# System Administration Guide: IP Services

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 816–4554–15
April 2008

080401@19860

# Contents

**Preface** ..... 29

**Part I Introducing System Administration: IP Services** ..... 35

**1 Solaris TCP/IP Protocol Suite (Overview)** ..... 37
What's New in This Release ..... 37
Introducing the TCP/IP Protocol Suite ..... 37
Protocol Layers and the Open Systems Interconnection Model ..... 38
TCP/IP Protocol Architecture Model ..... 39
How the TCP/IP Protocols Handle Data Communications ..... 44
Data Encapsulation and the TCP/IP Protocol Stack ..... 45
TCP/IP Internal Trace Support ..... 48
Finding Out More About TCP/IP and the Internet ..... 48
Computer Books About TCP/IP ..... 48
TCP/IP and Networking Related Web Sites ..... 48
Requests for Comments and Internet Drafts ..... 49

**Part II TCP/IP Administration** ..... 51

**2 Planning Your TCP/IP Network (Tasks)** ..... 53
Network Planning (Task Map) ..... 54
Determining the Network Hardware ..... 55
Deciding on an IP Addressing Format for Your Network ..... 55
IPv4 Addresses ..... 56
IPv4 Addresses in CIDR Format ..... 56
DHCP Addresses ..... 56
IPv6 Addresses ..... 57

Private Addresses and Documentation Prefixes ..... 57
Obtaining Your Network's IP Number ..... 57
Designing an IPv4 Addressing Scheme ..... 58
Designing Your IPv4 Addressing Scheme ..... 59
IPv4 Subnet Number ..... 60
Designing Your CIDR IPv4 Addressing Scheme ..... 61
Using Private IPv4 Addresses ..... 62
How IP Addresses Apply to Network Interfaces ..... 62
Naming Entities on Your Network ..... 63
Administering Host Names ..... 63
Selecting a Name Service and Directory Service ..... 63
Planning for Routers on Your Network ..... 65
Network Topology Overview ..... 66
How Routers Transfer Packets ..... 67

**3 Introducing IPv6 (Overview) ..... 69**
Major Features of IPv6 ..... 69
Expanded Addressing ..... 70
Address Autoconfiguration and Neighbor Discovery ..... 70
Header Format Simplification ..... 70
Improved Support for IP Header Options ..... 70
Application Support for IPv6 Addressing ..... 70
Additional IPv6 Resources ..... 71
IPv6 Network Overview ..... 72
IPv6 Addressing Overview ..... 74
Parts of the IPv6 Address ..... 74
Abbreviating IPv6 Addresses ..... 75
Prefixes in IPv6 ..... 75
Unicast Addresses ..... 76
Multicast Addresses ..... 79
Anycast Addresses and Groups ..... 79
IPv6 Neighbor Discovery Protocol Overview ..... 79
IPv6 Address Autoconfiguration ..... 81
Stateless Autoconfiguration Overview ..... 81
Overview of IPv6 Tunnels ..... 82

**4 Planning an IPv6 Network (Tasks)** ..... 83
IPv6 Planning (Task Maps) ..... 83
IPv6 Network Topology Scenario ..... 84
Preparing the Existing Network to Support IPv6 ..... 86
Preparing the Network Topology for IPv6 Support ..... 86
Preparing Network Services for IPv6 Support ..... 87
Preparing Servers for IPv6 Support ..... 87
▼ How to Prepare Network Services for IPv6 Support ..... 88
▼ How to Prepare DNS for IPv6 Support ..... 88
Planning for Tunnels in the Network Topology ..... 89
Security Considerations for the IPv6 Implementation ..... 90
Preparing an IPv6 Addressing Plan ..... 90
Obtaining a Site Prefix ..... 90
Creating the IPv6 Numbering Scheme ..... 91

**5 Configuring TCP/IP Network Services and IPv4 Addressing (Tasks)** ..... 93
What's New in This Chapter ..... 94
Before You Configure an IPv4 Network (Task Map) ..... 94
Determining Host Configuration Modes ..... 95
Systems That Should Run in Local Files Mode ..... 95
Systems That Are Network Clients ..... 96
Mixed Configurations ..... 97
IPv4 Network Topology Scenario ..... 97
Adding a Subnet to a Network (Task Map) ..... 98
Network Configuration Task Map ..... 98
Configuring Systems on the Local Network ..... 99
▼ How to Configure a Host for Local Files Mode ..... 100
▼ How to Set Up a Network Configuration Server ..... 102
Configuring Network Clients ..... 104
▼ How to Configure Hosts for Network Client Mode ..... 104
▼ How to Change the IPv4 Address and Other Network Configuration Parameters ..... 105
Packet Forwarding and Routing on IPv4 Networks ..... 109
Routing Protocols Supported by the Solaris OS ..... 110
IPv4 Autonomous System Topology ..... 113
Configuring an IPv4 Router ..... 115

▼ How to Configure an IPv4 Router ..... 116
Routing Tables and Routing Types ..... 121
Configuring Multihomed Hosts ..... 124
▼ How to Create a Multihomed Host ..... 125
Configuring Routing for Single-Interface Systems ..... 127
▼ How to Enable Static Routing on a Single-Interface Host ..... 128
▼ How to Enable Dynamic Routing on a Single-Interface Host ..... 130
Monitoring and Modifying Transport Layer Services ..... 132
▼ How to Log the IP Addresses of All Incoming TCP Connections ..... 132
▼ How to Add Services That Use the SCTP Protocol ..... 133
▼ How to Use TCP Wrappers to Control Access to TCP Services ..... 136
Administering Interfaces in Solaris 10 3/05 ..... 137
What's New in This Section ..... 137
Configuring Physical Interfaces in Solaris 10 3/05 ..... 137
▼ How to Add a Physical Interface After Installation in Solaris 10 3/05 ONLY ..... 138
▼ How to Remove a Physical Interface in Solaris 10 3/05 ONLY ..... 140
Configuring VLANs in Solaris 10 3/05 ONLY ..... 141
▼ How To Configure Static VLANs in Solaris 10 3/05 ONLY ..... 142

**6 Administering Network Interfaces (Tasks) ..... 145**
What's New in Administering Network Interfaces ..... 145
Interface Administration (Task Map) ..... 146
Basics for Administering Physical Interfaces ..... 146
Network Interface Names ..... 147
Plumbing an Interface ..... 148
Solaris OS Interface Types ..... 148
Administering Individual Network Interfaces ..... 148
▼ How to Obtain Interface Status ..... 149
▼ How to Configure a Physical Interface After System Installation ..... 150
▼ How to Remove a Physical Interface ..... 153
▼ SPARC: How to Ensure That the MAC Address of an Interface Is Unique ..... 154
Administering Virtual Local Area Networks ..... 156
Overview of VLAN Topology ..... 156
Planning for VLANs on a Network ..... 158
▼ How to Plan a VLAN Configuration ..... 159

Configuring VLANs ..... 159
▼ How to Configure a VLAN ..... 160
Overview of Link Aggregations ..... 161
Link Aggregation Basics ..... 162
Back-to-Back Link Aggregations ..... 164
Policies and Load Balancing ..... 164
Aggregation Mode and Switches ..... 165
Requirements for Link Aggregations ..... 165
▼ How to Create a Link Aggregation ..... 165
▼ How to Modify an Aggregation ..... 168
▼ How to Remove an Interface From an Aggregation ..... 169
▼ How to Delete an Aggregation ..... 170

**7 Configuring an IPv6 Network (Tasks) ..... 171**
Configuring an IPv6 Interface ..... 171
Enabling IPv6 on an Interface (Task Map) ..... 172
▼ How to Enable an IPv6 Interface for the Current Session ..... 172
▼ How to Enable Persistent IPv6 Interfaces ..... 174
▼ How to Turn Off IPv6 Address Autoconfiguration ..... 176
Configuring an IPv6 Router ..... 176
IPv6 Router Configuration (Task Map) ..... 176
▼ How to Configure an IPv6-Enabled Router ..... 177
Modifying an IPv6 Interface Configuration for Hosts and Servers ..... 181
Modifying an IPv6 Interface Configuration (Task Map) ..... 181
Using Temporary Addresses for an Interface ..... 181
▼ How to Configure a Temporary Address ..... 182
Configuring an IPv6 Token ..... 185
▼ How to Configure a User-Specified IPv6 Token ..... 185
Administering IPv6-Enabled Interfaces on Servers ..... 187
▼ How to Enable IPv6 on a Server's Interfaces ..... 187
Tasks for Configuring Tunnels for IPv6 Support (Task Map) ..... 188
Configuring Tunnels for IPv6 Support ..... 189
▼ How to Manually Configure IPv6 Over IPv4 Tunnels ..... 189
▼ How to Manually Configure IPv6 Over IPv6 Tunnels ..... 190
▼ How to Configure IPv4 Over IPv6 Tunnels ..... 191

▼ How to Configure a 6to4 Tunnel ..... 192
▼ How to Configure a 6to4 Tunnel to a 6to4 Relay Router ..... 195
Configuring Name Service Support for IPv6 ..... 197
▼ How to Add IPv6 Addresses to DNS ..... 197
Adding IPv6 Addresses to NIS ..... 198
▼ How to Display IPv6 Name Service Information ..... 198
▼ How to Verify That DNS IPv6 PTR Records Are Updated Correctly ..... 199
▼ How to Display IPv6 Information Through NIS ..... 200
▼ How to Display IPv6 Information Independent of the Name Service ..... 200

**8 Administering a TCP/IP Network (Tasks) ..... 203**
Major TCP/IP Administrative Tasks (Task Map) ..... 203
Monitoring the Interface Configuration With the `ifconfig` Command ..... 204
▼ How to Get Information About a Specific Interface ..... 205
▼ How to Display Interface Address Assignments ..... 206
Monitoring Network Status With the `netstat` Command ..... 208
▼ How to Display Statistics by Protocol ..... 209
▼ How to Display the Status of Transport Protocols ..... 210
▼ How to Display Network Interface Status ..... 212
▼ How to Display the Status of Sockets ..... 212
▼ How to Display the Status of Transmissions for Packets of a Specific Address Type ..... 214
▼ How to Display the Status of Known Routes ..... 215
Probing Remote Hosts With the `ping` Command ..... 216
▼ How to Determine if a Remote Host Is Running ..... 216
▼ How to Determine if a Host Is Dropping Packets ..... 216
Administering and Logging Network Status Displays ..... 217
▼ How to Control the Display Output of IP-Related Commands ..... 217
▼ How to Log Actions of the IPv4 Routing Daemon ..... 219
▼ How to Trace the Activities of the IPv6 Neighbor Discovery Daemon ..... 219
Displaying Routing Information With the `traceroute` Command ..... 220
▼ How to Find Out the Route to a Remote Host ..... 221
▼ How to Trace All Routes ..... 221
Monitoring Packet Transfers With the `snoop` Command ..... 222
▼ How to Check Packets From All Interfaces ..... 222
▼ How to Capture `snoop` Output Into a File ..... 223

▼ How to Check Packets Between an IPv4 Server and a Client ..... 224
▼ How to Monitor IPv6 Network Traffic ..... 225
Administering Default Address Selection ..... 225
▼ How to Administer the IPv6 Address Selection Policy Table ..... 226
▼ How to Modify the IPv6 Address Selection Table for the Current Session Only ..... 227

**9 Troubleshooting Network Problems (Tasks) ..... 229**
What's New in Troubleshooting Network Problems ..... 229
General Network Troubleshooting Tips ..... 229
Running Basic Diagnostic Checks ..... 230
▼ How to Perform Basic Network Software Checking ..... 230
Common Problems When Deploying IPv6 ..... 231
IPv4 Router Cannot Be Upgraded to IPv6 ..... 231
Problems After Upgrading Services to IPv6 ..... 231
Current ISP Does Not Support IPv6 ..... 231
Security Issues When Tunneling to a 6to4 Relay Router ..... 232
Known Issues With a 6to4 Router ..... 232

**10 TCP/IP and IPv4 in Depth (Reference) ..... 235**
What's New in TCP/IP and IPv4 in Depth ..... 235
TCP/IP Configuration Files ..... 235
`/etc/hostname.`*interface* File ..... 236
`/etc/nodename` File ..... 236
`/etc/defaultdomain` File ..... 237
`/etc/defaultrouter` File ..... 237
`hosts` Database ..... 237
`ipnodes` Database ..... 240
`netmasks` Database ..... 241
`inetd` Internet Services Daemon ..... 244
Network Databases and the `nsswitch.conf` File ..... 245
How Name Services Affect Network Databases ..... 245
`nsswitch.conf` File ..... 247
`bootparams` Database ..... 249
`ethers` Database ..... 250
Other Network Databases ..... 251

`protocols` Database ..... 252
`services` Database ..... 253
Routing Protocols in the Solaris OS ..... 253
Routing Information Protocol (RIP) ..... 254
ICMP Router Discovery (RDISC) Protocol ..... 254
Network Classes ..... 254
Class A Network Numbers ..... 254
Class B Network Numbers ..... 255
Class C Network Numbers ..... 255

**11 IPv6 in Depth (Reference) ..... 257**
What's New in IPv6 in Depth ..... 257
IPv6 Addressing Formats Beyond the Basics ..... 258
6to4-Derived Addresses ..... 258
IPv6 Multicast Addresses in Depth ..... 260
IPv6 Packet Header Format ..... 261
IPv6 Extension Headers ..... 262
Dual-Stack Protocols ..... 262
Solaris 10 IPv6 Implementation ..... 263
IPv6 Configuration Files ..... 263
IPv6-Related Commands ..... 268
IPv6-Related Daemons ..... 274
IPv6 Neighbor Discovery Protocol ..... 278
ICMP Messages From Neighbor Discovery ..... 278
Autoconfiguration Process ..... 278
Neighbor Solicitation and Unreachability ..... 280
Duplicate Address Detection Algorithm ..... 281
Proxy Advertisements ..... 281
Inbound Load Balancing ..... 281
Link-Local Address Change ..... 282
Comparison of Neighbor Discovery to ARP and Related IPv4 Protocols ..... 282
IPv6 Routing ..... 284
Router Advertisement ..... 284
IPv6 Tunnels ..... 285
Configured Tunnels ..... 287

6to4 Automatic Tunnels ..... 289
IPv6 Extensions to Solaris Name Services ..... 293
DNS Extensions for IPv6 ..... 293
Changes to the `nsswitch.conf` File ..... 293
Changes to Name Service Commands ..... 294
NFS and RPC IPv6 Support ..... 295
IPv6 Over ATM Support ..... 295

**Part III DHCP** ..... 297

**12 About Solaris DHCP (Overview)** ..... 299
About the DHCP Protocol ..... 299
Advantages of Using Solaris DHCP ..... 300
How DHCP Works ..... 301
Solaris DHCP Server ..... 304
DHCP Server Management ..... 305
DHCP Data Store ..... 305
DHCP Manager ..... 306
DHCP Command-Line Utilities ..... 307
Role-Based Access Control for DHCP Commands ..... 308
DHCP Server Configuration ..... 308
IP Address Allocation ..... 309
Network Configuration Information ..... 310
About DHCP Options ..... 310
About DHCP Macros ..... 311
Solaris DHCP Client ..... 312

**13 Planning for DHCP Service (Tasks)** ..... 315
Preparing Your Network for the DHCP Service (Task Map) ..... 315
Mapping Your Network Topology ..... 316
Determining the Number of DHCP Servers ..... 317
Updating System Files and Netmask Tables ..... 318
Making Decisions for Your DHCP Server Configuration (Task Map) ..... 319
Selecting a Host to Run the DHCP Service ..... 320

Choosing the DHCP Data Store ..... 320
Setting a Lease Policy ..... 321
Determining Routers for DHCP Clients ..... 322
Making Decisions for IP Address Management (Task Map) ..... 322
Number and Ranges of IP Addresses ..... 323
Client Host Name Generation ..... 323
Default Client Configuration Macros ..... 324
Dynamic and Permanent Lease Types ..... 325
Reserved IP Addresses and Lease Type ..... 325
Planning for Multiple DHCP Servers ..... 326
Planning DHCP Configuration of Your Remote Networks ..... 326
Selecting the Tool for Configuring DHCP ..... 327
DHCP Manager Features ..... 327
`dhcpconfig` Features ..... 327
Comparison of DHCP Manager and `dhcpconfig` ..... 328

**14 Configuring the DHCP Service (Tasks) ..... 329**
Configuring and Unconfiguring a DHCP Server Using DHCP Manager ..... 329
Configuring DHCP Servers ..... 330
▼ How to Configure a DHCP Server (DHCP Manager) ..... 332
Configuring BOOTP Relay Agents ..... 333
▼ How to Configure a BOOTP Relay Agent (DHCP Manager) ..... 333
Unconfiguring DHCP Servers and BOOTP Relay Agents ..... 334
DHCP Data on an Unconfigured Server ..... 334
▼ How to Unconfigure a DHCP Server or a BOOTP Relay Agent (DHCP Manager) ..... 335
Configuring and Unconfiguring a DHCP Server Using `dhcpconfig` Commands ..... 336
▼ How to Configure a DHCP Server (`dhcpconfig -D`) ..... 336
▼ How to Configure a BOOTP Relay Agent (`dhcpconfig -R` ) ..... 337
▼ How to Unconfigure a DHCP Server or a BOOTP Relay Agent (`dhcpconfig -U`) ..... 337

**15 Administering DHCP (Tasks) ..... 339**
About DHCP Manager ..... 340
DHCP Manager Window ..... 340
DHCP Manager Menus ..... 341
Starting and Stopping DHCP Manager ..... 342

▼ How to Start and Stop DHCP Manager ..... 342
Setting Up User Access to DHCP Commands ..... 343
▼ How to Grant Users Access to DHCP Commands ..... 343
Starting and Stopping the DHCP Service ..... 343
▼ How to Start and Stop the DHCP Service (DHCP Manager) ..... 344
▼ How to Enable and Disable the DHCP Service (DHCP Manager) ..... 345
▼ How to Enable and Disable the DHCP Service (`dhcpconfig -S`) ..... 345
DHCP Service and the Service Management Facility ..... 346
Modifying DHCP Service Options (Task Map) ..... 346
Changing DHCP Logging Options ..... 348
▼ How to Generate Verbose DHCP Log Messages (DHCP Manager) ..... 349
▼ How to Generate Verbose DHCP Log Messages (Command Line) ..... 350
▼ How to Enable and Disable DHCP Transaction Logging (DHCP Manager) ..... 350
▼ How to Enable and Disable DHCP Transaction Logging (Command Line) ..... 351
▼ How to Log DHCP Transactions to a Separate `syslog` File ..... 352
Enabling Dynamic DNS Updates by a DHCP Server ..... 352
▼ How to Enable Dynamic DNS Updating for DHCP Clients ..... 353
Client Host Name Registration ..... 355
Customizing Performance Options for the DHCP Server ..... 356
▼ How to Customize DHCP Performance Options (DHCP Manager) ..... 356
▼ How to Customize DHCP Performance Options (Command Line) ..... 357
Adding, Modifying, and Removing DHCP Networks (Task Map) ..... 358
Specifying Network Interfaces for DHCP Monitoring ..... 358
▼ How to Specify Network Interfaces for DHCP Monitoring (DHCP Manager) ..... 359
▼ How to Specify Network Interfaces for DHCP Monitoring (`dhcpconfig`) ..... 360
Adding DHCP Networks ..... 360
▼ How to Add a DHCP Network (DHCP Manager) ..... 361
▼ How to Add a DHCP Network (`dhcpconfig`) ..... 362
Modifying DHCP Network Configurations ..... 363
▼ How to Modify the Configuration of a DHCP Network (DHCP Manager) ..... 364
▼ How to Modify the Configuration of a DHCP Network (`dhtadm`) ..... 365
Removing DHCP Networks ..... 365
▼ How to Remove a DHCP Network (DHCP Manager) ..... 366
▼ How to Remove a DHCP Network (`pntadm`) ..... 367
Supporting BOOTP Clients With the DHCP Service (Task Map) ..... 367
▼ How to Set Up Support of Any BOOTP Client (DHCP Manager) ..... 368

