2` BE has been cloned and therefore complete but not now active.

8. View the contents of the compare file created in `/etc/lu/compare`

```
# cd /etc/lu/compare
```

```
# ls
```

```
sys_env_1:sys_env_2
```

```
# more sys_env_1:sys_env_2
```

```
:/root:root:22:40755:DIR:  
/lost+found:root:root:2:40700:DIR:  
/export:root:sys:3:40755:DIR:  
/var:28385:100:44:40775:DIR:  
/var/sadm:root:other:13:40755:DIR:  
/var/sadm/install:root:bin:4:40555:DIR:  
/var/sadm/install/admin:root:bin:2:40555:DIR:  
...
```

9. When you are ready to switch and make the new boot environment active, you use the `luactivate` command to activate the new boot environment and reboot. Files are synchronized between boot environments the first time that you boot a newly created boot environment. Activate the `sys_env_2` environment with the `luactivate` command.

```
# luactivate sys_env_2
```

```
*****
```

The target boot environment has been activated. It will be used when you reboot. NOTE: You MUST NOT USE the `reboot`, `halt`, or `uadmin` commands. You MUST USE either the `init` or the `shutdown` command when you reboot. If you do not use either `init` or `shutdown`, the system will not boot using the target BE.

```
*****
```

Example Procedure: Cloning An Alternate Boot Environment From a Running

In case of a failure while booting to the target BE, the following process needs to be followed to fallback to the currently working boot environment:

1. Enter the PROM monitor (ok prompt).
2. Change the boot device back to the original boot environment by typing:

```
setenv boot-device /pci@1f,0/pci@1/scsi@8/disk@0,0:a
```

3. Boot to the original boot environment by typing:

```
boot
```

```
*****
```

Activation of boot environment <sys_env_2> successful.

10. Use the `lustatus` command to see the change in status.

```
# lustatus
Boot Environment      Is      Active Active      Can      Copy
Name                Complete Now    On Reboot Delete Status
-----
sys_env_1            yes     yes     no       no      -
sys_env_2            yes     no      yes      no      -
```

Note that the `sys_env_2` environment is not yet active. It will become active on the next boot, however, because the `boot-device` OBP variable has been configured for the new environment. Make note of the procedure for booting the original environment as output in the `lucreate` command in case the new environment doesn't boot properly.

11. Use the `init 6` command to finish making `sys_env_2` the currently running environment.

```
# init 6
```

12. When the system comes back up, login and verify that the `sys_env_2` environment is active with the `lustatus` command.

```
# lustatus
Boot Environment      Is      Active Active      Can      Copy Name
Name                Complete Now    On Reboot Delete Status
-----
sys_env_1            yes     no      no       yes      -
sys_env_2            yes     yes     yes      no      -
```

Falling Back to the Original Boot Environment

If a failure occurs, you can quickly fall back to the original boot environment with an activation and reboot. The use of fallback takes only the time to reboot the system, which is much quicker than backing up and restoring the original. The new boot environment that failed to boot is preserved. The failure can then be analyzed. You can only fall back to the boot environment that was used by `luactivate` to activate the new boot environment.

You fall back to the previous boot environment the following ways:

Table 13-3 Solaris Live Upgrade Fallback Problems and Actions

| Problem | Action |
|--|--|
| The new boot environment boots successfully, but you are not happy with the results. | Run the <code>luactivate</code> command with the name of the previous boot environment and reboot. x86/x64 only: Starting with the Solaris 10 1/06 release, you can fall back by selecting the original boot environment that is found on the GRUB menu. The original boot environment and the new boot environment must be based on the GRUB software |
| The new boot environment does not boot. | Boot the fallback boot environment in single-user mode, run the <code>luactivate</code> command, and reboot. |
| You cannot boot in single-user mode. | Perform one of the following: <ul style="list-style-type: none">• Boot from DVD or CD media or a net installation image• Mount the root (/) file system on the fallback boot environment• Run the <code>luactivate</code> command and reboot |

Example Procedure: Modifying the State of the New Boot Environment

As explained in the Live Upgrade process summary earlier in the module, the state of the system can be changed in many ways depending on your reasons for implementing Live Upgrade. It could be to implement the next Solaris Express release or applying a set of updates/patches.

In this example and referring to Figure 13-2, a simple modification will be made for instructional purposes. A simple package will be added. When a differential archive is created later in this procedure, the difference captured in that archive will be the inclusion of this package.

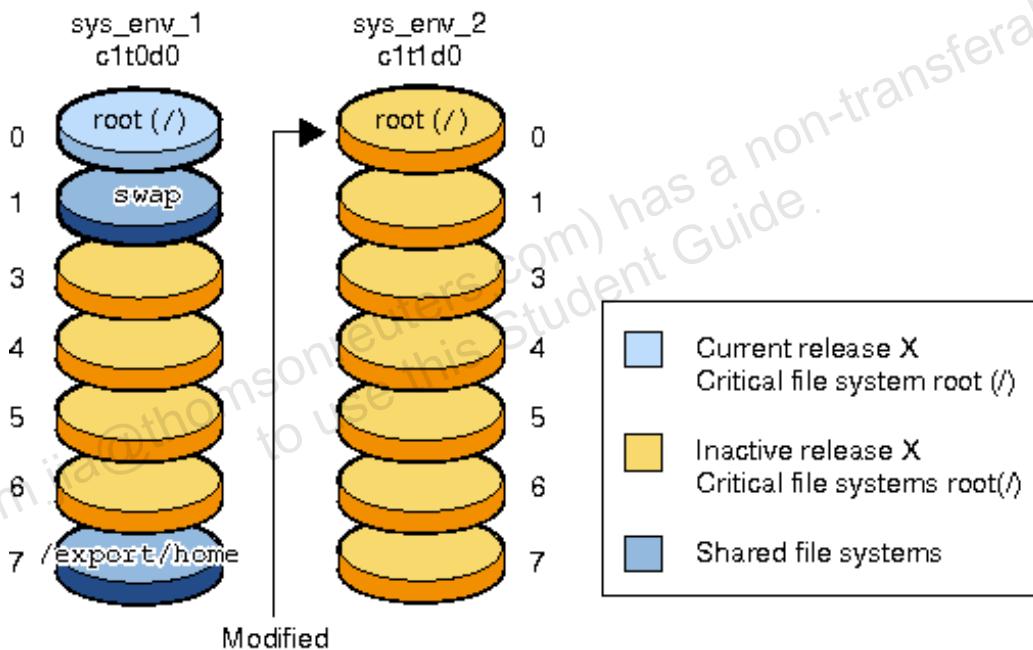


Figure 13-2 Modified Boot Environment

1. Modify the system state of the sys_env_2 environment by adding the SMCTop package to the system.

```
# cd /var/spool/pkg
# pkgadd -d .
```