▼ How to Set Up Support of Registered BOOTP Clients (DHCP Manager) ..... 369
Working With IP Addresses in the DHCP Service (Task Map) ..... 370
Adding IP Addresses to the DHCP Service ..... 374
▼ How to Add a Single IP Address (DHCP Manager) ..... 375
▼ How to Duplicate an Existing IP Address (DHCP Manager) ..... 376
▼ How to Add Multiple IP Addresses (DHCP Manager) ..... 376
▼ How to Add IP Addresses (pntadm) ..... 377
Modifying IP Addresses in the DHCP Service ..... 377
▼ How to Modify IP Address Properties (DHCP Manager) ..... 379
▼ How to Modify IP Address Properties (pntadm) ..... 379
Removing IP Addresses From the DHCP Service ..... 380
Marking IP Addresses as Unusable by the DHCP Service ..... 380
▼ How to Mark IP Addresses as Unusable (DHCP Manager) ..... 380
▼ How to Mark IP Addresses as Unusable (pntadm) ..... 381
Deleting IP Addresses From the DHCP Service ..... 381
▼ How to Delete IP Addresses From DHCP Service (DHCP Manager) ..... 382
▼ How to Delete IP Addresses From the DHCP Service (pntadm) ..... 383
Assigning a Reserved IP Address to a DHCP Client ..... 383
▼ How to Assign a Consistent IP Address to a DHCP Client (DHCP Manager) ..... 384
▼ How to Assign a Consistent IP Address to a DHCP Client (pntadm) ..... 385
Working With DHCP Macros (Task Map) ..... 385
▼ How to View Macros Defined on a DHCP Server (DHCP Manager) ..... 387
▼ How to View Macros Defined on a DHCP Server (dhtadm) ..... 388
Modifying DHCP Macros ..... 388
▼ How to Change Values for Options in a DHCP Macro (DHCP Manager) ..... 389
▼ How to Change Values for Options in a DHCP Macro (dhtadm) ..... 390
▼ How to Add Options to a DHCP Macro (DHCP Manager) ..... 390
▼ How to Add Options to a DHCP Macro (dhtadm) ..... 391
▼ How to Delete Options From a DHCP Macro (DHCP Manager) ..... 391
▼ How to Delete Options From a DHCP Macro (dhtadm) ..... 392
Creating DHCP Macros ..... 392
▼ How to Create a DHCP Macro (DHCP Manager) ..... 393
▼ How to Create a DHCP Macro (dhtadm) ..... 394
Deleting DHCP Macros ..... 395
▼ How to Delete a DHCP Macro (DHCP Manager) ..... 395
▼ How to Delete a DHCP Macro (dhtadm) ..... 395

Working With DHCP Options (Task Map) ..... 396
Creating DHCP Options ..... 399
▼ How to Create DHCP Options (DHCP Manager) ..... 400
▼ How to Create DHCP Options (`dhtadm`) ..... 401
Modifying DHCP Options ..... 402
▼ How to Modify DHCP Option Properties (DHCP Manager) ..... 402
▼ How to Modify DHCP Option Properties (`dhtadm`) ..... 403
Deleting DHCP Options ..... 404
▼ How to Delete DHCP Options (DHCP Manager) ..... 404
▼ How to Delete DHCP Options (`dhtadm`) ..... 404
Modifying the Solaris DHCP Client's Option Information ..... 405
Supporting Solaris Network Installation With the DHCP Service ..... 405
Supporting Remote Boot and Diskless Boot Clients (Task Map) ..... 406
Setting Up DHCP Clients to Receive Information Only (Task Map) ..... 407
Converting to a New DHCP Data Store ..... 408
▼ How to Convert the DHCP Data Store (DHCP Manager) ..... 409
▼ How to Convert the DHCP Data Store (`dhcpconfig -C`) ..... 410
Moving Configuration Data Between DHCP Servers (Task Map) ..... 410
▼ How to Export Data From a DHCP Server (DHCP Manager) ..... 412
▼ How to Export Data From a DHCP Server (`dhcpconfig -X`) ..... 413
▼ How to Import Data on a DHCP Server (DHCP Manager) ..... 414
▼ How to Import Data on a DHCP Server (`dhcpconfig -I`) ..... 414
▼ How to Modify Imported DHCP Data (DHCP Manager) ..... 415
▼ How to Modify Imported DHCP Data (`pntadm`, `dhtadm`) ..... 416

**16 Configuring and Administering the DHCP Client** ..... 417
About the Solaris DHCP Client ..... 417
DHCPv6 Server ..... 418
Differences Between DHCPv4 and DHCPv6 ..... 418
The Administrative Model ..... 418
Protocol Details ..... 419
Logical Interfaces ..... 420
Option Negotiation ..... 420
Configuration Syntax ..... 421
DHCP Client Startup ..... 421

DHCPv6 Communication ..... 422
How DHCP Client Protocols Manage Network Configuration Information ..... 423
DHCP Client Shutdown ..... 424
Enabling and Disabling a Solaris DHCP Client ..... 425
▼ How to Enable the Solaris DHCP Client ..... 425
▼ How to Disable a Solaris DHCP Client ..... 425
DHCP Client Administration ..... 426
`ifconfig` Command Options Used With the DHCP Client ..... 426
Setting DHCP Client Configuration Parameters ..... 428
DHCP Client Systems With Multiple Network Interfaces ..... 429
DHCPv4 Client Host Names ..... 430
▼ How to Enable a Solaris DHCPv4 Client to Request a Specific Host Name ..... 430
DHCP Client Systems and Name Services ..... 431
Setting Up DHCP Clients as NIS+ Clients ..... 433
▼ How to Set Up Solaris DHCP Clients as NIS+ Clients ..... 433
DHCP Client Event Scripts ..... 436

**17 Troubleshooting DHCP (Reference) ..... 441**
Troubleshooting DHCP Server Problems ..... 441
NIS+ Problems and the DHCP Data Store ..... 441
IP Address Allocation Errors in DHCP ..... 444
Troubleshooting DHCP Client Configuration Problems ..... 447
Problems Communicating With the DHCP Server ..... 447
▼ How to Run the DHCP Client in Debugging Mode ..... 448
▼ How to Run the DHCP Server in Debugging Mode ..... 448
▼ How to Use `snoop` to Monitor DHCP Network Traffic ..... 449
Problems With Inaccurate DHCP Configuration Information ..... 456
Problems With the DHCP Client-Supplied Host Name ..... 457

**18 DHCP Commands and Files (Reference) ..... 461**
DHCP Commands ..... 461
Running DHCP Commands in Scripts ..... 462
Files Used by the DHCP Service ..... 469
DHCP Option Information ..... 471
Determining if Your Site Is Affected ..... 471

Differences Between `dhcptags` and `inittab` Files ..... 472
Converting `dhcptags` Entries to `inittab` Entries ..... 473

**Part IV IP Security** ..... 475

**19 IP Security Architecture (Overview)** ..... 477
What's New in IPsec? ..... 477
Introduction to IPsec ..... 478
IPsec RFCs ..... 479
IPsec Terminology ..... 480
IPsec Packet Flow ..... 480
IPsec Security Associations ..... 482
Key Management in IPsec ..... 483
IPsec Protection Mechanisms ..... 484
Authentication Header ..... 484
Encapsulating Security Payload ..... 484
Authentication and Encryption Algorithms in IPsec ..... 485
IPsec Protection Policies ..... 486
Transport and Tunnel Modes in IPsec ..... 487
Virtual Private Networks and IPsec ..... 489
IPsec and NAT Traversal ..... 490
IPsec and SCTP ..... 491
IPsec and Solaris Zones ..... 491
IPsec Utilities and Files ..... 491
Changes to IPsec for the Solaris 10 Release ..... 492

**20 Configuring IPsec (Tasks)** ..... 495
Protecting Traffic With IPsec (Task Map) ..... 495
Protecting Traffic With IPsec ..... 496
▼ How to Secure Traffic Between Two Systems With IPsec ..... 496
▼ How to Secure a Web Server With IPsec ..... 499
▼ How to Display IPsec Policies ..... 501
▼ How to Generate Random Numbers on a Solaris System ..... 502
▼ How to Manually Create IPsec Security Associations ..... 503

▼ How to Verify That Packets Are Protected With IPsec ..... 507
▼ How to Create a Role for Configuring Network Security ..... 508
Protecting a VPN With IPsec ..... 509
Examples of Protecting a VPN With IPsec by Using Tunnels in Tunnel Mode ..... 509
Protecting a VPN With IPsec (Task Map) ..... 511
Description of the Network Topology for the IPsec Tasks to Protect a VPN ..... 512
▼ How to Protect a VPN With an IPsec Tunnel in Tunnel Mode Over IPv4 ..... 514
▼ How to Protect a VPN With an IPsec Tunnel in Tunnel Mode Over IPv6 ..... 520
▼ How to Protect a VPN With an IPsec Tunnel in Transport Mode Over IPv4 ..... 524
▼ How to Protect a VPN With an IPsec Tunnel in Transport Mode Over IPv6 ..... 530

**21 IP Security Architecture (Reference) ..... 537**
`ipsecconf` Command ..... 537
`ipsecinit.conf` File ..... 538
Sample `ipsecinit.conf` File ..... 538
Security Considerations for `ipsecinit.conf` and `ipsecconf` ..... 539
`ipsecalgs` Command ..... 540
Security Associations Database for IPsec ..... 540
Utilities for Key Generation in IPsec ..... 541
Security Considerations for `ipseckey` ..... 541
IPsec Extensions to Other Utilities ..... 542
`ifconfig` Command and IPsec ..... 542
`snoop` Command and IPsec ..... 544

**22 Internet Key Exchange (Overview) ..... 545**
What's New in IKE? ..... 545
Key Management With IKE ..... 546
IKE Key Negotiation ..... 546
IKE Key Terminology ..... 546
IKE Phase 1 Exchange ..... 547
IKE Phase 2 Exchange ..... 548
IKE Configuration Choices ..... 548
IKE With Preshared Keys ..... 548
IKE With Public Key Certificates ..... 548
IKE and Hardware Acceleration ..... 549

IKE and Hardware Storage ..... 549
IKE Utilities and Files ..... 550
Changes to IKE for the Solaris 10 Release ..... 551

**23 Configuring IKE (Tasks) ..... 553**
Configuring IKE (Task Map) ..... 553
Configuring IKE With Preshared Keys (Task Map) ..... 554
Configuring IKE With Preshared Keys ..... 554
▼ How to Configure IKE With Preshared Keys ..... 555
▼ How to Refresh IKE Preshared Keys ..... 558
▼ How to Add an IKE Preshared Key for a New Policy Entry in `ipsecinit.conf` ..... 559
▼ How to Verify That IKE Preshared Keys Are Identical ..... 563
Configuring IKE With Public Key Certificates (Task Map) ..... 564
Configuring IKE With Public Key Certificates ..... 565
▼ How to Configure IKE With Self-Signed Public Key Certificates ..... 565
▼ How to Configure IKE With Certificates Signed by a CA ..... 571
▼ How to Generate and Store Public Key Certificates on Hardware ..... 577
▼ How to Handle a Certificate Revocation List ..... 581
Configuring IKE for Mobile Systems (Task Map) ..... 583
Configuring IKE for Mobile Systems ..... 583
▼ How to Configure IKE for Off-Site Systems ..... 584
Configuring IKE to Find Attached Hardware (Task Map) ..... 591
Configuring IKE to Find Attached Hardware ..... 592
▼ How to Configure IKE to Find the Sun Crypto Accelerator 1000 Board ..... 592
▼ How to Configure IKE to Find the Sun Crypto Accelerator 4000 Board ..... 593
Changing IKE Transmission Parameters (Task Map) ..... 594
Changing IKE Transmission Parameters ..... 594
▼ How to Change the Duration of Phase 1 IKE Key Negotiation ..... 595

**24 Internet Key Exchange (Reference) ..... 597**
IKE Daemon ..... 597
IKE Policy File ..... 598
IKE Administration Command ..... 598
IKE Preshared Keys Files ..... 599
IKE Public Key Databases and Commands ..... 599

ikecert tokens Command ..... 600
ikecert certlocal Command ..... 600
ikecert certdb Command ..... 601
ikecert certrldb Command ..... 601
/etc/inet/ike/publickeys Directory ..... 602
/etc/inet/secret/ike.privatekeys Directory ..... 602
/etc/inet/ike/crls Directory ..... 602

**25 Solaris IP Filter (Overview) ..... 603**
What's New in Solaris IP Filter ..... 603
Packet Filter Hooks ..... 603
IPv6 Packet Filtering for Solaris IP Filter ..... 604
Introduction to Solaris IP Filter ..... 604
Information Sources for Open Source IP Filter ..... 604
Solaris IP Filter Packet Processing ..... 605
Guidelines for Using Solaris IP Filter ..... 608
Using Solaris IP Filter Configuration Files ..... 608
Working With Solaris IP Filter Rule Sets ..... 608
Using Solaris IP Filter's Packet Filtering Feature ..... 609
Using Solaris IP Filter's NAT Feature ..... 612
Using Solaris IP Filter's Address Pools Feature ..... 613
Packet Filter Hooks ..... 614
Solaris IP Filter and the pfil STREAMS Module ..... 615
IPv6 for Solaris IP Filter ..... 615
Solaris IP Filter Man Pages ..... 616

**26 Solaris IP Filter (Tasks) ..... 619**
Configuring Solaris IP Filter ..... 619
▼ How to Enable Solaris IP Filter ..... 620
▼ How to Re-Enable Solaris IP Filter ..... 621
▼ How to Enable Loopback Filtering ..... 622
Deactivating and Disabling Solaris IP Filter ..... 623
▼ How to Deactivate Packet Filtering ..... 623
▼ How to Deactivate NAT ..... 624
▼ How to Disable Packet Filtering ..... 625

Working With the `pfil` Module ..... 625
▼ How to Enable Solaris IP Filter in Previous Solaris 10 Releases ..... 626
▼ How to Activate a NIC for Packet Filtering ..... 628
▼ How to Deactivate Solaris IP Filter on a NIC ..... 630
▼ How to View `pfil` Statistics for Solaris IP Filter ..... 631
Working With Solaris IP Filter Rule Sets ..... 632
Managing Packet Filtering Rule Sets for Solaris IP Filter ..... 633
▼ How to View the Active Packet Filtering Rule Set ..... 633
▼ How to View the Inactive Packet Filtering Rule Set ..... 634
▼ How to Activate a Different or Updated Packet Filtering Rule Set ..... 634
▼ How to Remove a Packet Filtering Rule Set ..... 636
▼ How to Append Rules to the Active Packet Filtering Rule Set ..... 636
▼ How to Append Rules to the Inactive Packet Filtering Rule Set ..... 637
▼ How to Switch Between Active and Inactive Packet Filtering Rule Sets ..... 638
▼ How to Remove an Inactive Packet Filtering Rule Set From the Kernel ..... 639
Managing NAT Rules for Solaris IP Filter ..... 640
▼ How to View Active NAT Rules ..... 640
▼ How to Remove NAT Rules ..... 640
▼ How to Append Rules to the NAT Rules ..... 641
Managing Address Pools for Solaris IP Filter ..... 642
▼ How to View Active Address Pools ..... 642
▼ How to Remove an Address Pool ..... 642
▼ How to Append Rules to an Address Pool ..... 643
Displaying Statistics and Information for Solaris IP Filter ..... 644
▼ How to View State Tables for Solaris IP Filter ..... 644
▼ How to View State Statistics for Solaris IP Filter ..... 645
▼ How to View NAT Statistics for Solaris IP Filter ..... 646
▼ How to View Address Pool Statistics for Solaris IP Filter ..... 646
Working With Log Files for Solaris IP Filter ..... 647
▼ How to Set Up a Log File for Solaris IP Filter ..... 647
▼ How to View Solaris IP Filter Log Files ..... 648
▼ How to Flush the Packet Log File ..... 649
▼ How to Save Logged Packets to a File ..... 650
Creating and Editing Solaris IP Filter Configuration Files ..... 651
▼ How to Create a Configuration File for Solaris IP Filter ..... 651
Solaris IP Filter Configuration File Examples ..... 652

**Part V Mobile IP** ..... 659

**27 Mobile IP (Overview)** ..... 661
What's New in Mobile IP ..... 661
Introduction to Mobile IP ..... 662
Mobile IP Functional Entities ..... 663
How Mobile IP Works ..... 664
Agent Discovery ..... 666
Agent Advertisement ..... 666
Agent Solicitation ..... 667
Care-of Addresses ..... 667
Mobile IP With Reverse Tunneling ..... 668
Limited Private Addresses Support ..... 668
Mobile IP Registration ..... 670
Network Access Identifier (NAI) ..... 672
Mobile IP Message Authentication ..... 672
Mobile Node Registration Request ..... 672
Registration Reply Message ..... 673
Foreign Agent Considerations ..... 673
Home Agent Considerations ..... 673
Dynamic Home Agent Discovery ..... 674
Routing Datagrams to and From Mobile Nodes ..... 674
Encapsulation Methods ..... 674
Unicast Datagram Routing ..... 674
Broadcast Datagrams ..... 675
Multicast Datagram Routing ..... 675
Security Considerations for Mobile IP ..... 676
Use of IPsec With Mobile IP ..... 677

**28 Administering Mobile IP (Tasks)** ..... 679
Creating the Mobile IP Configuration File (Task Map) ..... 679
Creating the Mobile IP Configuration File ..... 680
▼ How to Plan for Mobile IP ..... 680
▼ How to Create the Mobile IP Configuration File ..... 681
▼ How to Configure the `General` Section ..... 681

▼ How to Configure the `Advertisements` Section ..... 682
▼ How to Configure the `GlobalSecurityParameters` Section ..... 682
▼ How to Configure the `Pool` Section ..... 683
▼ How to Configure the `SPI` Section ..... 683
▼ How to Configure the `Address` Section ..... 683
Modifying the Mobile IP Configuration File (Task Map) ..... 684
Modifying the Mobile IP Configuration File ..... 685
▼ How to Modify the `General` Section ..... 685
▼ How to Modify the `Advertisements` Section ..... 686
▼ How to Modify the `GlobalSecurityParameters` Section ..... 687
▼ How to Modify the `Pool` Section ..... 687
▼ How to Modify the `SPI` Section ..... 688
▼ How to Modify the `Address` Section ..... 688
▼ How to Add or Delete Configuration File Parameters ..... 689
▼ How to Display Current Parameter Values in the Configuration File ..... 691
Displaying Mobility Agent Status ..... 692
▼ How to Display Mobility Agent Status ..... 692
Displaying Mobility Routes on a Foreign Agent ..... 694
▼ How to Display Mobility Routes on a Foreign Agent ..... 694

**29 Mobile IP Files and Commands (Reference)** ..... 695
Overview of the Solaris Mobile IP Implementation ..... 695
Mobile IP Configuration File ..... 696
Configuration File Format ..... 696
Sample Configuration Files ..... 697
Configuration File Sections and Labels ..... 701
Configuring the Mobility IP Agent ..... 710
Mobile IP Mobility Agent Status ..... 711
Mobile IP State Information ..... 712
`netstat` Extensions for Mobile IP ..... 712
`snoop` Extensions for Mobile IP ..... 712

**Part VI IPMP** ..... 715

**30 Introducing IPMP (Overview)** ..... 717
Why You Should Use IPMP ..... 717
Solaris IPMP Components ..... 718
IPMP Terminology and Concepts ..... 718
Basic Requirements of IPMP ..... 721
IPMP Addressing ..... 721
Data Addresses ..... 721
Test Addresses ..... 722
Preventing Applications From Using Test Addresses ..... 723
IPMP Interface Configurations ..... 724
Standby Interfaces in an IPMP Group ..... 724
Common IPMP Interface Configurations ..... 724
IPMP Failure Detection and Recovery Features ..... 725
Link-Based Failure Detection ..... 725
Probe-Based Failure Detection ..... 726
Group Failures ..... 727
Detecting Physical Interface Repairs ..... 727
What Happens During Interface Failover ..... 727
IPMP and Dynamic Reconfiguration ..... 729
Attaching NICs ..... 729
Detaching NICs ..... 730
Reattaching NICs ..... 730
NICs That Were Missing at System Boot ..... 731

**31 Administering IPMP (Tasks)** ..... 733
Configuring IPMP (Task Maps) ..... 733
Configuring and Administering IPMP Groups (Task Map) ..... 733
Administering IPMP on Interfaces That Support Dynamic Reconfiguration (Task Map) ..... 734
Configuring IPMP Groups ..... 735
Planning for an IPMP Group ..... 735
▼ How to Plan for an IPMP Group ..... 735
Configuring IPMP Groups ..... 737