The following packages are available:

```
1 SMCTop      top
(sparsc) 3.5.1
```

Select package(s) you wish to process (or 'all' to process all packages). (default: all) [?,??,q]: **1**

Example Procedure: Modifying the State of the New Boot Environment

2. Verify that the new package has been added.

```
# pkginfo -l SMCTop
```

```
PKGINST: SMCTop
```

```
...
```

3. Use the lucompare command to compare the two boot environments.

```
# lucompare -t -o ./environ_compare_2_to_1 sys_env_1
```

```
Determining the configuration of "sys_env_1"...
```

```
Comparing / ...
```

4. Examine the first few lines of the compare file to see the type of information it contains.

```
# more environ_compare_2_to_1
```

```
< sys_env_2
```

```
> sys_env_1
```

```
Sizes differ
```

```
01 < /var/sadm/install/contents:root:root:1:100644:REGFIL:22638869:
```

```
02 > /var/sadm/install/contents:root:root:1:100644:REGFIL:22637090:
```

```
Checksums differ
```

```
01 <
```

```
/var/sadm/install/.lockfile:root:root:1:100600:REGFIL:128:1845941275:
```

```
02 >
```

```
/var/sadm/install/.lockfile:root:root:1:100600:REGFIL:128:582217747:
```

```
Sizes differ
```

```
01 < /var/sadm/pkg/SUNWcsu/pkginfo:root:root:1:100644:REGFIL:7214:
```

```
02 > /var/sadm/pkg/SUNWcsu/pkginfo:root:root:1:100644:REGFIL:5897:
```

```
...
```

Example Procedure: Creating a Differential Archive Using Live Upgrade Boot Environments

In this section of the procedure a differential flash archive is created capturing the changes between the original system and the evolved system as illustrated in Figure 13-3.

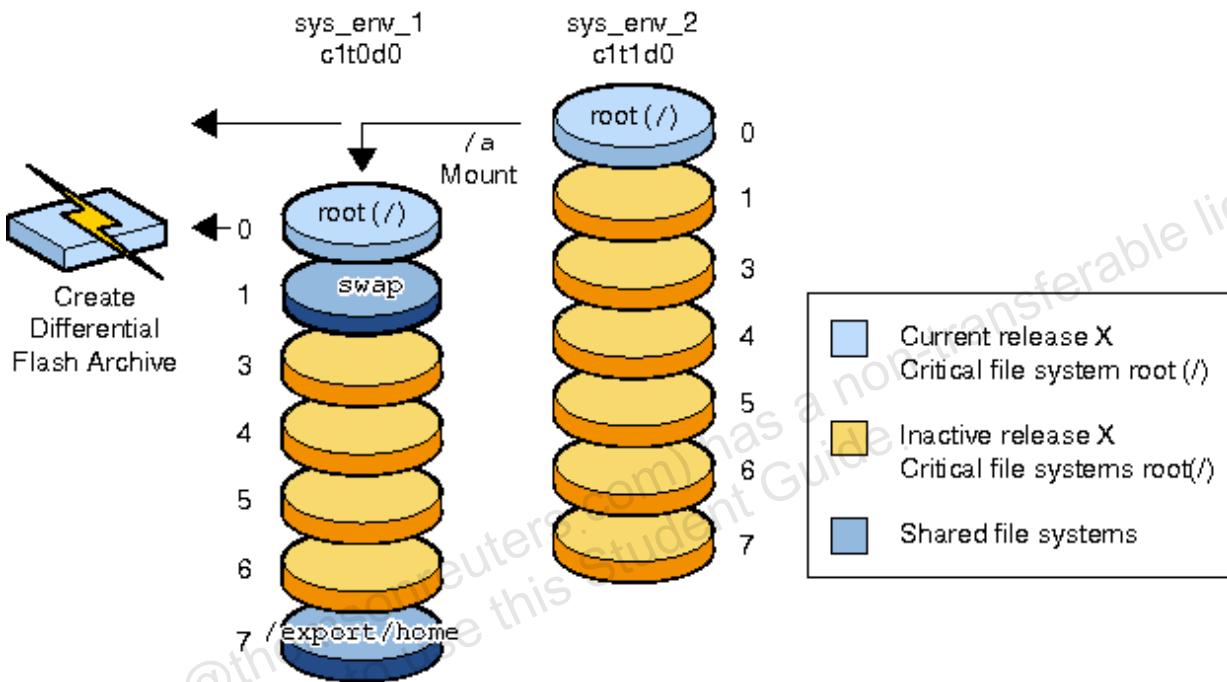


Figure 13-3 Creating a Differential Flash Archive in the Live Upgrade Environment

1. Prepare to create a differential flash archive by mounting the inactive environment (`sys_env_1`) on `/a` in the active environment with the `lumount` command.

```
# mkdir /a
# lumount sys_env_1 /a
/a
```

2. Use the `mount` command to see the original environment mounted.

```
# mount
...
/a on /dev/dsk/c1t0d0s0 ...
...
```

3. Create a differential archive which captures the difference between the current active environment and the inactive sys_env_1 environment mounted on /a. Exclude the flash archive (-x /a/xxx) that was created in the beginning of this procedure which now resides in the /a/xxx/ directory. Call the new differential archive differ_flar_on_sys_env_1_new_pkg.flar and store it in the /a/xxx directory. Dispense with the size check (-S) and compress the archive (-c).

```
# flarcreate -n differential_flash -S -c -A /a -x /a/xxx \
/a/xxx/differ_flar_on_sys_env_1_new_pkg.flar
Differential Flash
Checking integrity...
Integrity OK.
Running precreation scripts...
Precreation scripts done.
Creating the archive...
437639 blocks
Archive creation complete.
Running postcreation scripts...
Postcreation scripts done.

Running pre-exit scripts...
Pre-exit scripts done.
```

4. Use the flar info command to see the administrative information stored with the archive.

```
# flar info differ_flar_on_sys_env_1_new_pkg.flar
archive_id=c04e27bfc16c1c32cfa04cfa359217d6
files_archived_method=cpio
creation_date=20060226011846
creation_master=sys-01
content_name=differential_flash
creation_node=sys-01
creation_hardware_class=sun4u
creation_platform=SUNW,UltraAX-i2
creation_processor=sparc
creation_release=5.10
creation_os_name=SunOS
creation_os_version=Generic_118822-25
files_compressed_method=compress
content_architectures=sun4u
type=DIFFERENTIAL
```

Example Procedure: Applying a Differential Flash Archive Using Live Upgrade BE's

The next section of this procedure demonstrates one way of applying a differential archive. Typically this will involve installing a client with the original flash archive made at the beginning of the procedure and then extending that client's installed state by applying the differential archive. In this example however, Live Upgrade will be used on the same system to make a blank or empty third boot environment (-s - option) which will be upgraded to the an initial installed state using the master flash archive and then extended using the differential flash archive. Figure 13-4 illustrates the boot environments involved.

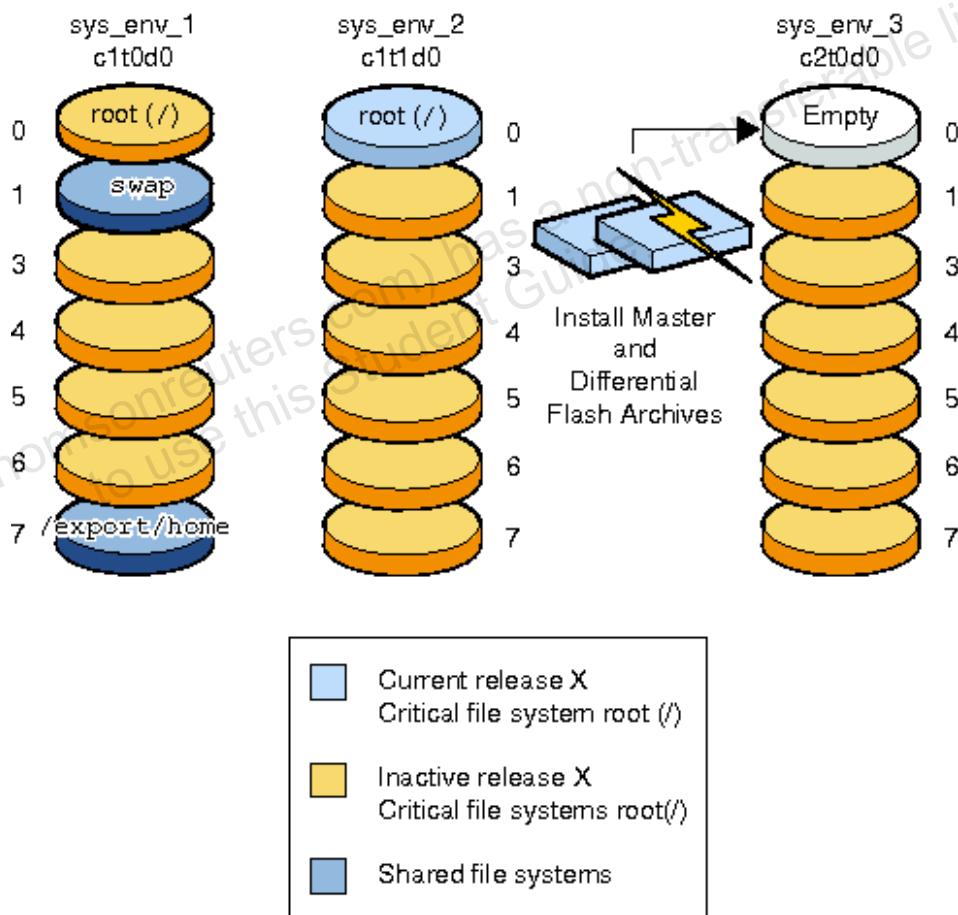


Figure 13-4 Applying Flash Archives to a Boot Environment

1. Prepare a third disk by partitioning it like the others.

```
# /usr/sbin/prvtvtoc /dev/rdsck/c1t1d0s2 | /usr/sbin/fmthard -s - \
/dev/rdsck/c2t0d0s2
fmthard: New volume table of contents now in place.
```

Example Procedure: Applying a Differential Flash Archive Using Live Upgrade BE's

2. Before making the new boot environment, unmount /a with the luumount command.

```
# luumount /a
```

3. Create a new boot environment with the following specifications:

- Use c2t0d0
- Do not clone a boot environment. Use the -s - option to make it empty
- Name the new boot environment sys_env_3

When prompted for the / and swap devices via the menu, select those devices appropriate for the new boot environment that is being created.

```
# lucreate -n "sys_env_3" -s -
```

...

Updating system configuration files.

...

Since lucreate cannot determine the new / device on its own, the menu appears and you need to specify, with the F2, ENTER and F3 keys, the / and swap devices:

| | | | | | | Active boot environment - None |
|-------------|--------|--------|---------|----------------------------------|--------|--------------------------------|
| Mount Point | | Device | FS Type | Size (MB) | % Used | |
| <hr/> | | | | | | |
| | | | | New boot environment - sys_env_3 | | |
| <hr/> | | | | | | |
| Recommended | | | | | | |
| Mount Point | | Device | FS Type | Size (MB) | Min | |
| Size (MB) | | | | | | |
| / | | | ufs | | 0 | |
| - | | | swap | | 0 | |
| | | | | | | |
| ESC | F2 | F3 | F4 | F5 | F6 | F7 |
| HELP | CHOICE | SAVE | SLICE | PRINT | CANCEL | SCHEDULE |
| | | | | | | SPLIT |
| | | | | | | MERGE |
| | | | | | | CLR |
| | | | | | | OTHR |

In this example, for the above menu interaction, c2t0d0s0 was specified for the / device and c2t0d0s1 was specified for the swap device. The F2 key is used to display a drop down menu from which to select the devices (using the ENTER key). When finished, the F3 key is used to save the configuration and then the menu exits and output continues.