▼ How to Configure an IPMP Group With Multiple Interfaces ..... 737
Configuring IPMP Groups With a Single Physical Interface ..... 745
▼ How to Configure a Single Interface IPMP Group ..... 745
Maintaining IPMP Groups ..... 746
▼ How to Display the IPMP Group Membership of an Interface ..... 747
▼ How to Add an Interface to an IPMP Group ..... 747
▼ How to Remove an Interface From an IPMP Group ..... 748
▼ How to Move an Interface From One IPMP Group to Another Group ..... 749
Replacing a Failed Physical Interface on Systems That Support Dynamic Reconfiguration ... 750
▼ How to Remove a Physical Interface That Has Failed (DR-Detach) ..... 750
▼ How to Replace a Physical Interface That Has Failed (DR-Attach) ..... 751
Recovering a Physical Interface That Was Not Present at System Boot ..... 752
▼ How to Recover a Physical Interface That Was Not Present at System Boot ..... 752
Modifying IPMP Configurations ..... 754
▼ How to Configure the `/etc/default/mpathd` File ..... 754

**Part VII IP Quality of Service (IPQoS)** ..... 757

**32 Introducing IPQoS (Overview)** ..... 759
IPQoS Basics ..... 759
What Are Differentiated Services? ..... 759
IPQoS Features ..... 760
Where to Get More Information About Quality-of-Service Theory and Practice ..... 760
Providing Quality of Service With IPQoS ..... 762
Implementing Service-Level Agreements ..... 762
Assuring Quality of Service for an Individual Organization ..... 762
Introducing the Quality-of-Service Policy ..... 762
Improving Network Efficiency With IPQoS ..... 763
How Bandwidth Affects Network Traffic ..... 763
Using Classes of Service to Prioritize Traffic ..... 764
Differentiated Services Model ..... 764
Classifier (`ipgpc`) Overview ..... 765
Meter (`tokenmt` and `tswtclmt`) Overview ..... 766
Marker (`dscpmk` and `dlcosmk`) Overview ..... 766
Flow Accounting (`flowacct`) Overview ..... 767

How Traffic Flows Through the IPQoS Modules ..... 767
Traffic Forwarding on an IPQoS-Enabled Network ..... 769
DS Codepoint ..... 769
Per-Hop Behaviors ..... 769

**33 Planning for an IPQoS-Enabled Network (Tasks)** ..... 773
General IPQoS Configuration Planning (Task Map) ..... 773
Planning the Diffserv Network Topology ..... 774
Hardware Strategies for the Diffserv Network ..... 774
IPQoS Network Topologies ..... 774
Planning the Quality-of-Service Policy ..... 777
QoS Policy Planning Aids ..... 777
QoS Policy Planning (Task Map) ..... 778
▼ How to Prepare a Network for IPQoS ..... 779
▼ How to Define the Classes for Your QoS Policy ..... 780
Defining Filters ..... 782
▼ How to Define Filters in the QoS Policy ..... 783
▼ How to Plan Flow Control ..... 784
▼ How to Plan Forwarding Behavior ..... 786
▼ How to Plan for Flow Accounting ..... 789
Introducing the IPQoS Configuration Example ..... 789
IPQoS Topology ..... 790

**34 Creating the IPQoS Configuration File (Tasks)** ..... 793
Defining a QoS Policy in the IPQoS Configuration File (Task Map) ..... 793
Tools for Creating a QoS Policy ..... 794
Basic IPQoS Configuration File ..... 795
Creating IPQoS Configuration Files for Web Servers ..... 795
▼ How to Create the IPQoS Configuration File and Define Traffic Classes ..... 798
▼ How to Define Filters in the IPQoS Configuration File ..... 800
▼ How to Define Traffic Forwarding in the IPQoS Configuration File ..... 801
▼ How to Enable Accounting for a Class in the IPQoS Configuration File ..... 804
▼ How to Create an IPQoS Configuration File for a Best-Effort Web Server ..... 806
Creating an IPQoS Configuration File for an Application Server ..... 808
▼ How to Configure the IPQoS Configuration File for an Application Server ..... 811

▼ How to Configure Forwarding for Application Traffic in the IPQoS Configuration File . 813
▼ How to Configure Flow Control in the IPQoS Configuration File ..... 815
Providing Differentiated Services on a Router ..... 818
▼ How to Configure a Router on an IPQoS-Enabled Network ..... 819

**35 Starting and Maintaining IPQoS (Tasks)** ..... 821
Administering IPQoS (Task Map) ..... 821
Applying an IPQoS Configuration ..... 822
▼ How to Apply a New Configuration to the IPQoS Kernel Modules ..... 822
▼ How to Ensure That the IPQoS Configuration Is Applied After Each Reboot ..... 823
Enabling `syslog` Logging for IPQoS Messages ..... 823
▼ How to Enable Logging of IPQoS Messages During Booting ..... 823
Troubleshooting with IPQoS Error Messages ..... 824

**36 Using Flow Accounting and Statistics Gathering (Tasks)** ..... 829
Setting Up Flow Accounting (Task Map) ..... 829
Recording Information About Traffic Flows ..... 829
▼ How to Create a File for Flow-Accounting Data ..... 830
Gathering Statistical Information ..... 832

**37 IPQoS in Detail (Reference)** ..... 835
IPQoS Architecture and the Diffserv Model ..... 835
Classifier Module ..... 835
Meter Module ..... 838
Marker Module ..... 840
`flowacct` Module ..... 845
IPQoS Configuration File ..... 847
`action` Statement ..... 849
Module Definitions ..... 849
`class` Clause ..... 850
`filter` Clause ..... 850
`params` Clause ..... 851
`ipqosconf` Configuration Utility ..... 851

**Glossary** ..... 853

**Index** ..... 863

# Preface

*System Administration Guide, IP Services* is part of a nine-volume set that covers a significant part of Solaris™ system administration information. This book assumes that you have already installed the Solaris 10 operating system (Solaris OS). You should be ready to configure your network or ready to configure any networking software that is required on your network. The Solaris OS 10 is part of the Solaris product family, which also includes the Solaris Common Desktop Environment (CDE). Solaris OS is compliant with AT&T's System V, Release 4 operating system.

---

**Note –** This Solaris release supports systems that use the SPARC® and x86 families of processor architectures: UltraSPARC®, SPARC64, AMD64, Pentium, and Xeon EM64T. The supported systems appear in the *Solaris 10 Hardware Compatibility List* at `http://www.sun.com/bigadmin/hcl`. This document cites any implementation differences between the platform types.

In this document these x86 related terms mean the following:

- "x86" refers to the larger family of 64-bit and 32-bit x86 compatible products.
- "x64" points out specific 64-bit information about AMD64 or EM64T systems.
- "32-bit x86" points out specific 32-bit information about x86 based systems.

For supported systems, see the *Solaris 10 Hardware Compatibility List*.

---

## Who Should Use This Book

This book is intended for anyone responsible for administering systems that run the Solaris OS release, which are configured in a network. To use this book, you should have at least two years of UNIX® system administration experience. Attending UNIX system administration training courses might be helpful.

# How the System Administration Volumes Are Organized

Here is a list of the topics that are covered by the volumes of the System Administration Guides.

| Book Title | Topics |
|---|---|
| *System Administration Guide: Basic Administration* | User accounts and groups, server and client support, shutting down and booting a system, managing services, and managing software (packages and patches) |
| *System Administration Guide: Advanced Administration* | Terminals and modems, system resources (disk quotas, accounting, and crontabs), system processes, and troubleshooting Solaris software problems |
| *System Administration Guide: Devices and File Systems* | Removable media, disks and devices, file systems, and backing up and restoring data |
| *System Administration Guide: IP Services* | TCP/IP network administration, IPv4 and IPv6 address administration, DHCP, IPsec, IKE, Solaris IP filter, Mobile IP, IP network multipathing (IPMP), and IPQoS |
| *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* | DNS, NIS, and LDAP naming and directory services, including transitioning from NIS to LDAP and transitioning from NIS+ to LDAP |
| *System Administration Guide: Naming and Directory Services (NIS+)* | NIS+ naming and directory services |
| *System Administration Guide: Network Services* | Web cache servers, time-related services, network file systems (NFS and Autofs), mail, SLP, and PPP |
| *System Administration Guide: Security Services* | Auditing, device management, file security, BART, Kerberos services, PAM, Solaris Cryptographic Framework, privileges, RBAC, SASL, and Solaris Secure Shell |
| *System Administration Guide: Virtualization Using the Solaris Operating System* | Resource management topics projects and tasks, extended accounting, resource controls, fair share scheduler (FSS), physical memory control using the resource capping daemon (rcapd), and resource pools; virtualization using Solaris Zones software partitioning technology and `lx` branded zones |
| *ZFS Administration Guide* | ZFS storage pool and file system creation and management, snapshots, clones, backups, using access control lists (ACLs) to protect ZFS files, using ZFS on a Solaris system with zones installed, emulated volumes, and troubleshooting and data recovery |
| *Solaris Trusted Extensions Administrator's Procedures* | System administration that is specific to a Solaris Trusted Extensions system |

| Book Title | Topics |
| --- | --- |
| *Solaris Trusted Extensions Configuration Guide* | Starting with the S10U5 release, describes how to plan for, enable, and initially configure Solaris Trusted Extensions |
| *System Administration Guide: Solaris Printing* | Solaris printing topics and tasks, using services, tools, protocols, and technologies to set up and administer printing services and printers |

# Related Books

The following trade books are referred to in this book.

- Stevens, W. Richard. *TCP/IP Illustrated, Volume 1, The Protocols*. Addison Wesley, 1994.
- Hunt Craig.*TCP/IP Network Administration, 3rd Edition*. O'Reilly, 2002.
- Perkins, Charles E. *Mobile IP Design Principles and Practices*. Massachusetts, 1998, Addison-Wesley Publishing Company.
- Solomon, James D. *Mobile IP: The Internet Unplugged*. New Jersey, 1998, Prentice-Hall, Inc.
- Ferguson, Paul and Geoff Huston. *Quality of Service*. John Wiley & Sons, Inc., 1998.
- Kilkki, Kalevi. *Differentiated Services for the Internet*. Macmillan Technical Publishing, 1999.

# Related Third-Party Web Site References

Third party URLs are referenced in this document and provide additional, related information.

---

**Note –** Sun is not responsible for the availability of third-party Web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

---

Solaris IP Filter is derived from open source IP Filter software. To view license terms, attribution, and copyright statements for IP Filter, the default path is `/usr/lib/ipf/IPFILTER.LICENCE`. If Solaris OS has been installed anywhere other than the default, modify the given path to access the file at the installed location.

# Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (`http://www.sun.com/documentation/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://www.sun.com/training/`)

# Typographic Conventions

The following table describes the typographic conventions that are used in this book.

**TABLE P–1** Typographic Conventions

| Typeface | Meaning | Example |
|---|---|---|
| `AaBbCc123` | The names of commands, files, and directories, and onscreen computer output | Edit your `.login` file.<br>Use `ls -a` to list all files.<br>`machine_name% you have mail.` |
| **`AaBbCc123`** | What you type, contrasted with onscreen computer output | `machine_name%` **`su`**<br>`Password:` |
| *aabbcc123* | Placeholder: replace with a real name or value | The command to remove a file is `rm` *filename*. |
| *AaBbCc123* | Book titles, new terms, and terms to be emphasized | Read Chapter 6 in the *User's Guide*.<br>A *cache* is a copy that is stored locally.<br>Do *not* save the file.<br>**Note:** Some emphasized items appear bold online. |

# Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

**TABLE P–2** Shell Prompts

| Shell | Prompt |
|---|---|
| C shell | `machine_name%` |
| C shell for superuser | `machine_name#` |
| Bourne shell and Korn shell | `$` |
| Bourne shell and Korn shell for superuser | `#` |

PART I

# Introducing System Administration: IP Services

This part contains introductory information about the TCP/IP protocol suite and its implementation in the Solaris Operating System (Solaris OS).

◆ ◆ ◆ CHAPTER 1

# Solaris TCP/IP Protocol Suite (Overview)

This chapter introduces the Solaris implementation of the TCP/IP network protocol suite. The information is intended for system and network administrators who are unfamiliar with basic TCP/IP concepts. The remaining parts of this book assume that you are familiar with these concepts.

This chapter contains the following information:


- "Introducing the TCP/IP Protocol Suite" on page 37
- "How the TCP/IP Protocols Handle Data Communications" on page 44
- "Finding Out More About TCP/IP and the Internet" on page 48


## What's New in This Release

Starting with the Solaris Express Developer Edition 9/07, the Mobile IP feature is no longer available in the Solaris Express Developer's releases. See the Solaris Express Developer Edition 9/07 Release Notes for more information. Mobile IP is available in the Solaris 10 OS 8/07 release.

## Introducing the TCP/IP Protocol Suite

This section presents an in-depth introduction to the protocols that are included in TCP/IP. Although the information is conceptual, you should learn the names of the protocols. You should also learn what each protocol does.

"TCP/IP" is the acronym that is commonly used for the set of network protocols that compose the *Internet Protocol suite*. Many texts use the term "Internet" to describe both the protocol suite and the global wide area network. In this book, "TCP/IP" refers specifically to the Internet protocol suite. "Internet" refers to the wide area network and the bodies that govern the Internet.

To interconnect your TCP/IP network with other networks, you must obtain a unique IP address for your network. At the time of this writing, you obtain this address from an Internet service provider (ISP).

If hosts on your network are to participate in the Internet Domain Name System (DNS), you must obtain and register a unique domain name. The InterNIC coordinates the registration of domain names through a group of worldwide registries. For more information on DNS, refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

# Protocol Layers and the Open Systems Interconnection Model

Most network protocol suites are structured as a series of layers, sometimes collectively referred to as a *protocol stack*. Each layer is designed for a specific purpose. Each layer exists on both the sending and receiving systems. A specific layer on one system sends or receives exactly the same object that another system's *peer process* sends or receives. These activities occur independently from activities in layers above or below the layer under consideration. In essence, each layer on a system acts independently of other layers on the same system. Each layer acts in parallel with the same layer on other systems.

## OSI Reference Model

Most network protocol suites are structured in layers. The International Organization for Standardization (ISO) designed the Open Systems Interconnection (OSI) Reference Model that uses structured layers. The OSI model describes a structure with seven layers for network activities. One or more protocols is associated with each layer. The layers represent data transfer operations that are common to all types of data transfers among cooperating networks.

The OSI model lists the protocol layers from the top (layer 7) to the bottom (layer 1). The following table shows the model.

**TABLE 1–1** Open Systems Interconnection Reference Model

| Layer No. | Layer Name | Description |
|---|---|---|
| 7 | Application | Consists of standard communication services and applications that everyone can use. |
| 6 | Presentation | Ensures that information is delivered to the receiving system in a form that the system can understand. |
| 5 | Session | Manages the connections and terminations between cooperating systems. |

**TABLE 1–1** Open Systems Interconnection Reference Model *(Continued)*

| Layer No. | Layer Name | Description |
|---|---|---|
| 4 | Transport | Manages the transfer of data. Also assures that the received data are identical to the transmitted data. |
| 3 | Network | Manages data addressing and delivery between networks. |
| 2 | Data link | Handles the transfer of data across the network media. |
| 1 | Physical | Defines the characteristics of the network hardware. |

The OSI model defines conceptual operations that are not unique to any particular network protocol suite. For example, the OSI network protocol suite implements all seven layers of the OSI model. TCP/IP uses some of OSI model layers. TCP/IP also combines other layers. Other network protocols, such as SNA, add an eighth layer.

# TCP/IP Protocol Architecture Model

The OSI model describes idealized network communications with a family of protocols. TCP/IP does not directly correspond to this model. TCP/IP either combines several OSI layers into a single layer, or does not use certain layers at all. The following table shows the layers of the Solaris implementation of TCP/IP. The table lists the layers from the topmost layer (application) to the bottommost layer (physical network).

**TABLE 1–2** TCP/IP Protocol Stack

| OSI Ref. Layer No. | OSI Layer Equivalent | TCP/IP Layer | TCP/IP Protocol Examples |
|---|---|---|---|
| 5,6,7 | Application, session, presentation | Application | NFS, NIS, DNS, LDAP, `telnet`, `ftp`, `rlogin`, `rsh`, `rcp`, RIP, RDISC, SNMP, and others |
| 4 | Transport | Transport | TCP, UDP, SCTP |
| 3 | Network | Internet | IPv4, IPv6, ARP, ICMP |
| 2 | Data link | Data link | PPP, IEEE 802.2 |
| 1 | Physical | Physical network | Ethernet (IEEE 802.3), Token Ring, RS-232, FDDI, and others |

The table shows the TCP/IP protocol layers and the OSI model equivalents. Also shown are examples of the protocols that are available at each level of the TCP/IP protocol stack. Each system that is involved in a communication transaction runs a unique implementation of the protocol stack.

## Physical Network Layer

The *physical network layer* specifies the characteristics of the hardware to be used for the network. For example, physical network layer specifies the physical characteristics of the communications media. The physical layer of TCP/IP describes hardware standards such as IEEE 802.3, the specification for Ethernet network media, and RS-232, the specification for standard pin connectors.

## Data-Link Layer

The *data-link layer* identifies the network protocol type of the packet, in this instance TCP/IP. The data-link layer also provides error control and "framing." Examples of data-link layer protocols are Ethernet IEEE 802.2 framing and Point-to-Point Protocol (PPP) framing.

## Internet Layer

The Internet layer, also known as the *network layer* or *IP layer*, accepts and delivers packets for the network. This layer includes the powerful Internet Protocol (IP), the Address Resolution Protocol (ARP), and the Internet Control Message Protocol (ICMP).

### IP Protocol

The IP protocol and its associated routing protocols are possibly the most significant of the entire TCP/IP suite. IP is responsible for the following:

- **IP addressing** – The IP addressing conventions are part of the IP protocol. "Designing an IPv4 Addressing Scheme" on page 58 introduces IPv4 addressing and "IPv6 Addressing Overview" on page 74 introduces IPv6 addressing.
- **Host-to-host communications** – IP determines the path a packet must take, based on the receiving system's IP address.
- **Packet formatting** – IP assembles packets into units that are known as *datagrams*. Datagrams are fully described in "Internet Layer: Where Packets Are Prepared for Delivery" on page 47.
- **Fragmentation** – If a packet is too large for transmission over the network media, IP on the sending system breaks the packet into smaller fragments. IP on the receiving system then reconstructs the fragments into the original packet.

The Solaris OS supports both IPv4 and IPv6 addressing formats, which are described in this book. To avoid confusion when addressing the Internet Protocol, one of the following conventions is used:

- When the term "IP" is used in a description, the description applies to both IPv4 and IPv6.
- When the term "IPv4" is used in a description, the description applies only to IPv4.
- When the term "IPv6" is used in a description, the description applies only to IPv6.

### ARP Protocol

The Address Resolution Protocol (ARP) conceptually exists between the data-link and Internet layers. ARP assists IP in directing datagrams to the appropriate receiving system by mapping Ethernet addresses (48 bits long) to known IP addresses (32 bits long).

### ICMP Protocol

The Internet Control Message Protocol (ICMP) detects and reports network error conditions. ICMP reports on the following:

- **Dropped packets** – Packets that arrive too fast to be processed
- **Connectivity failure** – A destination system cannot be reached
- **Redirection** – Redirecting a sending system to use another router

Chapter 8, "Administering a TCP/IP Network (Tasks)," contains more information on the Solaris OS commands that use ICMP for error detection.

## Transport Layer

The TCP/IP *transport layer* ensures that packets arrive in sequence and without error, by swapping acknowledgments of data reception, and retransmitting lost packets. This type of communication is known as *end-to-end*. Transport layer protocols at this level are Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Stream Control Transmission Protocol (SCTP). TCP and SCTP provide reliable, end-to-end service. UDP provides unreliable datagram service.

### TCP Protocol

TCP enables applications to communicate with each other as though they were connected by a physical circuit. TCP sends data in a form that appears to be transmitted in a character-by-character fashion, rather than as discrete packets. This transmission consists of the following:

- Starting point, which opens the connection
- Entire transmission in byte order
- Ending point, which closes the connection.

TCP attaches a header onto the transmitted data. This header contains many parameters that help processes on the sending system connect to peer processes on the receiving system.

TCP confirms that a packet has reached its destination by establishing an end-to-end connection between sending and receiving hosts. TCP is therefore considered a "reliable, connection-oriented" protocol.

### SCTP Protocol

SCTP is a reliable, connection-oriented transport layer protocol that provides the same services to applications that are available from TCP. Moreover, SCTP can support connections between systems that have more than one address, or *multihomed*. The SCTP connection between sending and receiving system is called an *association*. Data in the association is organized in chunks. Because SCTP supports multihoming, certain applications, particularly applications used by the telecommunications industry, need to run over SCTP, rather than TCP.