Example Procedure: Applying a Differential Flash Archive Using Live Upgrade BE's

The device </dev/dsk/c2t0d0s2> is not a root device for any boot environment.

Creating <ufs> file system for </> on </dev/dsk/c2t0d0s2>. Creation of boot environment <sys_env_3> successful.



Note – The menu appeared because the root file system location was not specified on the lucreate command line. The menu would not have appeared if this command were used instead:

```
# lucreate -n "sys_env_3" -s - -m ::/dev/dsk/c2t0d0s0:ufs
```

4. Use the lustatus command to see all statuses for the boot environments.

```
# lustatus
```

| Boot Environment Name | Is Complete | Active Now | Active On Reboot | Can Delete | Copy Status |
|-----------------------|-------------|------------|------------------|------------|-------------|
| sys_env_1 | yes | no | no | yes | - |
| sys_env_2 | yes | yes | yes | no | - |
| sys_env_3 | no | no | no | yes | - |

Note how sys_env_3 is not complete. It is empty or blank.

5. Make the master archive and differential archive images available on the local file system. (The archive was saved in the sys_env_1 BE and needs to be copied to the current sys_env_2 BE).

```
# mount /dev/dsk/c1t0d0s0 /a  
# cd /a/xxx  
# cp master* diff* /  
# umount /a
```

6. Make an install image accessible.

```
# mkdir /net2  
# mount 192.168.201.1:/export/install /net2  
# mount  
...  
/net2 on 192.168.201.1:/export/install...  
...
```

7. Use the luupgrade command to populate the new sys_env_3 BE with the master full flash archive. First use dry run method (-N).

```
# luupgrade -f -n sys_env_3 -s /net2/SunOS5.10_0509_sun4 -a \  
/master_sys_env_1.flar -N -l /errorlog
```

Validating the contents of the media </net2/SunOS5.10_0509_sun4>. The media is a standard Solaris media.
 Validating the contents of the miniroot </net2/SunOS5.10_0509_sun4/Solaris_10/Tools/Boot>.
 Locating the flash install program.
 Checking for existence of previously scheduled Live Upgrade requests.
 Constructing flash profile to use.
 Creating flash profile for BE <sys_env_3>.
 Performing the operating system flash of the BE <sys_env_3>.
 Execute Command:
</net2/SunOS5.10_0509_sun4/Solaris_10/Tools/Boot/usr/sbin/install.d/pfinstall -L /a -p / -t /tmp/.luupgrade.translist.tmp.24446 -o /net2/SunOS5.10_0509_sun4/Solaris_10/Tools/Boot /tmp/.luupgrade.profile.flash.24446>.

8. Run the luupgrade command again but this time without the dry run option.

```
# luupgrade -f -n sys_env_3 -s /net2/SunOS5.10_0509_sun4 -a \
/master_sys_env_1.flar -l /errorlog
```

Validating the contents of the media </net2/SunOS5.10_0509_sun4>. The media is a standard Solaris media.
 Validating the contents of the miniroot </net2/SunOS5.10_0509_sun4/Solaris_10/Tools/Boot>.
 Locating the flash install program.
 Checking for existence of previously scheduled Live Upgrade requests.
 Constructing flash profile to use.
 Creating flash profile for BE <sys_env_3>.
 Performing the operating system flash install of the BE <sys_env_3>.
 CAUTION: Interrupting this process may leave the boot environment unstable or unbootable.

...

Extracting Flash Archive: 100% completed (of 4640.55 megabytes)
 The operating system flash install completed.

The Live Flash Install of the boot environment <sys_env_3> is complete. Use the lustatus command to check the status of the new environment.

lustatus

| Boot Environment Name | Is Complete | Active Now | Active On Reboot | Can Delete | Copy Status |
|-----------------------|-------------|------------|------------------|------------|-------------|
| sys_env_1 | yes | no | no | yes | - |
| sys_env_2 | yes | yes | yes | no | - |
| sys_env_3 | yes | no | no | yes | - |

Note that now sys_env_3 shows being complete, but still not active.

Example Procedure: Applying a Differential Flash Archive Using Live Upgrade BE's

9. Create a profile file to reference in for applying the differential archive.

```
# cat /profile
install_type flash_update
archive_location local_file /differ_flar_on_sys_env_1_new_pkg.flar
no_content_check
no_master_check
```

10. Use the luupgrade command to apply the differential flash archive to the new sys_env_3 BE. Reference the profile just created.

```
# luupgrade -f -n sys_env_3 -s /net2/SunOS5.10_0509_sun4 -j /profile \
-l /errorlog
```

Validating the contents of the media </net2/SunOS5.10_0509_sun4>.

The media is a standard Solaris media.

Validating the contents of the miniroot

</net2/SunOS5.10_0509_sun4/Solaris_10/Tools/Boot>.

Locating the flash install program.

Checking for existence of previously scheduled Live Upgrade requests.

Constructing flash profile to use.

Performing the operating system flash update of the BE <sys_env_3>.

CAUTION: Interrupting this process may leave the boot environment unstable or unbootable.

Extracting Flash Archive: 100% completed (of 162.01 megabytes)

The operating system flash update completed.

The Live Flash Update of the boot environment <sys_env_3> is complete.

11. Check the status of the BE.

```
# lustatus
Boot Environment      Is      Active Active      Can      Copy
Name                Complete Now    On Reboot Delete Status -----
-----
```

| Boot Environment | Is Complete | Active Now | Active On Reboot | Can Delete | Copy Status |
|------------------|-------------|------------|------------------|------------|-------------|
| sys_env_1 | yes | no | no | yes | - |
| sys_env_2 | yes | yes | yes | no | - |
| sys_env_3 | yes | no | no | yes | - |

12. Make sys_env_3 active.

```
# luactivate sys_env_3
```

The target boot environment has been activated. It will be used when you reboot. NOTE: You MUST NOT USE the reboot, halt, or uadmin commands. You

Example Procedure: Applying a Differential Flash Archive Using Live Upgrade BE's

MUST USE either the init or the shutdown command when you reboot. If you do not use either init or shutdown, the system will not boot using the target BE.

```
*****
```

In case of a failure while booting to the target BE, the following process needs to be followed to fallback to the currently working boot environment:

1. Enter the PROM monitor (ok prompt).
2. Change the boot device back to the original boot environment by typing:

```
setenv boot-device /pci@1f,0/pci@1/scsi@8/disk@1,0:a
```

3. Boot to the original boot environment by typing:

```
boot
```

```
*****
```

Activation of boot environment <sys_env_3> successful.

13. Check the status now.

```
# lustatus
Boot Environment      Is      Active Active   Can      Copy
Name                 Complete Now    On Reboot Delete Status
-----
sys_env_1            yes     no      no       yes     -
sys_env_2            yes     yes     no       no      -
sys_env_3            yes     no      yes     no      -
```

14. Since the lustatus command reports that the next system reboot will activate the sys_env_3 BE, note the procedure to fall back to the current boot environment. Then, at the system console, reboot the system with the init 6 command.

```
# init 6
```

15. When the system comes back up, use the lustatus command to verify that the sys_env_3 BE is now active.

```
# lustatus
Boot Environment      Is      Active Active   Can      Copy
Name                 Complete Now    On Reboot Delete Status
```

Example Procedure: Applying a Differential Flash Archive Using Live Upgrade BE's

| | | | | | |
|-----------|-----|-----|-----|-----|---|
| sys_env_1 | yes | no | no | yes | - |
| sys_env_2 | yes | no | no | yes | - |
| sys_env_3 | yes | yes | yes | no | - |

16. Verify that the differential archive has been applied by verifying that the SMCTop package is included in the system.

```
# pkginfo -l SMCTop
PKGINST: SMCTop
...
```

Reverting to a previous BE

17. Make sys_env_1 the active and currently running environment again.

```
# luactivate sys_env_1
# init 6
```

18. Use the lustatus command to verify that sys_env_1 is again active and currently running.

```
# lustatus
Boot Environment      Is      Active   Active   Can     Copy
Name        Complete Now    On Reboot Delete Status
-----
sys_env_1       yes     yes     yes     no      -
sys_env_2       yes     no      no      yes     -
sys_env_3       yes     no      no      yes     -
```

Note – The ludelete *be_name* command will delete a boot environment. It must first be made inactive.



Example Procedure: Using Live Upgrade To Patch A System

Live Upgrade can be used to patch a system safely. Used in conjunction with a ZFS root pool and the process is relatively easy. Live Upgrade can also be used on systems with Solaris Zone containers.

Use the following steps when you need to upgrade or patch a ZFS root file system with zone roots on ZFS. These updates can either be a system upgrade or the application of patches.

In the steps that follow, newBE, is the example name of the boot environment that is upgraded or patched.

Note – This example uses the system that was JumpStarted in Exercise 3: (Optional) Creating a ZFS Mirrored Root Pool from Module 13.



1. Examine the state of your system using the `lustatus` command:

```
# lustatus
Boot Environment      Is          Active   Active   Can      Copy
Name                  Complete    Now      On Reboot Delete Status
-----
sol-10u7              yes        yes     yes      no      -
```

2. Create the boot environment to upgrade or patch using the `lucreate` command. In this example it is called newBE:

```
# lucreate -n newBE
```

Analyzing system configuration.

Comparing source boot environment <sol-10u7> file systems with the file system(s) you specified for the new boot environment. Determining which file systems should be in the new boot environment.

Updating boot environment description database on all BEs.

Updating system configuration files.

Creating configuration for boot environment <newBE>.

Source boot environment is <sol-10u7>.

Creating boot environment <newBE>.

Cloning file systems from boot environment <sol-10u7> to create boot environment <newBE>.

Example Procedure: Using Live Upgrade To Patch A System

Creating snapshot for <zpool1/ROOT/sol-10u7> on <zpool1/ROOT/sol-10u7@newBE>.

Creating clone for <zpool1/ROOT/sol-10u7@newBE> on <zpool1/ROOT/newBE>.

Setting cammount=noauto for </> in zone <global> on <zpool1/ROOT/newBE>.

Population of boot environment <newBE> successful.

Creation of boot environment <newBE> successful.

3. Examine the state of the boot environments using the **lustatus** command:

lustatus

| Boot Environment Name | Is Complete | Active Now | Active On Reboot | Can Delete | Copy Status |
|-----------------------|-------------|------------|------------------|------------|-------------|
| sol-10u7 | yes | yes | yes | no | - |
| newBE | yes | no | no | yes | - |

4. Examine the state of the ZFS file systems using the **zfs list** command:

zfs list

| NAME | USED | AVAIL | REFER | MOUNTPOINT |
|----------------------------|-------|-------|-------|--------------|
| zpool1 | 5.59G | 8.31G | 97.5K | /zpool1 |
| zpool1/ROOT | 4.22G | 8.31G | 18K | legacy |
| zpool1/ROOT/newBE | 91K | 8.31G | 4.22G | / |
| zpool1/ROOT/sol-10u7 | 4.22G | 8.31G | 4.22G | / |
| zpool1/ROOT/sol-10u7@newBE | 76.5K | - | 4.22G | - |
| zpool1/dump | 896M | 8.31G | 896M | - |
| zpool1/export | 38K | 8.31G | 20K | /export |
| zpool1/export/home | 18K | 8.31G | 18K | /export/home |
| zpool1/swap | 512M | 8.75G | 59.0M | - |

Notice that a new ZFS file system was created for the Alternate Boot Environment (ABE). In this example it is zpool1/ROOT/newBE.