### UDP Protocol

UDP provides datagram delivery service. UDP does not verify connections between receiving and sending hosts. Because UDP eliminates the processes of establishing and verifying connections, applications that send small amounts of data use UDP.

## Application Layer

The *application layer* defines standard Internet services and network applications that anyone can use. These services work with the transport layer to send and receive data. Many application layer protocols exist. The following list shows examples of application layer protocols:

- Standard TCP/IP services such as the `ftp`, `tftp`, and `telnet` commands
- UNIX "r" commands, such as `rlogin` and `rsh`
- Name services, such as NIS and the domain name system (DNS)
- Directory services (LDAP)
- File services, such as the NFS service
- Simple Network Management Protocol (SNMP), which enables network management
- Router Discovery Server protocol (RDISC) and Routing Information Protocol (RIP) routing protocols

### Standard TCP/IP Services

- **FTP and Anonymous FTP** – The File Transfer Protocol (FTP) transfers files to and from a remote network. The protocol includes the `ftp` command and the `in.ftpd` daemon. FTP enables a user to specify the name of the remote host and file transfer command options on the local host's command line. The `in.ftpd` daemon on the remote host then handles the requests from the local host. Unlike `rcp`, `ftp` works even when the remote computer does not run a UNIX based operating system. A user must log in to the remote system to make an `ftp` connection, unless the remote system has been configured to allow anonymous FTP.

  You can obtain an enormous amount of material from *anonymous FTP servers* that are connected to the Internet. Universities and other institutions set up these servers to offer software, research papers, and other information to the public domain. When you log in to this type of server, you use the login name `anonymous`, hence the term "anonymous FTP server."

Using anonymous FTP and setting up anonymous FTP servers is outside the scope of this manual. However, many books, such as *The Whole Internet User's Guide & Catalog*, discuss anonymous FTP in detail. Instructions for using FTP are in *System Administration Guide: Network Services*. The `ftp`(1) man page describes all `ftp` command options that are invoked through the command interpreter. The `ftpd`(1M) man page describes the services that are provided by the `in.ftpd` daemon.

- **Telnet** – The Telnet protocol enables terminals and terminal-oriented processes to communicate on a network that runs TCP/IP. This protocol is implemented as the `telnet` program on local systems and the `in.telnetd` daemon on remote machines. Telnet provides a user interface through which two hosts can communicate on a character-by-character or line-by-line basis. Telnet includes a set of commands that are fully documented in the `telnet`(1) man page.
- **TFTP** – The Trivial File Transfer Protocol (`tftp`) provides functions that are similar to `ftp`, but the protocol does not establish `ftp`'s interactive connection. As a result, users cannot list the contents of a directory or change directories. A user must know the full name of the file to be copied. The `tftp`(1)man page describes the `tftp` command set.

## UNIX "r" Commands

The UNIX "r" commands enable users to issue commands on their local machines that run on the remote host. These commands include the following:

- `rcp`
- `rlogin`
- `rsh`

Instructions for using these commands are in the `rcp`(1), `rlogin`(1), and `rsh`(1) man pages.

## Name Services

The Solaris OS provides the following name services:

- **DNS** – The domain name system (DNS) is the name service provided by the Internet for TCP/IP networks. DNS provides host names to the IP address service. DNS also serves as a database for mail administration. For a complete description of this service, see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*. See also the `resolver`(3RESOLV) man page.
- `/etc` **files** – The original host-based UNIX name system was developed for standalone UNIX machines and then adapted for network use. Many old UNIX operating systems and computers still use this system, but it is not well suited for large complex networks.
- **NIS** – Network Information Service (NIS) was developed independently of DNS and has a slightly different focus. Whereas DNS focuses on making communication simpler by using machine 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 machine names and addresses, users, the network itself, and network services. NIS name space information is stored in NIS maps. For more information on NIS Architecture and NIS Administration, see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

### Directory Service

The Solaris OS supports LDAP (Lightweight Directory Access Protocol) in conjunction with the Sun Open Net Environment (Sun ONE) Directory Server, as well as other LDAP directory servers. The distinction between a name service and a directory service is in the differing extent of functionality. A directory service provides the same functionality of a naming service, but provides additional functionalities as well. See *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

### File Services

The NFS application layer protocol provides file services for the Solaris OS. You can find complete information about the NFS service in *System Administration Guide: Network Services*.

### Network Administration

The Simple Network Management Protocol (SNMP) enables you to view the layout of your network and the status of key machines. SNMP also enables you to obtain complex network statistics from software that is based on a graphical user interface (GUI). Many companies offer network management packages that implement SNMP.

### Routing Protocols

The Routing Information Protocol (RIP) and the Router Discovery Server Protocol (RDISC) are two available routing protocols for TCP/IP networks. For complete lists of available routing protocols for the Solaris 10 OS, refer to Table 5–1 and Table 5–2.

## How the TCP/IP Protocols Handle Data Communications

When a user issues a command that uses a TCP/IP application layer protocol, a series of events is initiated. The user's command or message passes through the TCP/IP protocol stack on the local system. Then, the command or message passes across the network media to the protocols on the remote system. The protocols at each layer on the sending host add information to the original data.

Protocols on each layer of the sending host also interact with their peers on the receiving host. Figure 1–1 shows this interaction.

# Data Encapsulation and the TCP/IP Protocol Stack

The packet is the basic unit of information that is transferred across a network. The basic packet consists of a header with the sending and receiving systems' addresses, and a body, or *payload*, with the data to be transferred. As the packet travels through the TCP/IP protocol stack, the protocols at each layer either add or remove fields from the basic header. When a protocol on the sending system adds data to the packet header, the process is called *data encapsulation*. Moreover, each layer has a different term for the altered packet, as shown in the following figure.

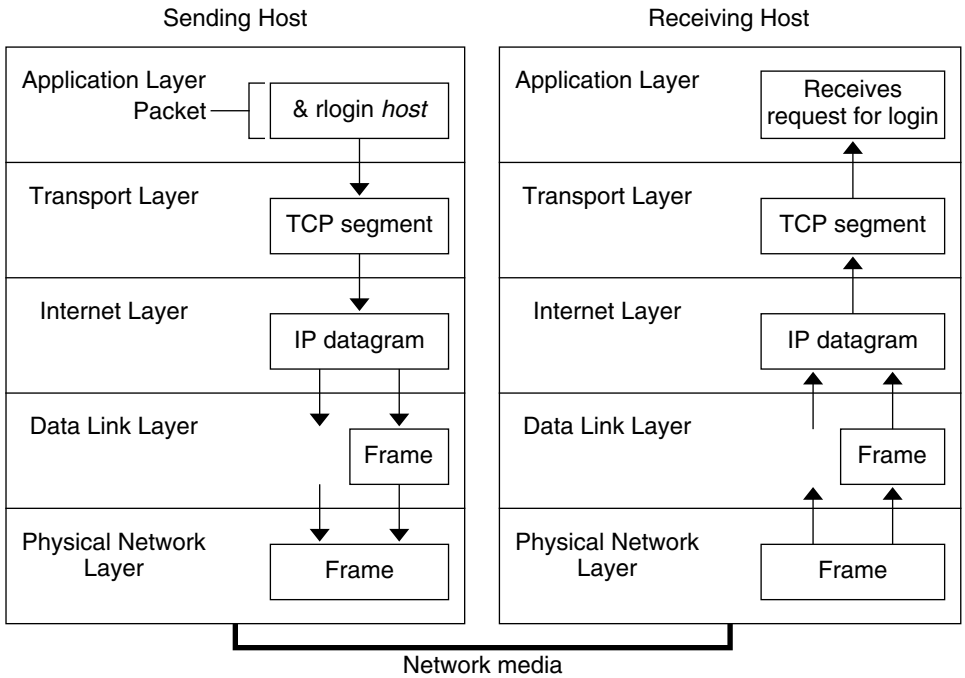



**FIGURE 1–1** How a Packet Travels Through the TCP/IP Stack

This section summarizes the life cycle of a packet. The life cycle starts when you issue a command or send a message. The life cycle finishes when the appropriate application on the receiving system receives the packet.

## Application Layer: Where a Communication Originates

The packet's history begins when a user on one system sends a message or issues a command that must access a remote system. The application protocol formats the packet so that the appropriate transport layer protocol, TCP or UDP, can handle the packet.

Suppose the user issues an `rlogin` command to log in to the remote system, as shown in Figure 1–1. The `rlogin` command uses the TCP transport layer protocol. TCP expects to receive data in the form of a stream of bytes that contain the information in the command. Therefore, `rlogin` sends this data as a TCP stream.

## Transport Layer: Where Data Encapsulation Begins

When the data arrives at the transport layer, the protocols at the layer start the process of data encapsulation. The transport layer encapsulates the application data into transport protocol data units.

The transport layer protocol creates a virtual flow of data between the sending and receiving application, differentiated by the transport port number. The port number identifies a *port*, a dedicated location in memory for receiving or sending data. In addition, the transport protocol layer might provide other services, such as reliable, in order data delivery. The end result depends on whether TCP, SCTP, or UDP handles the information.

### TCP Segmentation

TCP is often called a "connection-oriented" protocol because TCP ensures the successful delivery of data to the receiving host. Figure 1–1 shows how the TCP protocol receives the stream from the `rlogin` command. TCP then divides the data that is received from the application layer into segments and attaches a header to each segment.

Segment headers contain sending and receiving ports, segment ordering information, and a data field that is known as a *checksum*. The TCP protocols on both hosts use the checksum data to determine if the data transfers without error.

### Establishing a TCP Connection

TCP uses segments to determine whether the receiving system is ready to receive the data. When the sending TCP wants to establish connections, TCP sends a segment that is called a *SYN* to the TCP protocol on the receiving host. The receiving TCP returns a segment that is called an *ACK* to acknowledge the successful receipt of the segment. The sending TCP sends another ACK segment, then proceeds to send the data. This exchange of control information is referred to as a *three-way handshake*.

### UDP Packets

UDP is a "connectionless" protocol. Unlike TCP, UDP does not check that data arrived at the receiving host. Instead, UDP formats the message that is received from the application layer into *UDP packets*. UDP attaches a header to each packet. The header contains the sending and receiving ports, a field with the length of the packet, and a checksum.

The sending UDP process attempts to send the packet to its peer UDP process on the receiving host. The application layer determines whether the receiving UDP process acknowledges the reception of the packet. UDP requires no notification of receipt. UDP does not use the three-way handshake.

## Internet Layer: Where Packets Are Prepared for Delivery

The transport protocols TCP, UDP, and SCTP pass their segments and packets down to the Internet layer, where the IP protocol handles the segments and packets. IP prepares them for delivery by formatting them into units called *IP datagrams*. IP then determines the IP addresses for the datagrams, so that they can be delivered effectively to the receiving host.

### IP Datagrams

IP attaches an *IP header* to the segment or packet's header, in addition to the information that is added by TCP or UDP. Information in the IP header includes the IP addresses of the sending and receiving hosts, the datagram length, and the datagram sequence order. This information is provided if the datagram exceeds the allowable byte size for network packets and must be fragmented.

## Data-Link Layer: Where Framing Takes Place

Data-link layer protocols, such as PPP, format the IP datagram into a *frame*. These protocols attach a third header and a footer to "frame" the datagram. The frame header includes a *cyclic redundancy check* (CRC) field that checks for errors as the frame travels over the network media. Then, the data-link layer passes the frame to the physical layer.

## Physical Network Layer: Where Frames Are Sent and Received

The physical network layer on the sending host receives the frames and converts the IP addresses into the hardware addresses appropriate to the network media. The physical network layer then sends the frame out over the network media.

## How the Receiving Host Handles the Packet

When the packet arrives on the receiving host, the packet travels through the TCP/IP protocol stack in the reverse order from which it was sent. Figure 1–1 illustrates this path. Moreover, each protocol on the receiving host strips off header information that is attached to the packet by its peer on the sending host. The following process occurs:

1. The physical network layer receives the packet in its frame form. The physical network layer computes the CRC of the packet, then sends the frame to the data link layer.
2. The data-link layer verifies that the CRC for the frame is correct and strips off the frame header and the CRC. Finally, the data-link protocol sends the frame to the Internet layer.

3. The Internet layer reads information in the header to identify the transmission. Then, the Internet layer determines if the packet is a fragment. If the transmission is fragmented, IP reassembles the fragments into the original datagram. IP then strips off the IP header and passes the datagram on to transport layer protocols.
4. The transport layer (TCP, SCTP, and UDP) reads the header to determine which application layer protocol must receive the data. Then, TCP, SCTP, or UDP strips off its related header. TCP, SCTP, or UDP sends the message or stream to the receiving application.
5. The application layer receives the message. The application layer then performs the operation that the sending host requested.

## TCP/IP Internal Trace Support

TCP/IP provides internal trace support by logging TCP communication when an RST packet terminates a connection. When an RST packet is transmitted or received, information on as many as 10 packets, which were just transmitted, is logged with the connection information.

# Finding Out More About TCP/IP and the Internet

Information about TCP/IP and the Internet is widely available. If you require specific information that is not covered in this text, you can probably find what you need in the sources cited next.

## Computer Books About TCP/IP

Many trade books about TCP/IP and the Internet are available from your local library or computer bookstore. The following two books are considered the classic texts on TCP/IP:

- Craig Hunt. *TCP/IP Network Administration* – This book contains some theory and much practical information for managing a heterogeneous TCP/IP network.
- W. Richard Stevens. *TCP/IP Illustrated, Volume I* – This book is an in-depth explanation of the TCP/IP protocols. This book is ideal for network administrators who require a technical background in TCP/IP and for network programmers.

## TCP/IP and Networking Related Web Sites

The Internet has a wealth of web sites and user groups that are devoted to TCP/IP protocols and their administration. Many manufacturers, including Sun Microsystems, offer web-based resources for general TCP/IP information. The following are helpful web resources for TCP/IP information and general system administration information.

| Web Site | Description |
| --- | --- |
| The Internet Engineering Task Force (IETF) web site (`http://www.ietf.org/home.html`) | The IETF is the body responsible for the architecture and governance of the Internet. The IETF web site contains information about the various activities of this organization. The site also includes links to the major publications of the IETF. |
| Sun Microsystem's BigAdmin Portal (`http://www.sun.com/bigadmin`) | BigAdmin provides information for administering Sun computers. The site offers FAQs, resources, discussions, links to documentation, and other materials that pertain to Solaris OS administration, including networking. |

## Requests for Comments and Internet Drafts

The Internet Engineering Task Force (IETF) working groups publish standards documents that are known as *Requests for Comments* (RFCs). Standards that are under development are published in *Internet Drafts*. The Internet Architecture Board (IAB) must approve all RFCs before they are placed in the public domain. Typically RFCs and Internet drafts are directed to developers and other highly technical readers. However, a number of RFCs that deal with TCP/IP topics contain valuable information for system administrators. These RFCs are cited in various places throughout this book.

Generally, For Your Information (FYI) documents appear as a subset of the RFCs. FYIs contain information that does not deal with Internet standards. FYIs contain Internet information of a more general nature. For example, FYI documents include a bibliography that list introductory TCP/IP books and papers. FYI documents provide an exhaustive compendium of Internet-related software tools. Finally, FYI documents include a glossary of Internet and general networking terms.

You will find references to relevant RFCs throughout this guide and other books in the Solaris System Administrator Collection.

PART II

# TCP/IP Administration

This part contains tasks and conceptual information for configuring, administering, and troubleshooting TCP/IP networks.

◆ ◆ ◆ CHAPTER 2

# Planning Your TCP/IP Network (Tasks)

This chapter describes the issues you must resolve in order to create your network in an organized, cost-effective manner. After you resolve these issues, you can devise a network plan as you configure and administer your network in the future.

This chapter contains the following information:


- "Determining the Network Hardware" on page 55
- "Obtaining Your Network's IP Number" on page 57
- "Deciding on an IP Addressing Format for Your Network" on page 55
- "Naming Entities on Your Network" on page 63
- "Planning for Routers on Your Network" on page 65


For tasks for configuring a network, refer to Chapter 5, "Configuring TCP/IP Network Services and IPv4 Addressing (Tasks)."

# Network Planning (Task Map)

| Task | Description | For Information |
|---|---|---|
| 1. Plan your hardware requirements and network topology | Determine the types of equipment that you need and the layout of this equipment at your site. | ■ For general network topology questions, refer to "Determining the Network Hardware" on page 55.<br>■ For IPv6 topology planning, refer to "Preparing the Network Topology for IPv6 Support" on page 86.<br>■ For information about a specific type of equipment, refer to the equipment manufacturer's documentation. |
| 2. Obtain a registered IP address for your network | Your network must have a unique IP address if you plan to communicate outside your local network, for example, over the Internet. | Refer to "Obtaining Your Network's IP Number" on page 57. |
| 3. Devise an IP addressing scheme for your systems, based on your IPv4 network prefix or IPv6 site prefix. | Determine how addresses are to be deployed at your site. | Refer to "Designing an IPv4 Addressing Scheme" on page 58 or refer to "Preparing an IPv6 Addressing Plan" on page 90. |
| 4. Create a list that contains the IP addresses and host names of all machines on your network. | Use the list to build network databases | Refer to "Network Databases" on page 64 |
| 5. Determine which name service to use on your network. | Decide whether to use NIS, LDAP, DNS, or the network databases in the local `/etc` directory. | Refer to "Selecting a Name Service and Directory Service" on page 63 |
| 6. Establish administrative subdivisions, if appropriate for your network | Decide if your site requires that you divide your network into administrative subdivisions | Refer to "Administrative Subdivisions" on page 65 |
| 7. Determine where to place routers in the network design. | If your network is large enough to require routers, create a network topology that supports them. | Refer to "Planning for Routers on Your Network" on page 65 |

| Task | Description | For Information |
|---|---|---|
| 8. If required, design a strategy for subnets. | You might need to create subnets for administering your IP address space or to make more IP addresses available for users. | For IPv4 subnet planning, refer to "What Is Subnetting?" on page 241<br>For IPv6 subnet planning, refer to "Creating a Numbering Scheme for Subnets" on page 91 |

# Determining the Network Hardware

When you design your network, you must decide what type of network best meets the needs of your organization. Some of the planning decisions you must make involve the following network hardware:

- The network topology, the layout, and connections of the network hardware
- The number of host systems your network can support
- The types of hosts that the network supports
- The types of servers that you might need
- The type of network media to use: Ethernet, Token Ring, FDDI, and so on
- Whether you need bridges or routers extend this media or connect the local network to external networks
- Whether some systems need separately purchased interfaces in addition to their built in interfaces

Based on these factors, you can determine the size of your local area network.

---

**Note –** How you plan the network hardware is outside the scope of this manual. For assistance, refer to the manuals that come with your hardware.

---

# Deciding on an IP Addressing Format for Your Network

The number of systems that you expect to support affects how you configure your network. Your organization might require a small network of several dozen standalone systems that are located on one floor of a single building. Alternatively, you might need to set up a network with more than 1,000 systems in several buildings. This setup can require you to further divide your network into subdivisions that are called *subnets*.

When you plan your network addressing scheme, consider the following factors:

- The type of IP address that you want to use: IPv4 or IPv6
- The number of potential systems on your network

- The number of systems that are multihomed or routers, which require an IP address for each interface
- Whether to use private addresses on your network
- Whether to have a DHCP server that manages pools of IPv4 addresses

The worldwide growth of the Internet since 1990 has resulted in a shortage of available IP addresses. To remedy this situation, the Internet Engineering Task Force (IETF) has developed a number of IP addressing alternatives. Types of IP addresses in use today include the following:

If your organization has been assigned more than one IP address for your network or uses subnets, appoint a centralized authority within your organization to assign network IP addresses. That authority should maintain control of a pool of assigned network IP addresses, and assign network, subnet, and host addresses as required. To prevent problems, ensure that duplicate or random network numbers do not exist in your organization.

## IPv4 Addresses

These 32-bit addresses are the original IP addressing format that was designed for TCP/IP. Originally, IP networks have three classes, A, B, and C. The *network number* that is assigned to a network reflects this class designation plus 8 or more bits to represent a host. Class-based IPv4 addresses require you to configure a netmask for the network number. Furthermore, to make more addresses available for systems on the local network, these addresses were often divided into subnets.

Today, IP addresses are referred to as *IPv4 addresses*. Although you can no longer obtain class-based IPv4 network numbers from an ISP, many existing networks still have them. For more information about administering IPv4 addresses, refer to "Designing Your IPv4 Addressing Scheme" on page 59.

## IPv4 Addresses in CIDR Format

The IETF has developed Classless Inter-Domain Routing (CIDR) addresses as a short to medium term fix for the shortage of IPv4 addresses. In addition, CIDR format was designed as a remedy to the lack of capacity of the global Internet routing tables. An IPv4 address with CIDR notation is 32 bits in length and has the same dotted decimal format. However, CIDR adds a prefix designation after the rightmost byte to define the network portion of the IPv4 address. For more information, refer to "Designing Your CIDR IPv4 Addressing Scheme" on page 61.

## DHCP Addresses

The Dynamic Host Configuration Protocol (DHCP) protocol enables a system to receive configuration information from a DHCP server, including an IP address, as part of the booting process. DHCP servers maintain pools of IP address from which to assign addresses to DHCP

clients. A site that uses DHCP can use a smaller pool of IP addresses than would be needed if all clients were assigned a permanent IP address. You can set up the Solaris DHCP service to manage your site's IP addresses, or a portion of the addresses. For more information, refer to Chapter 12, "About Solaris DHCP (Overview)."

## IPv6 Addresses

The IETF has deployed 128–bit IPv6 addresses as the long term solution to the shortage of available IPv4 addresses. IPv6 addresses provide greater address space than is available with IPv4. The Solaris OS supports IPv4 and IPv6 addressing on the same host, through the use of dual-stack TCP/IP. As with IPv4 addresses in CIDR format, IPv6 addresses have no notion of network classes or netmasks. As in CIDR, IPv6 addresses use prefixes to designate the portion of the address that defines the site's network. For an introduction to IPv6, refer to "IPv6 Addressing Overview" on page 74.

## Private Addresses and Documentation Prefixes

The IANA has reserved a block of IPv4 addresses and an IPv6 site prefix for use on private networks. You can deploy these addresses on systems within an enterprise network but be aware that packets with private addresses cannot be routed across the Internet. For more information on private addresses, refer to "Using Private IPv4 Addresses" on page 62.

---

**Note –** Private IPv4 addresses are also reserved for documentation purposes. The examples in this book use private IPv4 addresses and the reserved IPv6 documentation prefix.

---

# Obtaining Your Network's IP Number

An IPv4 network is defined by a combination of an IPv4 network number plus a network mask, or *netmask*. An IPv6 network is defined by its *site prefix*, and, if subnetted, its *subnet prefix*.

Unless your network plans to be private in perpetuity, your local users most likely need to communicate beyond the local network. Therefore, you must obtain a registered IP number for your network from the appropriate organization before your network can communicate externally. This address becomes the network number for your IPv4 addressing scheme or the site prefix for your IPv6 addressing scheme.

Internet Service Providers provide IP addresses for networks with pricing that is based on different levels of service. Investigate with various ISPs to determine which provides the best service for your network. ISP's typically offer dynamically allocated addresses or static IP addresses to businesses. Some ISPs offer both IPv4 and IPv6 addresses.

If your site is an ISP, you obtain IP address blocks for your customers from the Internet Registry (IR) for your locale. The Internet Assigned Numbers Authority (IANA) is ultimately responsible for delegating registered IP addresses to IRs around the world. Each IR has registration information and templates for the locale that the IR services. For information about the IANA and its IRs, refer to the IANA's IP Address Service page (`http://www.iana.org/ipaddress/ip-addresses.htm`).

**Note –** Do not arbitrarily assign IP addresses to your network, even if you are not currently attaching the network to external TCP/IP networks. Instead, use private addresses as described in "Using Private IPv4 Addresses" on page 62.

# Designing an IPv4 Addressing Scheme

**Note –** For IPv6 address planning information, refer to "Preparing an IPv6 Addressing Plan" on page 90.

This section gives an overview IPv4 addressing to aid you in designing an IPv4 addressing plan. For information on IPv6 addresses, see "IPv6 Addressing Overview" on page 74. For information on DHCP addresses, see Chapter 12, "About Solaris DHCP (Overview)."

Each IPv4-based network must have the following:

- A unique network number that is assigned by either an ISP, an IR, or, for older networks, registered by the IANA. If you plan to use private addresses, the network numbers you devise must be unique within your organization.
- Unique IPv4 addresses for the interfaces of every system on the network.
- A network mask.

The IPv4 address is a 32-bit number that uniquely identifies a network interface on a system, as explained in "How IP Addresses Apply to Network Interfaces" on page 62. An IPv4 address is written in decimal digits, divided into four 8-bit fields that are separated by periods. Each 8-bit field represents a byte of the IPv4 address. This form of representing the bytes of an IPv4 address is often referred to as the *dotted-decimal format*.

The following figure shows the component parts of an IPv4 address, `172.16.50.56`.

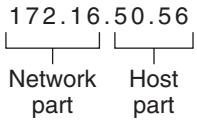



**FIGURE 2–1** IPv4 Address Format

`172.16` Registered IPv4 network number. In class-based IPv4 notation, this number also defines the IP network class, Class B in this example, that would have been registered by the IANA.

`50.56` Host part of the IPv4 address. The host part uniquely identifies an interface on a system on a network. Note that for each interface on a local network, the network part of the address is the same, but the host part must be different.

If you plan to subnet a class-based IPv4 network, you need to define a subnet mask, or *netmask*, as explained in "`netmasks` Database" on page 241.

The next example shows of the CIDR format address `192.168.3.56/22`

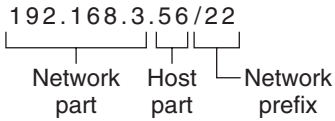



**FIGURE 2–2** CIDR Format IPv4 Address

`192.168.3` Network part, which consists of the IPv4 network number that is received from an ISP or IR.

`56` Host part, which you assign to an interface on a system.

`/22` Network prefix, which defines how many bits of the address comprise the network number. The network prefix also provides the subnet mask for the IP address. Network prefixes are also assigned by the ISP or IR.

A Solaris-based network can combine standard IPv4 addresses, CIDR format IPv4 addresses, DHCP addresses, IPv6 addresses, and private IPv4 addresses.

# Designing Your IPv4 Addressing Scheme

This section describes the classes into which standard IPv4 address are organized. Though the IANA no longer gives out class-based network numbers, these network numbers are still in use on many networks. You might need to administer the address space for a site with class-based network numbers. For a complete discussion of IPv4 network classes, refer to "Network Classes" on page 254.

The following table shows the division of the standard IPv4 address into network and host address spaces. For each class, "Range" specifies the range of decimal values for the first byte of the network number. "Network Address" indicates the number of bytes of the IPv4 address that are dedicated to the network part of the address. Each byte is represented by *xxx*. "Host Address" indicates the number of bytes that are dedicated to the host part of the address. For example, in a class A network address, the first byte is dedicated to the network, and the last three bytes are dedicated to the host. The opposite designation is true for a class C network.

**TABLE 2–1** Division of the IPv4 Classes

| Class | Byte Range | Network Number | Host Address |
|---|---|---|---|
| A | 0–127 | *xxx* | *xxx.xxx.xxx* |
| B | 128–191 | *xxx.xxx* | *xxx.xxx* |
| C | 192–223 | *xxx.xxx.xxx* | *xxx* |

The numbers in the first byte of the IPv4 address define whether the network is class A, B, or C. The remaining three bytes have a range from 0–255. The two numbers 0 and 255 are reserved. You can assign the numbers 1–254 to each byte, depending on the network class that was assigned to your network by the IANA.

The following table shows which bytes of the IPv4 address are assigned to you. The table also shows the range of numbers within each byte that are available for you to assign to your hosts.

**TABLE 2–2** Range of Available IPv4 Classes

| Network Class | Byte 1 Range | Byte 2 Range | Byte 3 Range | Byte 4 Range |
|---|---|---|---|---|
| A | 0–127 | 1–254 | 1–254 | 1–254 |
| B | 128–191 | Preassigned by IANA | 1–254 | 1–254 |
| C | 192–223 | Preassigned by IANA | Preassigned by IANA | 1–254 |

# IPv4 Subnet Number

Local networks with large numbers of hosts are sometimes divided into subnets. If you divide your IPv4 network number into subnets, you need to assign a network identifier to each subnet. You can maximize the efficiency of the IPv4 address space by using some of the bits from the host part of the IPv4 address as a network identifier. When used as a network identifier, the specified part of the address becomes the subnet number. You create a subnet number by using a netmask, which is a bitmask that selects the network and subnet parts of an IPv4 address. Refer to "Creating the Network Mask for IPv4 Addresses" on page 242 for details.

# Designing Your CIDR IPv4 Addressing Scheme

The network classes that originally constituted IPv4 are no longer in use on the global Internet. Today, the IANA distributes classless CIDR format addresses to its registries around the world. Any IPv4 address that you obtain from an ISP is in CIDR format, as shown in Figure 2–2.

The network prefix of the CIDR address indicates how many IPv4 addresses are available for hosts on your network. Note that these host addresses are assigned to interfaces on a host. If a host has more than one physical interface, you need to assign a host address for every physical interface that is in use.

The network prefix of a CIDR address also defines the length of the subnet mask. Most Solaris 10 commands recognize the CIDR prefix designation of a network's subnet mask. However, the Solaris installation program and `/etc/netmask file` require you to set the subnet mask by using dotted decimal representation. In these two cases, use the dotted decimal representation of the CIDR network prefix, as shown in the next table.

**TABLE 2–3** CIDR Prefixes and Their Decimal Equivalent

| CIDR Network Prefix | Available IP Addresses | Dotted Decimal Subnet Equivalent |
|---|---|---|
| /19 | 8,192 | 255.255.224.0 |
| /20 | 4,096 | 255.255.240.0 |
| /21 | 2,048 | 255.255.248.0 |
| /22 | 1024 | 255.255.252.0 |
| /23 | 512 | 255.255.254.0 |
| /24 | 256 | 255.255.255.0 |
| /25 | 128 | 255.255.255.128 |
| /26 | 64 | 255.255.255.192 |
| /27 | 32 | 255.255.255.224 |

For more information on CIDR addresses, refer to the following sources:

- For technical details on CIDR, refer to RFC 1519, Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy (`http://www.ietf.org/rfc/rfc1519.txt?number=1519`).
- More general information about CIDR is available from Pacific Bell Internet at Classless Inter-Domain Routing (CIDR) Overview (`http://public.pacbell.net/dedicated/cidr.html`).
- Another CIDR overview can be found in the Wikipedia article,"Classless inter-domain routing" (`http://en.wikipedia.org/wiki/Classless_inter-domain_routing`).

## Using Private IPv4 Addresses

The IANA has reserved three blocks of IPv4 addresses for companies to use on their private networks. These addresses are defined in RFC 1918, Address Allocation for Private Internets (`http://www.ietf.org/rfc/rfc1918.txt?number=1918`). You can use these *private addresses*, also known as 1918 addresses, for systems on local networks within a corporate intranet. However, private addresses are not valid on the Internet. Do not use them on systems that must communicate outside the local network.

| IPv4 Address Range | netmask |
|---|---|
| 10.0.0.0 - 10.255.255.255 | 10.0.0.0 |
| 172.16.0.0 - 172.31.255.255 | 172.16.0.0 |
| 192.168.0.0 - 192.168.255.255 | 192.168.0.0 |

## How IP Addresses Apply to Network Interfaces

To connect to the network, a system must have at least one *physical network interface*. Each network interface must have its own unique IP address. During Solaris installation, you must supply the IP address for the first interface that the installation program finds. Usually that interface has the name *device-name*0, for example `eri0` or `hme0`. This interface is considered the *primary network interface*.

If you add a second network interface to a host, that interface also must have its own unique IP address. When you add the second network interface, the host then becomes *multihomed*. By contrast, when you add a second network interface to a host and enable IP forwarding, that host becomes a router. See "Configuring an IPv4 Router" on page 115 for an explanation.

Each network interface has a device name, a device driver, and an associated device file in the `/devices` directory. The network interface might have a device name such as `eri` or `smc0`, which are device names for two commonly used Ethernet interfaces.

For information and tasks related to interfaces, refer to "Administering Interfaces in Solaris 10 3/05" on page 137 or Chapter 6, "Administering Network Interfaces (Tasks)."

---

**Note –** This book assumes that your systems have Ethernet network interfaces. If you plan to use different network media, refer to the manuals that come with the network interface for configuration information.

---

# Naming Entities on Your Network

After you receive your assigned network IP address and you have given the IP addresses to your systems, the next task is to assign names to the hosts. Then you must determine how to handle name services on your network. You use these names initially when you set up your network and later when you expand your network through routers, bridges, or PPP.

The TCP/IP protocols locate a system on a network by using its IP address. However, if you use a recognizable name, then you can easily identify the system. Therefore, the TCP/IP protocols (and the Solaris OS) require both the IP address and the host name to uniquely identify a system.

From a TCP/IP perspective, a network is a set of named entities. A host is an entity with a name. A router is an entity with a name. The network is an entity with a name. A group or department in which the network is installed can also be given a name, as can a division, a region, or a company. In theory, the hierarchy of names that can be used to identify a network has virtually no limit. The domain name identifies a *domain*.

## Administering Host Names

Many sites let users pick host names for their machines. Servers also require at least one host name, which is associated with the IP address of its primary network interface.

As a system administrator, you must ensure that each host name in your domain is unique. In other words, no two machines on your network can both have the name "fred." However, the machine "fred" might have multiple IP addresses.

When planning your network, make a list of IP addresses and their associated host names for easy access during the setup process. The list can help you verify that all host names are unique.

## Selecting a Name Service and Directory Service

The Solaris OS enables you to use three types of name services: local files, NIS, and DNS. Name services maintain critical information about the machines on a network, such as the host names, IP addresses, Ethernet addresses, and so forth. The Solaris OS also gives you the option of using the LDAP directory service in addition to or instead of a name service. For an introduction to name services on Solaris, refer to Part I, "About Naming and Directory Services," in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

## Network Databases

When you install the operating system, you supply the host name and IP address of your server, clients, or standalone system as part of the procedure. The Solaris installation program adds this information into the `hosts` and, in Solaris 10 11/06 and earlier Solaris 10 releases, the `ipnodes` network database. This database is part of a set of network databases that contain information necessary for TCP/IP operation on your network. The name service that you select for your network reads these databases.

The configuration of the network databases is critical. Therefore, you need to decide which name service to use as part of the network planning process. Moreover, the decision to use name services also affects whether you organize your network into an administrative domain. "Network Databases and the `nsswitch.conf` File" on page 245 has detailed information on the set of network databases.

## Using NIS or DNS as the Name Service

The NIS and DNS name services maintain network databases on several servers on the network. *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* describes these name services and explains how to configure the databases. In addition, the guide explain the "namespace" and "administrative domain" concepts in detail.

## Using Local Files as the Name Service

If you do not implement NIS, LDAP, or DNS, the network uses *local files* to provide the name service. The term "local files" refers to the series of files in the `/etc` directory that the network databases use. The procedures in this book assume you are using local files for your name service, unless otherwise indicated.

---

**Note –** If you decide to use local files as the name service for your network, you can set up another name service at a later date.

---

## Domain Names

Many networks organize their hosts and routers into a hierarchy of administrative domains. If you are using the NIS or DNS name service, you must select a domain name for your organization that is unique worldwide. To ensure that your domain name is unique, you should register the domain name with the InterNIC. If you plan to use DNS, you also need to register your domain name with the InterNIC.

The domain name structure is hierarchical. A new domain typically is located below an existing, related domain. For example, the domain name for a subsidiary company can be located below the domain of the parent company. If the domain name has no other relationship, an organization can place its domain name directly under one of the existing top-level domains.

The following are a few examples of top-level domains:

- `.com` – Commercial companies (international in scope)
- `.edu` – Educational institutions (international in scope)
- `.gov` – U.S. government agencies
- `.fr` – France

You select the name that identifies your organization, with the provision that the name must be unique.

### Administrative Subdivisions

The question of administrative subdivisions deals with matters of size and control. The more hosts and servers that you have in a network, the more complex your management task. You might want to handle such situations by setting up additional administrative divisions. Add networks of a particular class. Divide existing networks into subnets. The decision about setting up administrative subdivisions for your network is determined by the following factors:

- **How large is the network?**

  A single administrative division can handle a single network of several hundred hosts, all in the same physical location and requiring the same administrative services. However, sometimes you should establish several administrative subdivisions. Subdivisions are particularly useful if you have a small network with subnets and the network is scattered over an extensive geographical area.

- **Do users on the network have similar needs?**

  For example, you might have a network that is confined to a single building and supports a relatively small number of machines. These machines are divided among a number of subnetworks. Each subnetwork supports groups of users with different needs. In this example, you might use an administrative subdivision for each subnet.

## Planning for Routers on Your Network

Recall that in TCP/IP, two types of entities exist on a network: hosts and routers. All networks must have hosts, while not all networks require routers. The physical topology of the network determines if you need routers. This section introduces the concepts of network topology and routing. These concepts are important when you decide to add another network to your existing network environment.

---

**Note –** For complete details and tasks for router configuration on IPv4 networks, refer to "Packet Forwarding and Routing on IPv4 Networks" on page 109. For complete details and tasks for router configuration on IPv6 networks, refer to "Configuring an IPv6 Router" on page 176.

---

# Network Topology Overview

Network topology describes how networks fit together. Routers are the entities that connect networks to each other. A router is any machine that has two or more network interfaces and implements IP forwarding. However, the system cannot function as a router until properly configured, as described in "Configuring an IPv4 Router" on page 115.

Routers connect two or more networks to form larger internetworks. The routers must be configured to pass packets between two adjacent networks. The routers also should be able to pass packets to networks that lie beyond the adjacent networks.

The following figure shows the basic parts of a network topology. The first illustration shows a simple configuration of two networks that are connected by a single router. The second illustration shows a configuration of three networks, interconnected by two routers. In the first example, Router R joins Network 1 and Network 2 into a larger internetwork. In the second example, Router R1 connects Networks 1 and 2. Router R2 connects Networks 2 and 3. The connections form a network that includes Networks 1, 2, and 3.

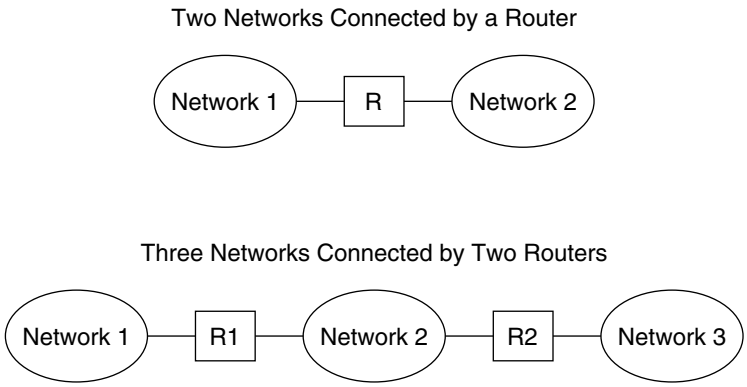



**FIGURE 2–3** Basic Network Topology

In addition to joining networks into internetworks, routers route packets between networks that are based on the addresses of the destination network. As internetworks grow more complex, each router must make more and more decisions about the packet destinations.

The following figure shows a more complex case. Router R3 directly connects networks 1 and 3. The redundancy improves reliability. If network 2 goes down, router R3 still provides a route between networks 1 and 3. You can interconnect many networks. However, the networks must use the same network protocols.

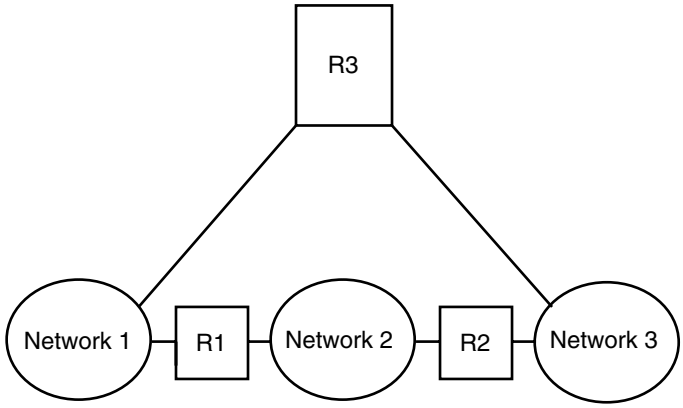



**FIGURE 2–4** A Network Topology That Provides an Additional Path Between Networks

## How Routers Transfer Packets

The IP address of the recipient, which is a part of the packet header, determines how the packet is routed. If this address includes the network number of the local network, the packet goes directly to the host with that IP address. If the network number is not the local network, the packet goes to the router on the local network.