5. Apply patches to the new boot environment using the **luupgrade** command:

luupgrade -t -n newBE -t -s /patchmdir 139099-02 118666-19

Validating the contents of the media </patchmdir>.

The media contains 2 software patches that can be added.

Mounting the BE <newBE>.

Adding patches to the BE <newBE>.

Validating patches...

Loading patches installed on the system...

Done!

Loading patches requested to install.

Done!

The following requested patches have packages not installed on the system
Package SUNWgtarS from directory SUNWgtarS in patch 139099-02 is not
installed on the system. Changes for package SUNWgtarS will not be
applied to the system.

Package SUNWj5jmp from directory SUNWj5jmp in patch 118666-19 is not
installed on the system. Changes for package SUNWj5jmp will not be
applied to the system.

Checking patches that you specified for installation.

Done!

Approved patches will be installed in this order:

139099-02 118666-19

Checking installed patches...

Verifying sufficient filesystem capacity (dry run method) ...

Installing patch packages...

Patch 139099-02 has been successfully installed.

See /a/var/sadm/patch/139099-02/log for details

Patch packages installed:

SUNWgtar

SUNWsfman

Checking installed patches...

Verifying sufficient filesystem capacity (dry run method) ...

Installing patch packages...

Patch 118666-19 has been successfully installed.

See /a/var/sadm/patch/118666-19/log for details

Patch packages installed:

SUNWj5cfg

Example Procedure: Using Live Upgrade To Patch A System

```
SUNWj5dev  
SUNWj5dmo  
SUNWj5man  
SUNWj5rt
```

Unmounting the BE <newBE>.

The patch add to the BE <newBE> completed.

```
#
```

6. Activate the new boot environment using the luactivate command after the updates to the new boot environment are complete.

```
# luactivate newBE
```

7. Boot from newly-activated boot environment using the init 6 command:

```
# init 6
```

8. Verify that the boot environment that was patched is now active:

```
# llistatus
```

| Boot Environment | Is Complete | Active Now | Active On Reboot | Can Delete | Copy Status |
|------------------|-------------|------------|------------------|------------|-------------|
| sol-10u7 | yes | no | no | yes | - |
| newBE | yes | yes | yes | no | - |

Upgrading by Using a JumpStart Profile

You can create a JumpStart profile to use with Solaris Live Upgrade. If you are familiar with the custom JumpStart program, this is the same profile that custom JumpStart uses.

A JumpStart profile can be used to customize the upgrade of a boot environment, or to install a Solaris Flash archive into a boot environment.

When you install the Solaris OS with a Solaris Flash archive, the archive and the installation media must contain identical OS versions. For example, if the archive is the Solaris 10 operating system and you are using DVD media, then you must use Solaris 10 DVD media to install the archive. If the OS versions do not match, the installation on the target system fails. Identical operating systems are necessary when you use the following keyword or command:

- archive_location keyword in a profile
- luupgrade command with -s, -a, -j, and -J options

Note – When you use Live Upgrade to install the Solaris OS using a Solaris Flash archive, a separate OS image is still required for the installation process to complete. The Solaris OS image provides a miniroot that Live Upgrade requires, however, the boot environment is loaded using data from the Flash archive. The versions of the Solaris OS image and the Solaris Flash archive must match.



Use a text editor to create a the profile file. Name the file descriptively. Ensure that the name of the profile reflects how you intend to use the profile to install the Solaris software on a system. For example, you might name this profile upgrade_Solaris_10. Ensure that root owns the profile and that the permissions are set to 644.

JumpStart Profile Keywords Used with Solaris Live Upgrade

Only a subset of possible JumpStart profile keywords can be used in a Solaris Live Upgrade profile. Profiles used with Solaris Live Upgrade must include this keyword:

- `install_type`

Defines whether to upgrade the Solaris OS in a boot environment, or install a Solaris Flash archive or differential archive into a boot environment. Use the following values with this keyword:

- `upgrade` - for an upgrade
- `flash_install` - for a Solaris Flash installation
- `flash_update` - for a Solaris Flash differential installation

If the `install_type` keyword is set to `flash_install` or `flash_update`, you must also include the `archive_location` keyword.

- `archive_location`

Retrieves a Solaris Flash archive from a designated location.

JumpStart profiles used with Solaris Live Upgrade may include the following keywords. Depending on the type of installation declared by the `install_type` keyword, different groups of these keywords are valid within the profile.

- `archive_location`
- `cluster`
- `geo`
- `local_customization`
- `locale`
- `package`
- `forced_deployment`
- `local_customization`
- `no_content_check`
- `no_master_check`

For more information keywords that are used in Solaris Live Upgrade profiles, refer to the Solaris 10 05/09 Installation Guide: Solaris Live Upgrade and Upgrade Planning, available at this URL:

<http://docs.sun.com/app/docs/doc/819-6396?l=en&q=819-6396>

Creating a Solaris Live Upgrade Profile

In this example, a profile provides the upgrade parameters. This profile is to be used to upgrade an inactive boot environment with the Solaris Live Upgrade luupgrade command and the -u and -j options. This profile adds a package and a cluster. A regional locale and additional locales are also added to the profile. If you add locales to the profile, make sure that you have created a boot environment with additional disk space.

| # profile keywords | profile values |
|--------------------|----------------|
| # ----- | ----- |
| install_type | upgrade |
| package | SUNWxwman add |
| cluster | SUNWCacc add |
| geo | C_Europe |
| locale | zh_TW |
| locale | zh_TW.BIG5 |
| locale | zh_TW.UTF-8 |
| locale | zh_HK.UTF-8 |
| locale | zh_HK.BIG5HK |
| locale | zh |
| locale | zh_CN.GB18030 |
| locale | zh_CN.GBK |
| locale | zh_CN.UTF-8 |

Creating a Solaris Live Upgrade Profile to Install a Flash Archive

The following example of a profile is to be used by Solaris Live Upgrade to install a Flash archive into an empty boot environment. The Solaris Flash archive is retrieved from an NFS server. This profile is to be used with the Solaris Live Upgrade luupgrade command and the -j option.