Routers maintain routing information in *routing tables*. These tables contain the IP address of the hosts and routers on the networks to which the router is connected. The tables also contain pointers to these networks. When a router receives a packet, the router checks its routing table to determine if the table lists the destination address in the header. If the table does not contain the destination address, the router forwards the packet to another router that is listed in its routing table. Refer to "Configuring an IPv4 Router" on page 115 for detailed information on routers.

The following figure shows a network topology with three networks that are connected by two routers.

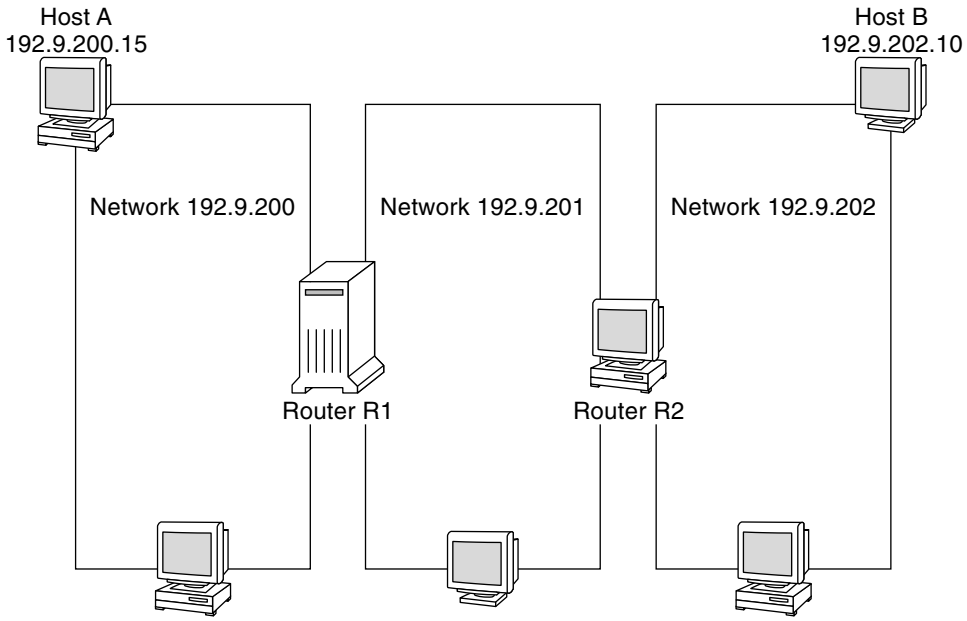



**FIGURE 2–5** A Network Topology With Three Interconnected Networks

Router R1 connects networks `192.9.200` and `192.9.201`. Router R2 connects networks `192.9.201` and `192.9.202`. If Host A on network `192.9.200` sends a message to Host B on network `192.9.202`, the following events occur:

1. Host A sends a packet out over network `192.9.200`. The packet header contains the IPv4 address of the recipient Host B, `192.9.202.10`.
2. None of the machines on network `192.9.200` has the IPv4 address `192.9.202.10`. Therefore, Router R1 accepts the packet.
3. Router R1 examines its routing tables. No machine on network `192.9.201` has the address `192.9.202.10`. However, the routing tables do list Router R2.
4. R1 then selects R2 as the "next hop" Router. R1 sends the packet to R2.
5. Because R2 connects network `192.9.201` to `192.9.202`, R2 has routing information for Host B. Router R2 then forwards the packet to network `192.9.202`, where Host B accepts the packet.

◆ ◆ ◆ CHAPTER 3

# Introducing IPv6 (Overview)

This chapter presents an overview of the Solaris Internet Protocol version 6 (IPv6) implementation. This implementation includes the associated daemon and utilities that support the IPv6 address space.

IPv6 and IPv4 addresses coexist in the Solaris networking environment. Systems that are configured with IPv6 addresses retain their IPv4 addresses, if these addresses already exist. Operations that involve IPv6 addresses do not adversely affect IPv4 operations, and vice versa.

The following major topics are discussed:


- "Major Features of IPv6" on page 69
- "IPv6 Network Overview" on page 72
- "IPv6 Addressing Overview" on page 74
- "IPv6 Neighbor Discovery Protocol Overview" on page 79
- "IPv6 Address Autoconfiguration" on page 81
- "Overview of IPv6 Tunnels" on page 82


For more detailed information about IPv6, consult the following chapters.

- IPv6 network planning – Chapter 4, "Planning an IPv6 Network (Tasks),"
- IPv6-related tasks – Chapter 7, "Configuring an IPv6 Network (Tasks)," andChapter 8, "Administering a TCP/IP Network (Tasks)."
- IPv6 details – Chapter 11, "IPv6 in Depth (Reference),"

## Major Features of IPv6

The defining feature of IPv6 is increased address space in comparison to IPv4. IPv6 also improves Internet capabilities in numerous areas, as outlined in this section.

## Expanded Addressing

IP address size increases from 32 bits in IPv4 to 128 bits in IPv6, to support more levels of addressing hierarchy. In addition, IPv6 provides many more addressable IPv6 systems. For more information, see "IPv6 Addressing Overview" on page 74.

## Address Autoconfiguration and Neighbor Discovery

The IPv6 *Neighbor Discovery (ND)* protocol facilitates the autoconfiguration of IPv6 addresses. *Autoconfiguration* is the ability of an IPv6 host to automatically generate its own IPv6 address, which makes address administration easier and less time-consuming. For more information, see "IPv6 Address Autoconfiguration" on page 81.

The Neighbor Discovery protocol corresponds to a combination of these IPv4 protocols: Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP), Router Discovery (RDISC), and ICMP Redirect. IPv6 routers use Neighbor Discovery to advertise the IPv6 site prefix. IPv6 hosts use Neighbor Discovery for various purposes, which include soliciting the prefix from an IPv6 router. For more information, see "IPv6 Neighbor Discovery Protocol Overview" on page 79.

## Header Format Simplification

The IPv6 header format either drops or makes optional certain IPv4 header fields. This change keeps the bandwidth cost of the IPv6 header as low as possible, despite the increased address size. Even though IPv6 addresses are four times longer than IPv4 addresses, the IPv6 header is only twice the size of the IPv4 header.

## Improved Support for IP Header Options

Changes in the way IP header options are encoded allow for more efficient forwarding. Also, IPv6 options have less stringent limits on their length. The changes provide greater flexibility for introducing new options in the future.

## Application Support for IPv6 Addressing

Many critical Solaris network services recognize and support IPv6 addresses, for example:

- Name services, such as DNS, LDAP, and NIS. For more information on IPv6 support by these name services, see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.
- Authentication and privacy applications, such as IP Security Architecture (IPsec) and Internet Key Exchange (IKE). For more information, see Part IV.

- Differentiated services, as provided by IP Quality of Service (IPQoS). For more information, see Part VII.
- Failover detection, as provided by IP network multipathing (IPMP). For more information, see Part VI.

# Additional IPv6 Resources

In addition to this Part, you can obtain information about IPv6 from the sources that are listed in the following sections.

## IPv6 Requests for Comments and Internet Drafts

Many RFCs are available regarding IPv6. The following table lists the major IPv6 articles and their Internet Engineering Task Force (IETF) web locations as of this writing.

**TABLE 3–1** IPv6–Related RFCs and Internet Drafts

| RFC or Internet Draft | Subject | Location |
|---|---|---|
| RFC 2461, *Neighbor Discovery for IP Version 6 (IPv6)* | Describes the features and functions of IPv6 Neighbor Discovery protocol | http://www.ietf.org/rfc/rfc2461.txt$number=2461 (`http://www.ietf.org/rfc/rfc2461.txt?number-2461`) |
| RFC 3306, *Unicast—Prefix—Based IPv6 Multicast Addresses* | Describes the format and types of IPv6 multicast addresses | ftp://ftp.rfc-editor.org/in-notes/rfc3306.txt (`ftp://ftp.rfc-editor.org/in-notes/rfc3306.txt`) |
| RFC 3484: *Default Address Selection for Internet Protocol version 6 (IPv6)* | Describes the algorithms used in IPv6 default address selection | http://www.ietf.org/rfc/rfc3484?number=3484 (`http://www.ietf.org/rfc/rfc3484.txt?number=3484`) |
| RFC 3513, *Internet Protocol version 6 (IPv6) Addressing Architecture* | Contains complete details about the types of IPv6 addresses and includes many examples | http://www.ietf.org/rfc/rfc3513.txt?number=3513 (`http://www.ietf.org/rfc/rfc3513.txt?number=3513`) |
| RFC 3587, *IPv6 Global Unicast Address Format* | Defines the standard format for IPv6 unicast addresses | http://www.ietf.org/rfc/rfc3587.txt?number=3587 (`http://www.ietf.org/rfc/rfc3587.txt?number=3587`) |

## Web Sites

The following web sites provide useful information about IPv6.

**TABLE 3–2** IPv6–Related Web Sites

| Web Site | Description | Location |
|---|---|---|
| IPv6 Forum | Links to IPv6–related presentations, events, classes, and implementations worldwide are available from this society's web site | http://www.ipv6forum.com |
| Internet Educational Task Force IPv6 Working Group | Links to all relevant IPv6 RFCs and Internet Drafts are on the home page of this IETF working group | http://www.ietf.org/html.charters/ipv6-charter.html |

# IPv6 Network Overview

This section introduces terms that are fundamental to the IPv6 network topology. The following figure shows the basic parts of an IPv6 network.

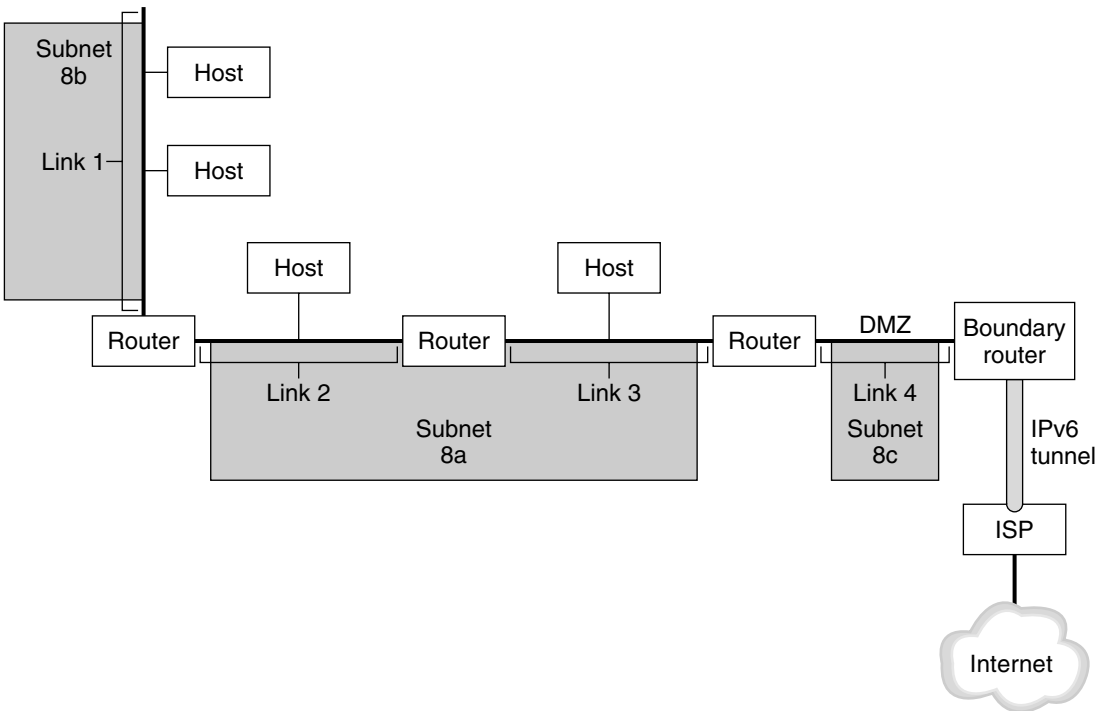



**FIGURE 3–1** Basic Components of an IPv6 Network

The figure depicts an IPv6 network and its connection to an ISP. The internal network consists of Links 1, 2, 3, and 4. Each link is populated by hosts and terminated by a router. Link 4, which

is the network's DMZ, is terminated on one end by the boundary router. The boundary router runs an IPv6 tunnel to an ISP, which provides Internet connectivity for the network. Links 2 and 3 are administered as Subnet 8a. Subnet 8b consists only of systems on Link 1. Subnet 8c is contiguous with the DMZ on Link 4.

As illustrated in Figure 3–1, an IPv6 network has essentially the same components as an IPv4 network. However, IPv6 terminology differs slightly from IPv4 terminology. Here is a list of familiar terms for network components as they are used in an IPv6 context.

**node** Any system with an IPv6 address and interface that is configured for IPv6 support. This generic term applies to both hosts and routers.

**IPv6 router** A node that forwards IPv6 packets. At least one of the router's interfaces must be configured for IPv6 support. An IPv6 router can also advertise the registered IPv6 site prefix for the enterprise over the internal network.

**IPv6 host** A node with an IPv6 address. An IPv6 host can have more than one interface that is configured for IPv6 support. As in IPv4, IPv6 hosts do not forward packets.

**link** A single, contiguous network medium that is bounded on either end by a router.

**neighbor** An IPv6 node that is on the same link as the local node.

**IPv6 subnet** The administrative segment of an IPv6 network. Components of an IPv6 subnet can directly correspond to all nodes on a link, as in IPv4. Nodes on a link can be administered in separate subnets, if required. Additionally, IPv6 does support multilink subnets, where nodes on more than one link can be components of a single subnet. Links 2 and 3 in Figure 3–1 are components of multilink Subnet 8a.

**IPv6 tunnel** A tunnel that provides a virtual point-to-point path between an IPv6 node and another IPv6 node endpoint. IPv6 supports manually configurable tunnels and automatic 6to4 tunnels.

**boundary router** The router at the edge of a network that provides one end of the IPv6 tunnel to an endpoint outside the local network. This router must have at least one IPv6 interface to the internal network. For the external network, the router can have an IPv6 interface or an IPv4 interface.

# IPv6 Addressing Overview

IPv6 addresses are assigned to interfaces, rather than to nodes, in recognition that a node can have more than one interface. Moreover, you can assign more than one IPv6 address to an interface.

---

**Note –** For complete technical information about the IPv6 address format, go to RFC 2374, IPv6 Global Unicast Address Format (`http://www.ietf.org/rfc/rfc2374.txt?number=2374`)

---

IPv6 defines three address types:

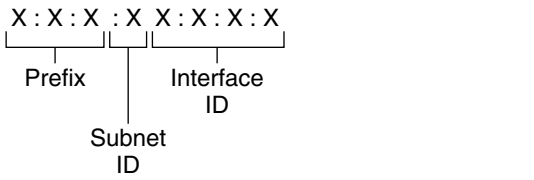
**unicast** Identifies an interface of an individual node.

**multicast** Identifies a group of interfaces, usually on different nodes. Packets that are sent to the multicast address go to all members of the *multicast group*.

**anycast** Identifies a group of interfaces, usually on different nodes. Packets that are sent to the anycast address go to the *anycast group* member node that is physically closest to the sender.

## Parts of the IPv6 Address

An IPv6 address is 128 bits in length and consists of eight, 16-bit fields, with each field bounded by a colon. Each field must contain a hexadecimal number, in contrast to the dotted-decimal notation of IPv4 addresses. In the next figure, the x's represent hexadecimal numbers.




Example:

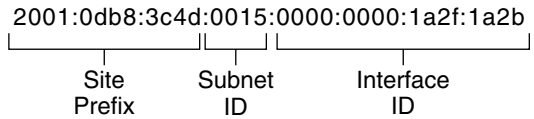



**FIGURE 3–2** Basic IPv6 Address Format

The leftmost three fields (48 bits) contain the *site prefix*. The prefix describes the *public topology* that is usually allocated to your site by an ISP or Regional Internet Registry (RIR).

The next field is the 16-bit *subnet ID*, which you (or another administrator) allocate for your site. The subnet ID describes the *private topology*, also known as the *site topology*, because it is internal to your site.

The rightmost four fields (64 bits) contain the *interface ID*, also referred to as a *token*. The interface ID is either automatically configured from the interface's MAC address or manually configured in EUI-64 format.

Consider again the address in Figure 3–2:

`2001:0db8:3c4d:0015:0000:0000:1a2f:1a2b`

This example shows all 128 bits of an IPv6 address. The first 48 bits, `2001:0db8:3c4d`, contain the site prefix, representing the public topology. The next 16 bits, `0015`, contain the subnet ID, representing the private topology for the site. The lower order, rightmost 64 bits, `0000:0000:1a2f:1a2b`, contain the interface ID.

## Abbreviating IPv6 Addresses

Most IPv6 addresses do not occupy all of their possible 128 bits. This condition results in fields that are padded with zeros or contain only zeros.

The IPv6 addressing architecture allows you use the two-colon (::) notation to represent contiguous 16-bit fields of zeros. For example, you might abbreviate the IPv6 address in Figure 3–2 by replacing the two contiguous fields of zeros in the interface ID with two colons. The resulting address is `2001:0db8:3c4d:0015::1a2f:1a2b`. Other fields of zeros can be represented as a single 0. You can also omit any leading zeros in a field, such as changing `0db8` to `db8`.

So the address `2001:0db8:3c4d:0015:0000:0000:1a2f:1a2b` can be abbreviated as `2001:db8:3c4d:15::1a2f:1a2b`.

You can use the two colon notation to replace any contiguous fields of all zeros in the IPv6 address. For example, the IPv6 address `2001:0db8:3c4d:0015:0000:d234::3eee:0000` can be collapsed into `2001:db8:3c4d:15:0:d234:3eee::`.

## Prefixes in IPv6

The leftmost fields of the IPv6 address contain the prefix, which is used for routing IPv6 packets. IPv6 prefixes have the following format:

*prefix/length in bits*

Prefix length is stated in classless inter-domain routing (CIDR) notation. CIDR notation is a slash at the end of the address that is followed by the prefix length in bits. For information on CIDR format IP addresses, refer to "Designing Your CIDR IPv4 Addressing Scheme" on page 61.

The *site prefix* of an IPv6 address occupies up to 48 of the leftmost bits of the IPv6 address. For example, the site prefix of the IPv6 address `2001:db8:3c4d:0015:0000:0000:1a2f:1a2b/48` is contained in the leftmost 48 bits, `2001:db8:3c4d`. You use the following representation, with zeros compressed, to represent this prefix:

```
2001:db8:3c4d::/48
```

---

**Note –** The prefix `2001:db8::/32` is a special IPv6 prefix that is used specifically for documentation examples.

---

You can also specify a *subnet prefix*, which defines the internal topology of the network to a router. The example IPv6 address has the following subnet prefix.

```
2001:db8:3c4d:15::/64
```

The subnet prefix always contains 64 bits. These bits include 48 bits for the site prefix, in addition to 16 bits for the subnet ID.

The following prefixes have been reserved for special use:

`2002::/16` Indicates that a 6to4 routing prefix follows.

`fe80::/10` Indicates that a link-local address follows.

`ff00::/8` Indicates that a multicast address follows.

# Unicast Addresses

IPv6 includes two different unicast address assignments:

- Global unicast address
- Link-local address

The type of unicast address is determined by the leftmost (high order) contiguous bits in the address, which contain the prefix.

The unicast address format is organized in the following hierarchy:

- Public topology
- Site (private) topology

- Interface ID

## Global Unicast Address

The global unicast address is globally unique in the Internet. The example IPv6 address that is shown in "Prefixes in IPv6" on page 75 is a global unicast address. The next figure shows the scope of the global unicast address, as compared to the parts of the IPv6 address.

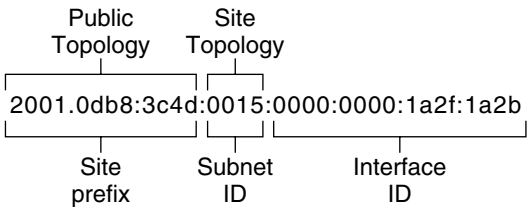



**FIGURE 3–3** Parts of the Global Unicast Address

### Public Topology

The site prefix defines the *public topology* of your network to a router. You obtain the site prefix for your enterprise from an ISP or Regional Internet Registry (RIR).

### Site Topology and IPv6 Subnets

IN IPv6, the *subnet ID* defines an administrative subnet of the network and is up to 16 bits in length. You assign a subnet ID as part of IPv6 network configuration. The *subnet prefix* defines the site topology to a router by specifying the specific link to which the subnet has been assigned.

IPv6 subnets are conceptually the same as IPv4 subnets, in that each subnet is usually associated with a single hardware link. However, IPv6 subnet IDs are expressed in hexadecimal notation, rather than in dotted decimal notation.

### Interface ID

The *interface ID* identifies an interface of a particular node. An interface ID must be unique within the subnet. IPv6 hosts can use the Neighbor Discovery protocol to automatically generate their own interface IDs. Neighbor Discovery automatically generates the interface ID, based on the MAC or EUI-64 address of the host's interface. You can also manually assign interface IDs, which is recommended for IPv6 routers and IPv6-enabled servers. For instructions on how to create a manual EUI-64 address, refer to RFC 3513 Internet Protocol Version 6 (IPv6) Addressing Architecture.

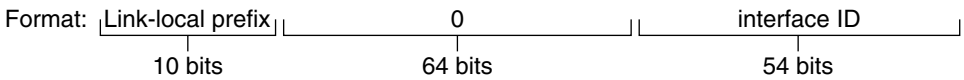
## Transitional Global Unicast Addresses

For transition purposes, the IPv6 protocol includes the ability to embed an IPv4 address within an IPv6 address. This type of IPv4 address facilitates the tunneling of IPv6 packets over existing IPv4 networks. One example of a transitional global unicast address is the 6to4 address. For more information on 6to4 addressing, refer to "6to4 Automatic Tunnels" on page 289.

## Link-Local Unicast Address

The link-local unicast address can be used only on the local network link. Link-local addresses are not valid nor recognized outside the enterprise. The following example shows the format of the link-local address.

**EXAMPLE 3–1** Parts of the Link-Local Unicast Address




Example: fe80::123e:456d

A *link-local prefix* has the following format:

`fe80::`*interface-ID*`/10`

The following is an example of a link-local address:

```
fe80::23a1:b152
```

| | |
|---|---|
| `fe80` | Hexadecimal representation of the 10-bit binary prefix 1111111010. This prefix identifies the type of IPv6 address as link local. |
| *interface-ID* | Hexadecimal address of the interface, which is usually derived from the 48-bit MAC address. |

When you enable IPv6 during Solaris installation, the lowest numbered interface on the local machine is configured with a link-local address. Each interface requires at least one link-local address to identify the node to other nodes on the local link. Therefore, you need to manually configure link-local addresses for additional interfaces of a node. After configuration, the node uses its link-local addresses for automatic address configuration and neighbor discovery.

## Multicast Addresses

IPv6 supports the use of multicast addresses. The multicast address identifies a *multicast group*, which is a group of interfaces, usually on different nodes. An interface can belong to any number of multicast groups. If the first 16 bits of an IPv6 address is `ff00`*n*, the address is a multicast address.

Multicast addresses are used for sending information or services to all interfaces that are defined as members of the multicast group. For example, one use of multicast addresses is to communicate with all IPv6 nodes on the local link.

When an interface's IPv6 unicast address is created, the kernel automatically makes the interface a member of certain multicast groups. For example, the kernel makes each node a member of the Solicited Node multicast group, which is used by the Neighbor Discovery protocol to detect reachability. The kernel also automatically makes a node a member of the All-Nodes or All Routers multicast groups.

For detailed information about multicast addresses, refer to "IPv6 Multicast Addresses in Depth" on page 260. For technical information, see RFC 3306, Unicast-Prefix-based IPv6 Multicast Addresses (`ftp://ftp.rfc-editor.org/in-notes/rfc3306.txt`), which explains the multicast address format. For more information about the proper use of multicast addresses and groups, RFC 3307, Allocation Guidelines for IPv6 Multicast Addresses (`ftp://ftp.rfc-editor.org/in-notes/rfc3307.txt`).

## Anycast Addresses and Groups

IPv6 anycast addresses identify a group of interfaces on different IPv6 nodes. Each group of interfaces is known as an *anycast group*. When a packet is sent to the anycast address, the anycast group member that is physically closest to the sender receives the packet.

---

**Note –** The Solaris Operating System (Solaris OS) implementation of IPv6 does not support the creation of anycast addresses and groups. However, Solaris IPv6 nodes can send packets to anycast addresses. For more information, see "Considerations for Tunnels to a 6to4 Relay Router" on page 291.

---

# IPv6 Neighbor Discovery Protocol Overview

IPv6 introduces the Neighbor Discovery protocol, which uses messaging as the means to handle the interaction between neighbor nodes. *Neighbor nodes* are IPv6 nodes that are on the same

link. For example, by issuing neighbor discovery-related messages, a node can learn a neighbor's link-local address. Neighbor Discovery controls the following major activities on the IPv6 local link:

- **Router discovery** – Aids hosts in locating routers on the local link.
- **Address autoconfiguration** – Enables a node to automatically configure IPv6 addresses for its interfaces.
- **Prefix discovery** – Enables nodes to discover the known subnet prefixes that have been allocated to a link. Nodes use prefixes to distinguish destinations that are on the local link from those destinations that are only reachable through a router.
- **Address resolution** – Helps nodes to determine the link-local address of a neighbor, given only the destinations's IP address.
- **Next-hop determination** – Uses an algorithm to determine the IP address of a packet recipient one hop that is beyond the local link. The next-hop can be a router or the destination node.
- **Neighbor unreachability detection** – Aids nodes to determine if a neighbor is no longer reachable. For both routers and hosts, address resolution can be repeated.
- **Duplicate address detection** – Enables a node to determine if an address that the node wants to use is not already in use.
- **Redirection** – Enables a router to inform a host of a better first-hop node to use to reach a particular destination.

Neighbor Discovery uses the following ICMP message types for communication among nodes on a link:

- Router solicitation
- Router advertisement
- Neighbor solicitation
- Neighbor advertisement
- Redirection

For detailed information on Neighbor Discovery messages and other Neighbor Discovery protocol topics, refer to "IPv6 Neighbor Discovery Protocol" on page 278. For technical information on Neighbor Discovery, see RFC 2461, Neighbor Discovery for IP Version 6 (IPv6) (`http://www.ietf.org/rfc/rfc2461.txt?number=2461`).

# IPv6 Address Autoconfiguration

A major feature of IPv6 is a host's ability to autoconfigure an interface. Through Neighbor Discovery, the host locates an IPv6 router on the local link and requests a site prefix. The host does the following, as part of the autoconfiguration process:

- Creates a link-local address for each interface, which does not require a router on the link.
- Verifies the address's uniqueness on a link, which does not require a router on the link.
- Determines if the global addresses should be obtained through the stateless mechanism, the stateful mechanism, or both mechanisms. (Requires a router on the link.)

## Stateless Autoconfiguration Overview

Stateless autoconfiguration requires no manual configuration of hosts, minimal (if any) configuration of routers, and no additional servers. The stateless mechanism enables a host to generate its own addresses. The stateless mechanism uses local information as well as nonlocal information that is advertised by routers to generate the addresses.

You can implement temporary addresses for an interface, which are also autoconfigured. You enable a temporary address token for one or more interfaces on a host. However, unlike standard, autoconfigured IPv6 addresses, a temporary address consists of the site prefix and a randomly generated 64 bit number. This random number becomes the interface ID portion of the IPv6 address. A link-local address is not generated with the temporary address as the interface ID.

Routers advertise all prefixes that have been assigned on the link. IPv6 hosts use Neighbor Discovery to obtain a subnet prefix from a local router. Hosts automatically create IPv6 addresses by combining the subnet prefix with an interface ID that is generated from an interface's MAC address. In the absence of routers, a host can generate only link-local addresses. Link-local addresses can only be used for communication with nodes on the same link.

---

**Note –** Do not use stateless autoconfiguration to create the IPv6 addresses of servers. Hosts automatically generate interface IDs that are based on hardware-specific information during autoconfiguration. The current interface ID could become invalid if the existing interface is swapped for a new interface.

---

# Overview of IPv6 Tunnels

For most enterprises, the introduction of IPv6 to an existing IPv4 network must occur on a gradual, step-by-step basis. The Solaris dual-stack network environment supports both IPv4 and IPv6 functionality. Because most networks use the IPv4 protocol, IPv6 networks currently require a way to communicate outside their borders. IPv6 networks use tunnels for this purpose.

In most IPv6 tunneling scenarios, the outbound IPv6 packet is encapsulated inside an IPv4 packet. The boundary router of the IPv6 network sets up a point-to-point tunnel over various IPv4 networks to the boundary router of the destination IPv6 network. The packet travels over the tunnel to the destination network's boundary router, which decapsulates the packet. Then, the router forwards the separate IPv6 packet to the destination node.

The Solaris IPv6 implementation supports the following tunneling scenarios:

- A manually configured tunnel between two IPv6 networks, over an IPv4 network. The IPv4 network can be the Internet or a local network within an enterprise.
- A manually configured tunnel between two IPv4 networks, over an IPv6 network, usually within an enterprise.
- A dynamically configured automatic 6to4 tunnel between two IPv6 networks, over an IPv4 network at an enterprise or over the Internet.

For detailed information about IPv6 tunnels, refer to "IPv6 Tunnels" on page 285. For information about IPv4- to-IPv4 tunnels and VPN, refer to "Virtual Private Networks and IPsec" on page 489.

◆ ◆ ◆ CHAPTER 4

# Planning an IPv6 Network (Tasks)

Deploying IPv6 on a new network or an existing network requires a major planning effort. This chapter contains the planning tasks that are necessary before you can configure IPv6 at your site. For existing networks, IPv6 deployment should be phased in gradually. The topics in this chapter help you phase in IPv6 onto an otherwise IPv4-only network.

The following topics are discussed in this chapter:


- "IPv6 Planning (Task Maps)" on page 83
- "IPv6 Network Topology Scenario" on page 84
- "Preparing the Existing Network to Support IPv6" on page 86
- "Preparing an IPv6 Addressing Plan" on page 90


For an introduction to IPv6 concepts, refer to Chapter 3, "Introducing IPv6 (Overview)." For detailed information, refer to Chapter 11, "IPv6 in Depth (Reference)."

## IPv6 Planning (Task Maps)

Complete the tasks in the next task map in sequential order to accomplish the planning tasks necessary for IPv6 deployment.

| Task | Description | For Instructions |
|---|---|---|
| 1. Prepare your hardware to support IPv6. | Ensure that your hardware can be upgraded to IPv6. | "Preparing the Network Topology for IPv6 Support" on page 86 |
| 2. Get an ISP that supports IPv6. | Ensure that your current ISP supports IPv6. Otherwise, find an ISP who can support IPv6. You can use two ISPs, one ISP for IPv6 and one for ISP IPv4 communications. | |

| Task | Description | For Instructions |
|---|---|---|
| 3. Ensure that your applications are IPv6 ready. | Verify that your applications can run in an IPv6 environment. | "How to Prepare Network Services for IPv6 Support" on page 88 |
| 4. Get a site prefix. | Obtain a 48-bit site prefix for your site from your ISP or from the nearest RIR. | "Obtaining a Site Prefix" on page 90 |
| 5. Create a subnet addressing plan. | You need to plan the overall IPv6 network topology and addressing scheme before you can configure IPv6 on the various nodes in your network. | "Creating a Numbering Scheme for Subnets" on page 91 |
| 6. Design a plan for tunnel usage. | Determine which routers should run tunnels to other subnets or external networks. | "Planning for Tunnels in the Network Topology" on page 89 |
| 7. Create an addressing plan for entities on the network. | Your plan for addressing servers, routers, and hosts should be in place before IPv6 configuration. | "Creating an IPv6 Addressing Plan for Nodes" on page 91 |
| 8. Develop an IPv6 security policy. | Investigate IP Filter, IP security architecture (IPsec), Internet Key Exchange (IKE), and other Solaris security features as you develop an IPv6 security policy. | Part IV |
| 9. (Optional) Set up a DMZ. | For security purposes, you need an addressing plan for the DMZ and its entities before you configure IPv6. | "Security Considerations for the IPv6 Implementation" on page 90 |
| 10. Enable the nodes to support IPv6. | Configure IPv6 on all routers and hosts. | "IPv6 Router Configuration (Task Map)" on page 176 |
| 11. Turn on network services. | Make sure that existing servers can support IPv6. | "Major TCP/IP Administrative Tasks (Task Map)" on page 203 |
| 12. Update name servers for IPv6 support. | Make sure that DNS, NIS, and LDAP servers are updated with the new IPv6 addresses. | "Configuring Name Service Support for IPv6" on page 197 |

# IPv6 Network Topology Scenario

The tasks throughout this chapter explain how to plan for IPv6 services on a typical enterprise network. The following figure shows the network that is referred to throughout the chapter. Your proposed IPv6 network might include some or all of the network links that are illustrated in this figure.

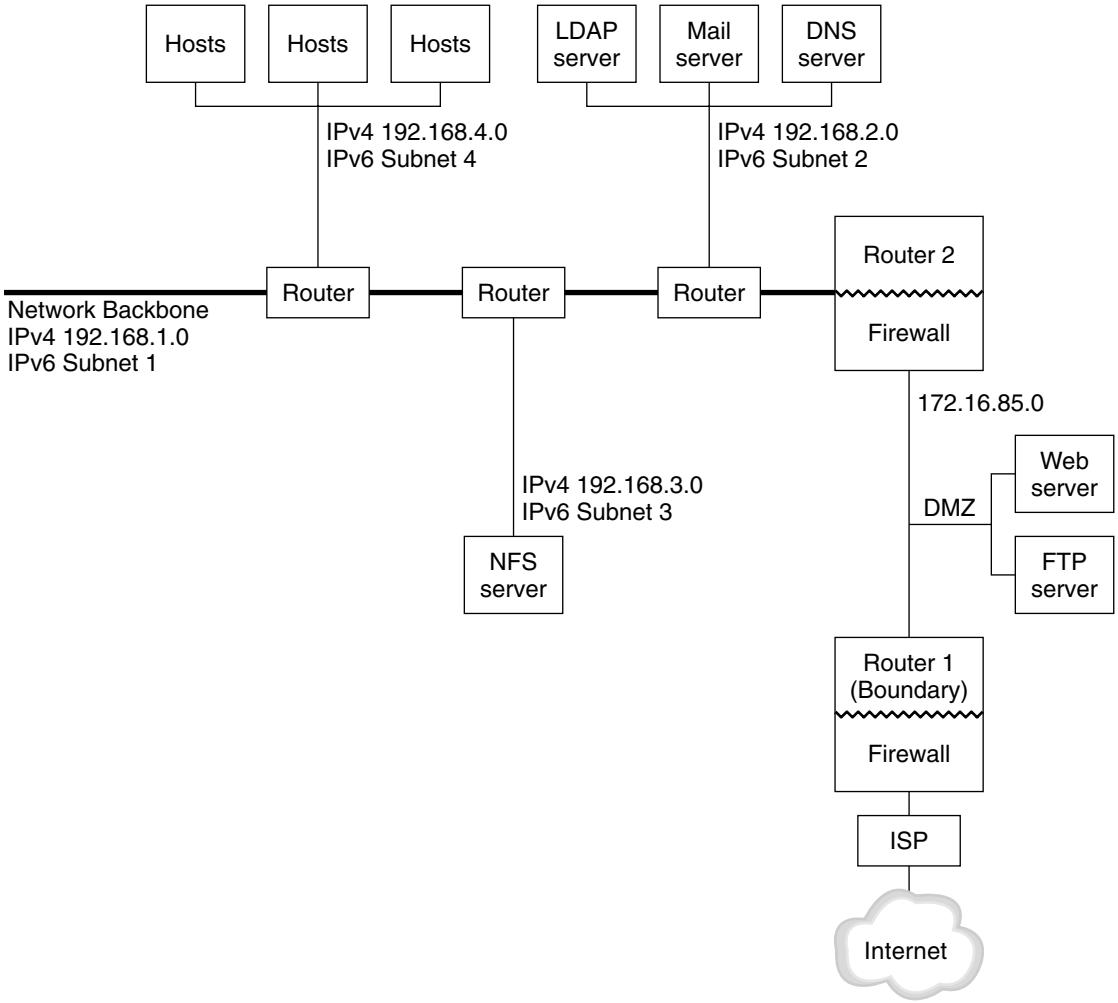



**FIGURE 4–1** IPv6 Network Topology Scenario

The enterprise network scenario consists of five subnets with existing IPv4 addresses. The links of the network correspond directly to the administrative subnets. The four internal networks are shown with RFC 1918-style private IPv4 addresses, which is a common solution for the lack of IPv4 addresses. The addressing scheme of these internal networks follows:

- Subnet 1 is the internal network backbone `192.168.1`.
- Subnet 2 is the internal network `192.168.2`, with LDAP, `sendmail`, and DNS servers.
- Subnet 3 is the internal network `192.168.3`, with the enterprise's NFS servers.

- Subnet 4 is the internal network `192.168.4`, which contains hosts for the enterprise's employees.

The external, public network `172.16.85` functions as the corporation's DMZ. This network contains web servers, anonymous FTP servers, and other resources that the enterprise offers to the outside world. Router 2 runs a firewall and separates public network `172.16.85` from the internal backbone. On the other end of the DMZ, Router 1 runs a firewall and serves as the enterprise's boundary server.

In Figure 4–1, the public DMZ has the RFC 1918 private address `172.16.85`. In the real world, the public DMZ must have a registered IPv4 address. Most IPv4 sites use a combination of public addresses and RFC 1918 private addresses. However, when you introduce IPv6, the concept of public addresses and private addresses changes. Because IPv6 has a much larger address space, you use public IPv6 addresses on both private networks and public networks.

# Preparing the Existing Network to Support IPv6

**Note –** The Solaris dual protocol stack supports concurrent IPv4 and IPv6 operations. You can successfully run IPv4–related operations during and after deployment of IPv6 on your network.

IPv6 introduces additional features to an existing network. Therefore, when you first deploy IPv6, you must ensure that you do not disrupt any operations that are working with IPv4. The subjects covered in this section describe how to introduce IPv6 to an existing network in a step-by-step fashion.

## Preparing the Network Topology for IPv6 Support

The first step in IPv6 deployment is to assess which existing entities on your network can support IPv6. In most cases, the network topology-wires, routers, and hosts-can remain unchanged as you implement IPv6. However, you might have to prepare existing hardware and applications for IPv6 before actually configuring IPv6 addresses on network interfaces.

Verify which hardware on your network can be upgraded to IPv6. For example, check the manufacturers' documentation for IPv6 readiness regarding the following classes of hardware:

- Routers
- Firewalls
- Servers
- Switches

**Note –** All procedures in the this Part assume that your equipment, particularly routers, can be upgraded to IPv6.

Some router models cannot be upgraded to IPv6. For more information and a workaround, refer to "IPv4 Router Cannot Be Upgraded to IPv6" on page 231.

## Preparing Network Services for IPv6 Support

The following typical IPv4 network services in the current Solaris release are IPv6 ready:

- `sendmail`
- NFS
- HTTP (Apache 2.x or Orion)
- DNS
- LDAP

The IMAP mail service is for IPv4 only.

Nodes that are configured for IPv6 can run IPv4 services. When you turn on IPv6, not all services accept IPv6 connections. Services that have been ported to IPv6 will accept a connection. Services that have not been ported to IPv6 continue to work with the IPv4 half of the protocol stack.

Some issues can arise after you upgrade services to IPv6. For details, see "Problems After Upgrading Services to IPv6" on page 231.

## Preparing Servers for IPv6 Support

Because servers are considered IPv6 hosts, by default their IPv6 addresses are automatically configured by the Neighbor Discovery protocol. However, many servers have multiple network interface cards (NICs) that you might want to swap out for maintenance or replacement. When you replace one NIC, Neighbor Discovery automatically generates a new interface ID for that NIC. This behavior might not be acceptable for a particular server.

Therefore, consider manually configuring the interface ID portion of the IPv6 addresses for each interface of the server. For instructions, refer to "How to Configure a User-Specified IPv6 Token" on page 185. Later, when you need to replace an existing NIC, the already configured IPv6 address is applied to the replacement NIC.

## ▼ How to Prepare Network Services for IPv6 Support

1 **Update the following network services to support IPv6:**

- Mail servers
- NIS servers
- NFS

---

**Note –** LDAP supports IPv6 without requiring IPv6-specific configuration tasks.

---

2 **Verify that your firewall hardware is IPv6 ready.**

Refer to the appropriate firewall-related documentation for instructions.

3 **Verify that other services on your network have been ported to IPv6.**

For more information, refer to marketing collateral and associated documentation for the software.