| # profile keywords | profile values |
|--------------------|--|
| # ----- | ----- |
| install_type | flash_install |
| archive_location | nfs installserver:/export/solaris/archive/full_archive |

Creating a Solaris Live Upgrade Profile to Install a Differential Archive

The following example of a profile is to be used by Solaris Live Upgrade to install a differential archive into a boot environment. Only files that are specified by the differential archive are added, deleted, or changed. The Solaris Flash archive is retrieved from an NFS server. Because the image was built by the original master system, the clone system is not checked for a valid system image. This profile is to be used with the Solaris Live Upgrade luupgrade command and the -u and -j options.

```
# profile keywords          profile values
# -----
install_type               flash_update
archive_location           nfs installserver:/export/solaris/archive/diff_archive
no_master_check
```

Upgrading a Boot Environment by Using a Custom JumpStart Profile (Command-Line Interface)

In this example, the second_disk boot environment is upgraded by using a profile. The -j option is used to access the profile. The -s option specifies the source Solaris OS image. The boot environment is then ready to be activated. The pkgadd command adds the Solaris Live Upgrade packages from the release you are upgrading to.

```
# pkgadd -d /server/packages SUNWlur SUNWluu SUNWlucfg
# luupgrade -u -n second_disk \
-s /net/installmachine/export/solarisX/OS_image \
-j /var/tmp/profile
#
```

To Install a Solaris Flash Archive on a Boot Environment (Command-Line Interface)

1. Become superuser or assume an equivalent role
2. Install the Solaris Live Upgrade SUNWlur SUNWluu and SUNWlucfg packages on your system. These packages must be from the release you are upgrading to.
3. Type:

```
# luupgrade -f -n BE_name -s os_image_path -a archive
```

Where:

- -f

Indicates to install an operating system from a Solaris Flash archive.

- -n BE_name

Specifies the name of the boot environment that is to be installed with an archive.

- -s os_image_path

Specifies the path name of a directory that contains an operating system image. This directory can be on an installation medium, such as a DVD-ROM, CD-ROM, or it can be an NFS or UFS directory. The OS image is required to provide a miniroot to Solaris Live Upgrade.

- **-a archive**

Path to the Solaris Flash archive when the archive is available on the local file system. The operating system image versions that are specified with the **-s** option and the **-a** option must be identical.

In the following example, an archive is installed on the `second_disk` boot environment. The archive is located on the local system. The operating system versions for the **-s** and **-a** options are both Solaris10 05/09 releases.

All files are overwritten on `second_disk` except shareable files. The `pkgadd` command adds the Solaris Live upgrade packages from the release you are upgrading to.

```
# pkgadd -d /server/packages SUNWlur SUNWluu SUNWlucfg  
# luupgrade -f -n second_disk \  
-s /net/installmachine/export/Solaris_10/OS_image \  
-a /net/server/archive/10
```

The boot environment is ready to be activated.

To Install a Solaris Flash Archive With a Profile (Command-Line Interface)

This procedure provides the steps to install a Solaris Flash archive or differential archive by using a profile.

1. Become superuser or assume an equivalent role.
2. Create a profile. In this example, a profile provides the location of the archive to be installed.

```
# profile keywords          profile values
# -----
install_type           flash_install
archive_location        nfs installserver:/export/solaris/flasharchive/solarisarchive
```

3. After creating the profile, you can run the luupgrade command and install the archive. The -j option is used to access the profile. The pkgadd command adds the Solaris Live Upgrade packages from the release you are upgrading to. For example:

```
# pkgadd -d /server/packages SUNWlur SUNWluu SUNWlucfg
# luupgrade -f -n second_disk \
-s /net/installmachine/export/solarisX/OS_image \
-j /var/tmp/profile
```

Note – The -s option specifies the spooled Solaris OS image that the luupgrade command uses to find miniroot. The boot environment installs using data from the Flash archive.

Exercise: Patching and Upgrading Using Solaris Live Upgrade

In this exercise, you complete the following tasks:

- Create a new UFS boot environment
- Create a new ZFS boot environment
- Patch a UFS Boot Environment
- Upgrade the UFS Boot Environment to ZFS Boot Environment
- Boot into the new ZFS Boot Environment

Preparation

Before starting with the lab exercise, install the packages, SUNWoptdir and SUNWusrdir, found in the /opt/ses/lab/packages directory, on the student system.

```
# pkgadd -d /opt/ses/lab/packages SUNWoptdir  
# pkgadd -d /opt/ses/lab/packages SUNWusrdir
```

This exercise uses a dummy patch, 123456-01.zip or 654321-01.zip, found in the /opt/ses/lab/patches directory. It also requires the Solaris 10 OS Software DVD appropriate for your system architecture.

Note – The packages and patches used in this exercise are for demonstration only.

Further, refer to the lecture notes and additional resources to perform the steps listed in this exercise.



Task 1 - Creating a New UFS Boot Environment

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.

2. Use the format utility to create three partitions on the spare disk in your system. Set slice 4 to use 10gb, slice 5 to use 20 gb, and set partition 6 to use the remainder of the disk. The “all free hog” method available in the partition menu of the format utility may offer the easiest method.
3. Use the newfs command to create a UFS file system on slice 4 that you created. Replace the example slice name with the one that is correct for your system.
4. Use the lucreate command to create a new UFS boot environment with the following parameters:
 5. **-c: BE1**
 6. **-n: BE2**
 7. **-m: /:/dev/dsk/c0t1d0s4:ufs**
 8. Use the lustatus command to display the information about the boot environments.
 9. Use the lufslist command to list the configuration of the new boot environment, **BE2**.

Task 2 - Creating a New ZFS Boot Environment

Complete the following steps:

1. Use the newfs command to create a UFS file system on slice 5 that you created. Replace the example slice name with the one that is correct for your system.
2. Use the zpool command to create a ZFS root pool, **rpool**.
3. Use the lucreate command to create a new ZFS boot environment with the following parameters:
 4. **-n: BE3**
 5. **-p: rpool**
 6. Use the lustatus command to display the information about the boot environments.
 7. Use the lufslist command to list the configuration of the new ZFS boot environment, **BE3**.

Task 3 - Patching a UFS Boot Environment

Complete the following tasks:

1. Copy the patch archive, 123456-01.zip (SPARC) found in /opt/ses/lab/patches directory to the /var/tmp directory.
2. Use the unzip command to extract the patch from the .zip archive.
3. Use the luupgrade command to patch the UFS boot environment, BE2 with the patch, 123456-01.
4. Examine the patch installation log file.

Task 4 - Upgrading UFS Boot to ZFS Boot

Complete the following steps:

1. Insert the Solaris 10 Software SPARC or x64/x86 DVD into the drive and ensure that the media is mounted.
2. Use the luupgrade command to upgrade the UFS boot environment, **BE2** to a new ZFS boot environment, **BE3**.
3. Eject the Solaris 10 OS Software DVD.

Task 5 - Booting to a ZFS Boot Environment

Complete the following steps:

1. Use the luactivate command to activate the new ZFS boot environment, **BE3**.
2. Use the init 6 command to reboot the system to the new ZFS boot environment.
3. After reboot, use the df command to display information for the root (/) file system. In the output, notice that the root (/) file system is now ZFS-enabled.

Exercise Summary



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

- Experiences
- Interpretations
- Conclusions
- Applications

Exercise Solution: Patching and Upgrading Using Live Upgrade

In this exercise, you complete the following tasks:

- Create a new UFS boot environment
- Create a new ZFS boot environment
- Patch a UFS Boot Environment
- Upgrade a UFS Boot Environment to a ZFS Boot Environment
- Boot to the ZFS Boot Environment

Preparation

Before starting with the lab exercise, install the packages, SUNWoptdir and SUNWusrdir, found in the /opt/ses/lab/packages directory, on the student system.

```
# pkgadd -d /opt/ses/lab/packages SUNWoptdir  
# pkgadd -d /opt/ses/lab/packages SUNWusrdir
```

This exercise uses a dummy patch, 123456-01.zip or 654321-01.zip, found in the /opt/ses/lab/patches directory. It also requires the Solaris 10 OS Software DVD appropriate for your system architecture.

Note – The packages and patches used in this exercise are for demonstration only.

Further, refer to the lecture notes and additional resources to perform the steps listed in this exercise.



Task 1 - Creating a New UFS Boot Environment

Complete the following steps:

1. On the Sun Secure Global Desktop, click the icon for opening a console session on your assigned lab system.

2. Use the format utility to create three partitions on the spare disk in your system. Set slice 4 to use 10gb, slice 5 to use 20 gb, and set partition 6 to use the remainder of the disk. The “all free hog” method available in the partition menu of the format utility may offer the easiest method.
3. Use the newfs command to create a UFS file system on slice 4 that you created. Replace the example slice name with the one that is correct for your system.
- ```
newfs /dev/rdsk/c0t1d0s4
```
4. Use the lucreate command to create a new UFS boot environment with the following parameters:
- -c: BE1
  - -n: BE2
  - -m :/dev/dsk/c0t1d0s4:ufs
- ```
# lucreate -c BE1 -n BE2 -m :/dev/dsk/c0t1d0s4:ufs
```
5. Use the lustatus command to display the information about the boot environments.
- ```
lustatus
```
6. Use the lufslist command to list the configuration of the new boot environment, BE2.
- ```
# lufslist BE2
```

Task 2 - Creating a New ZFS Boot Environment

Complete the following steps:

1. Use the newfs command to create a UFS file system on slice 5 that you created. Replace the example slice name with the one that is correct for your system.
- ```
newfs /dev/rdsk/c0t1d0s5
```
2. Use the zpool command to create a ZFS root pool, rpool.
- ```
# zpool create -f rpool c0t1d0s5
```
3. Use the lucreate command to create a new ZFS boot environment with the following parameters:
- -n: BE3
 - -p: rpool
- ```
lucreate -n BE3 -p rpool
```

4. Use the `lustatus` command to display the information about the boot environments.

```
lustatus
```

5. Use the `lufslist` command to list the configuration of the new ZFS boot environment, **BE3**.

```
lufslist BE3
```

## Task 3 - Patching a UFS Boot Environment

Complete the following tasks:

1. Copy the patch archive, `123456-01.zip` (SPARC) found in `/opt/ses/lab/patches` directory to the `/var/tmp` directory.

```
cd /opt/ses/lab/patches
cp 123456-01.zip /var/tmp
```

2. Use the `tar` command to extract the patch from the `.tar` archive.

```
cd /var/tmp
unzip 123456-01.zip
```

3. Use the `luupgrade` command to patch the UFS boot environment, **BE2** with the patch, `121430-31`.

```
luupgrade -n BE2 -t -s /var/tmp 123456-01
```

## Task 4 - Upgrading UFS Boot to ZFS Boot

Complete the following steps:

1. Insert the Solaris 10 Software SPARC or x64/x86 DVD into the drive and ensure that the media is mounted to `/cdrom/cdrom0`.

2. Use the `luupgrade` command to upgrade the UFS boot environment, **BE2** to a new ZFS boot environment, **BE3**.

```
luupgrade -n BE3 -u -s /cdrom/cdrom0
```

3. Eject the Solaris 10 OS Software DVD.

## Task 5 - Booting to a ZFS Boot Environment

Complete the following steps:

1. Use the luactivate command to activate the new ZFS boot environment, **BE3**.

```
luactivate BE3
```

2. Use the init 6 command to reboot the system to the new ZFS boot environment.

```
init 6
```

3. After reboot, use the df command to display information for the root (/) file system. In the output, notice that the root (/) file system is now ZFS-enabled.

```
df -h
```

Unauthorized reproduction or distribution prohibited. Copyright© 2010, Oracle and/or its affiliates.

Tim Jia (tim.jia@thomsonreuters.com) has a non-transferable license  
to use this Student Guide.