4 **If your site deploys the following services, make sure that you have taken the appropriate measures for these services:**

- Firewalls

  Consider strengthening the policies that are in place for IPv4 to support IPv6. For more security considerations, see "Security Considerations for the IPv6 Implementation" on page 90.
- Mail

  In the MX records for DNS, consider adding the IPv6 address of your mail server.
- DNS

  For DNS-specific considerations, see "How to Prepare DNS for IPv6 Support" on page 88.
- IPQoS

  Use the same Diffserv policies on a host that were used for IPv4. For more information, see "Classifier Module" on page 835.

5 **Audit any network services that are offered by a node prior to converting that node to IPv6.**

## ▼ How to Prepare DNS for IPv6 Support

The current Solaris release supports DNS resolution on both the client side and the server side. Do the following to prepare DNS services for IPv6.

For more information that is related to DNS support for IPv6, refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

1. **Ensure that the DNS server that performs recursive name resolution is dual-stacked (IPv4 and IPv6) or for IPv4 only.**
2. **On the DNS server, populate the DNS database with relevant IPv6 database AAAA records in the forward zone.**

---

**Note –** Servers that run multiple critical services require special attention. Ensure that the network is working properly. Also ensure that all critical services are ported to IPv6. Then, add the server's IPv6 address to the DNS database.

---

3. **Add the associated PTR records for the AAAA records into the reverse zone.**
4. **Add either IPv4 only data, or both IPv6 and IPv4 data into the NS record that describes zones.**

# Planning for Tunnels in the Network Topology

The IPv6 implementation supports a number of tunnel configurations to serve as transition mechanisms as your network migrates to a mix of IPv4 and IPv6. Tunnels enable isolated IPv6 networks to communicate. Because most of the Internet runs IPv4, IPv6 packets from your site need to travel across the Internet through tunnels to destination IPv6 networks.

Here are some major scenarios for using tunnels in the IPv6 network topology:

- The ISP from which you purchase IPv6 service allows you to create a tunnel from your site's boundary router to the ISP network. Figure 4–1 shows such a tunnel. In such a case, you would run a manual, IPv6 over IPv4 tunnel.
- You manage a large, distributed network with IPv4 connectivity. To connect the distributed sites that use IPv6, you can run an automatic 6to4 tunnel from the edge router of each subnet.
- Sometimes, a router in your infrastructure cannot be upgraded to IPv6. In this case, you can create a manual tunnel over the IPv4 router, with two IPv6 routers as endpoints.

For procedures for configuring tunnels, refer to "Tasks for Configuring Tunnels for IPv6 Support (Task Map)" on page 188. For conceptual information regarding tunnels, refer to "IPv6 Tunnels" on page 285.

## Security Considerations for the IPv6 Implementation

When you introduce IPv6 into an existing network, you must take care not to compromise the security of the site. Be aware of the following security issues as you phase in your IPv6 implementation:

- The same amount of filtering is required for both IPv6 packets and IPv4 packets.
- IPv6 packets are often tunneled through a firewall. Therefore, you should implement either of the following scenarios:
  - Have the firewall do content inspection inside the tunnel.
  - Put an IPv6 firewall with similar rules at the opposite tunnel endpoint.
- Some transition mechanisms exist that use IPv6 over UDP over IPv4 tunnels. These mechanisms might prove dangerous by short-circuiting the firewall.
- IPv6 nodes are globally reachable from outside the enterprise network. If your security policy prohibits public access, you must establish stricter rules for the firewall. For example, consider configuring a stateful firewall.

This book includes security features that can be used within an IPv6 implementation.

- The IP security architecture (IPsec) feature enables you to provide cryptographic protection for IPv6 packets. For more information, refer to Chapter 19, "IP Security Architecture (Overview)."
- The Internet Key Exchange (IKE) feature enables you to use public key authentication for IPv6 packets. For more information, refer to Chapter 22, "Internet Key Exchange (Overview)."

# Preparing an IPv6 Addressing Plan

A major part of the transition from IPv4 to IPv6 includes the development of an addressing plan. This task involves the following preparations:


- "Obtaining a Site Prefix" on page 90
- "Creating the IPv6 Numbering Scheme" on page 91


## Obtaining a Site Prefix

Before you configure IPv6, you must obtain a site prefix. The site prefix is used to derive IPv6 addresses for all the nodes in your IPv6 implementation. For an introduction to site prefixes, refer to "Prefixes in IPv6" on page 75.

Any ISP that supports IPv6 can provide your organization with a 48-bit IPv6 site prefix. If your current ISP only supports IPv4, you can use another ISP for IPv6 support while retaining your

current ISP for IPv4 support. In such an instance, you can use one of several workarounds. For more information, see "Current ISP Does Not Support IPv6" on page 231.

If your organization is an ISP, then you obtain site prefixes for your customers from the appropriate Internet registry. For more information, see the Internet Assigned Numbers Authority (IANA) (`http://www.iana.org`).

# Creating the IPv6 Numbering Scheme

Unless your proposed IPv6 network is entirely new, use your existing IPv4 topology as the basis for the IPv6 numbering scheme.

## Creating a Numbering Scheme for Subnets

Begin your numbering scheme by mapping your existing IPv4 subnets into equivalent IPv6 subnets. For example, consider the subnets illustrated in Figure 4–1. Subnets 1–4 use the RFC 1918 IPv4 private address designation for the first 16 bits of their addresses, in addition to the digits 1–4 to indicate the subnet. For illustrative purposes, assume that the IPv6 prefix `2001:db8:3c4d/48` has been assigned to the site. The following table shows how the private IPv4 prefixes map into IPv6 prefixes.

| IPv4 Subnet Prefix | Equivalent IPv6 Subnet Prefix |
|---|---|
| `192.168.1.0/24` | `2001:db8:fd3c4d:1::/64` |
| `192.168.2.0/24` | `2001:db8:3c4d:2::/64` |
| `192.168.3.0/24` | `2001:db8:3c4d:3::/64` |
| `192.168.4.0/24` | `2001:db8:3c4d:4::/64` |

## Creating an IPv6 Addressing Plan for Nodes

For most hosts, stateless autoconfiguration of IPv6 addresses for their interfaces is an appropriate, time saving strategy. When the host receives the site prefix from the nearest router, Neighbor Discovery automatically generates IPv6 addresses for each interface on the host.

Servers need to have stable IPv6 addresses. If you do not manually configure a server's IPv6 addresses, a new IPv6 address is autoconfigured whenever a NIC card is replaced on the server. Keep the following tips in mind when you create addresses for servers:

- Give servers meaningful and stable interface IDs. One strategy is to use a sequential numbering scheme for interface IDs. For example, the internal interface of the LDAP server in Figure 4–1 might become `2001:db8:3c4d:2::2`.

- Alternatively, if you do not regularly renumber your IPv4 network, consider using the existing IPv4 addresses of the routers and servers as their interface IDs. In Figure 4–1, suppose Router 1's interface to the DMZ has the IPv4 address `123.456.789.111`. You can convert the IPv4 address to hexadecimal and use the result as the interface ID. The new interface ID would be `::7bc8:156F`.

  Only use this approach if you own the registered IPv4 address, rather than having obtained the address from an ISP. If you use an IPv4 address that was given to you by an ISP, you create a dependency that would create problems if you change ISPs.

Due to the limited number of IPv4 addresses, in the past a network designer had to consider where to use global, registered addresses and private, RFC 1918 addresses. However, the notion of global and private IPv4 addresses does not apply to IPv6 addresses. You can use global unicast addresses, which include the site prefix, on all links of the network, including the public DMZ.

◆ ◆ ◆ CHAPTER 5

# Configuring TCP/IP Network Services and IPv4 Addressing (Tasks)

TCP/IP network administration evolves in two stages. The first stage is to assemble the hardware. Then, you configure the daemons, files, and services that implement the TCP/IP protocol.

This chapter explains how to configure TCP/IP on a network that implements IPv4 addressing and services.

---

**Note –** Many of the tasks in this chapter apply to both IPv4-only and IPv6-enabled networks. Where configuration tasks differ between the two addressing formats, the IPv4 configuration steps are in this chapter. The tasks in this chapter then cross reference the equivalent IPv6 tasks in Chapter 7, "Configuring an IPv6 Network (Tasks)."

---

This chapter contains the following information:


- "Before You Configure an IPv4 Network (Task Map)" on page 94
- "Determining Host Configuration Modes" on page 95
- "Adding a Subnet to a Network (Task Map)" on page 98
- "Configuring Systems on the Local Network " on page 99
- "Network Configuration Task Map" on page 98
- "Packet Forwarding and Routing on IPv4 Networks" on page 109
- "Monitoring and Modifying Transport Layer Services" on page 132
- "Administering Interfaces in Solaris 10 3/05" on page 137

# What's New in This Chapter

In Solaris 10 8/07, the following changes are made:

- You can configure and manage routing through the Service Management Facility (SMF) as an alternative to using the `routeadm` command. For instructions, refer to the procedures and examples in "Packet Forwarding and Routing on IPv4 Networks" on page 109and the `routeadm`(1M) man page.
- The `/etc/inet/ipnodes` file becomes obsolete. Use `/etc/inet/ipnodes` only for earlier Solaris 10 releases, as explained in the individual procedures.

# Before You Configure an IPv4 Network (Task Map)

Before you configure TCP/IP, complete the tasks that are listed in the following table.

| Task | Description | For Instructions |
|---|---|---|
| 1. Design the network topology. | Determine the physical layout of the network. | "Network Topology Overview" on page 66 and "IPv4 Autonomous System Topology" on page 113 |
| 2. Obtain a network number from your ISP or Regional Internet Registry (RIR). | Get a registered network number, which enables systems at your site to communicate externally. | "Designing Your IPv4 Addressing Scheme" on page 59. |
| 3. Plan the IPv4 addressing scheme for the network. If applicable, include subnet addressing. | Use the network number as the basis for your addressing plan. | "Designing Your IPv4 Addressing Scheme" on page 59. |
| 4. Assemble the network hardware depending on the network topology. Assure that the hardware is functioning properly. | Set up the systems, network media, routers, switches, hubs and bridges that you outlined in the network topology design. | The hardware manuals and "Network Topology Overview" on page 66. |
| 5. Assign IPv4 addresses and host names to all systems in the network. | Assign the IPv4 addresses during Solaris OS installation or post installation, in the appropriate files. | "Designing Your IPv4 Addressing Scheme" on page 59 and "How to Change the IPv4 Address and Other Network Configuration Parameters" on page 105 |
| 6. Run configuration software that is required by network interfaces and routers, if applicable. | Configure routers and multihomed hosts. | "Planning for Routers on Your Network" on page 65 and "Configuring an IPv4 Router" on page 115 for information on routers. |

| Task | Description | For Instructions |
|---|---|---|
| 7. Determine which name service or directory service your network uses: NIS, LDAP, DNS, or local files. | Configure your selected name service and/or directory service. | *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP).* |
| 8. Select domain names for your network, if applicable. | Choose a domain name for your network and register it with the InterNIC. | *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* |

# Determining Host Configuration Modes

As a network administrator, you configure TCP/IP to run on hosts and routers (if applicable). You can configure these systems to obtain configuration information from files on the local system or from files that are located on other systems on the network. You need the following configuration information:

- Host name of each system
- IP address of each system
- Domain name to which each system belongs
- Default router
- IPv4 netmask in use on each system's network

A system that obtains TCP/IP configuration information from local files operates in *local files mode*. A system that obtains TCP/IP configuration information from a remote network server operates in *network client mode*.

## Systems That Should Run in Local Files Mode

To run in local files mode, a system must have local copies of the TCP/IP configuration files. These files are described in "TCP/IP Configuration Files" on page 235. The system should have its own disk, though this recommendation is not strictly necessary.

Most servers should run in local files mode. This requirement includes the following servers:

- Network configuration servers
- NFS servers
- Name servers that supply NIS, LDAP, or DNS services
- Mail servers

Additionally, routers should run in local files mode.

Systems that function exclusively as print servers do not need to run in local files mode. Whether individual hosts should run in local files mode depends on the size of your network.

If you are running a very small network, the amount of work that is involved in maintaining these files on individual hosts is manageable. If your network serves hundreds of hosts, the task becomes difficult, even with the network divided into a number of administrative subdomains. Thus, for large networks, using local files mode is usually less efficient. However, because routers and servers must be self-sufficient, they should be configured in local files mode.

### Network Configuration Servers

*Network configuration servers* are the servers that supply the TCP/IP configuration information to hosts that are configured in network client mode. These servers support three booting protocols:

- RARP – Reverse Address Resolution Protocol (RARP) maps Ethernet addresses (48 bits) to IPv4 addresses (32 bits), which is the reverse of ARP. When you run RARP on a network configuration server, hosts that are running in network client mode obtain their IP addresses and TCP/IP configuration files from the server. The `in.rarpd` daemon enables RARP services. Refer to the `in.rarpd`(1M) man page for details.
- TFTP – The Trivial File Transfer Protocol (TFTP) is an application that transfers files between remote systems. The `in.tftpd` daemon executes TFTP services, enabling file transfer between network configuration servers and their network clients. Refer to the `in.tftpd`(1M) man page for details.
- Bootparams – The Bootparams protocol supplies parameters for booting that are required by clients that boot off the network. The `rpc.bootparamd` daemon executes these services. Refer to the `bootparamd`(1M) man page for details.

Network configuration servers can also function as NFS file servers.

If you are configuring any hosts as network clients, then you must also configure at least one system on your network as a network configuration server. If your network is subnetted, then you must have at least one network configuration server for each subnet with network clients.

## Systems That Are Network Clients

Any host that obtains its configuration information from a network configuration server operates in network client mode. Systems that are configured as network clients do not require local copies of the TCP/IP configuration files.

*Network client mode* simplifies administration of large networks. Network client mode minimizes the number of configuration tasks that you perform on individual hosts. Network client mode assures that all systems on the network adhere to the same configuration standards.

You can configure network client mode on all types of computers. For example, you can configure network client mode on standalone systems.

## Mixed Configurations

Configurations are not limited to either an all-local-files mode or an all-network-client mode. Routers and servers should always be configured in local mode. For hosts, you can use any combination of local files and network client mode.

## IPv4 Network Topology Scenario

Figure 5–1 shows the hosts of a fictitious network with the network number `192.9.200`. The network has one network configuration server, which is called `sahara`. Hosts `tenere` and `nubian` have their own disks and run in local files mode. Host `faiyum` also has a disk, but this system operates in network client mode.

Finally, the system `timbuktu` is configured as a router. The system includes two network interfaces. The first interface is named `timbuktu`. This interface belongs to network `192.9.200`. The second interface is named `timbuktu-201`. This interface belongs to network `192.9.201`. Both networks are in the organizational domain `deserts.worldwide.com`. The domain uses local files as its name service.

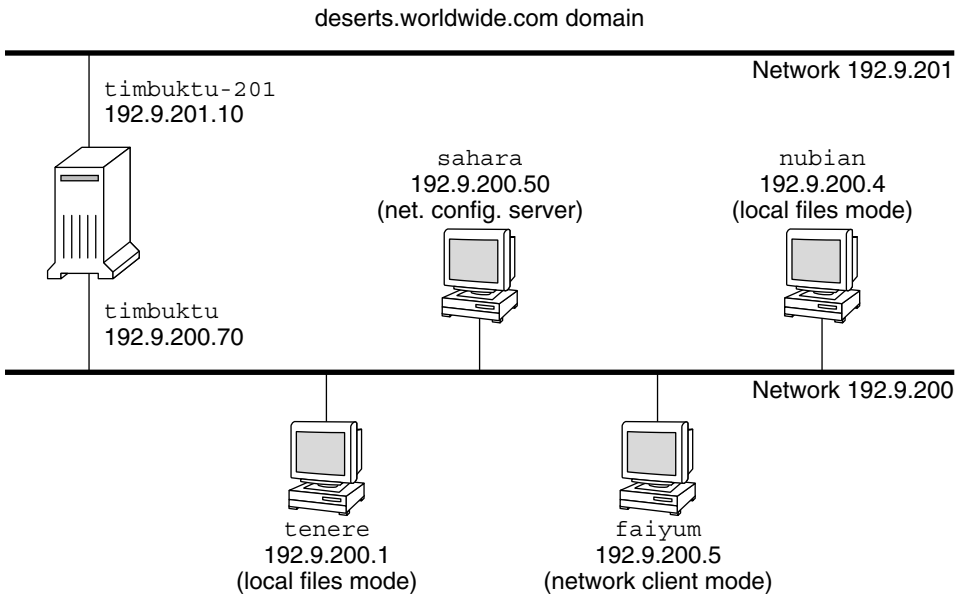



**FIGURE 5–1** Hosts in an IPv4 Network Topology Scenario

# Adding a Subnet to a Network (Task Map)

If you are changing from a network that does not use a subnet to a network that does use a subnet, perform the tasks in the following task map.

**Note –** The information in this section applies to IPv4 subnets only. For information on planning IPv6 subnets, refer to "Preparing the Network Topology for IPv6 Support" on page 86 and "Creating a Numbering Scheme for Subnets" on page 91.

| Task | Description | For Instructions |
|---|---|---|
| 1. Determine if your network topology requires subnets. | Decide on the new subnet topology, including where to locate routers and hosts on the subnets. | "Planning for Routers on Your Network" on page 65, "What Is Subnetting?" on page 241, and "Network Classes" on page 254 |
| 2. Assign the IP addresses with the new subnet number to the systems to become members of the subnet. | Configure IP addresses that use the new subnet number, either during Solaris OS installation or later, in the `/etc/hostname.`*interface* file. | "Deciding on an IP Addressing Format for Your Network" on page 55 |
| 3. Configure the network mask of the subnet on all prospective systems in the subnet. | Modify the `/etc/inet/netmasks` file, if you are manually configuring network clients. Or, supply the netmask to the Solaris installation program. | "`netmasks` Database" on page 241 and "Creating the Network Mask for IPv4 Addresses" on page 242 |
| 4. Edit the network databases with the new IP addresses of all systems in the subnet. | Modify `/etc/inet/hosts` and, for Solaris 10 11/06 and earlier releases,`/etc/inet/ipnodes`, on all hosts to reflect the new host addresses. | "`hosts` Database" on page 237 |
| 5. Reboot all systems. | | |

# Network Configuration Task Map

| Task | Description | For Instructions |
|---|---|---|
| Configure a host for local files mode | Involves editing the `nodename`, `hostname`, `hosts`, `defaultdomain`, `defaultrouter`, and `netmasks` files | "How to Configure a Host for Local Files Mode" on page 100 |

| Task | Description | For Instructions |
|---|---|---|
| Set up a network configuration server | Involves turning on the `in.tftp` daemon, and editing the `hosts`, `ethers`, and `bootparams` files | "How to Set Up a Network Configuration Server" on page 102 |
| Configure a host for network client mode | Involves creating the `hostname` file, editing the `hosts` file, and deleting the `nodename` and `defaultdomain` files, if they exist | "How to Configure Hosts for Network Client Mode" on page 104 |
| Specify a routing strategy for the network client | Involves determining whether to use static routing or dynamic routing on the host. | "How to Enable Static Routing on a Single-Interface Host" on page 128 and "How to Enable Dynamic Routing on a Single-Interface Host" on page 130. |
| Modify the existing network configuration | Involves changing the host name, IP address, and other parameters that were set at installation or configured at a later time. | "How to Change the IPv4 Address and Other Network Configuration Parameters" on page 105 |

# Configuring Systems on the Local Network

Network software installation occurs along with the installation of the operating system software. At that time, certain IP configuration parameters must be stored in appropriate files so that they can be read at boot time.

The network configuration process involves creating or editing the network configuration files. How configuration information is made available to a system's kernel is conditional. The availability depends on whether these files are stored locally (local files mode) or acquired from the network configuration server (network client mode).

The parameters that are supplied during network configuration follow:

- The IP address of each network interface on every system.
- The host names of each system on the network. You can type the host name in a local file or a name service database.
- The NIS, LDAP, or DNS domain name in which the system resides, if applicable.
- The default router addresses. You supply this information if you have a simple network topology with only one router attached to each network. You also supply this information if your routers do not run routing protocols such as the Router Discovery Server Protocol (RDISC) or the Router Information Protocol (RIP). For more information on default routers, refer to "Packet Forwarding and Routing on IPv4 Networks" on page 109 See Table 5–1 for a list of routing protocols supported in the Solaris OS.
- Subnet mask (required only for networks with subnets).

If the Solaris installation program detects more one interface on the system, you can optionally configure the additional interfaces during installation. For complete instructions, see *Solaris Express Installation Guide: Basic Installations*.

This chapter contains information on creating and editing local configuration files. See *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for information on working with name service databases.

## ▼ How to Configure a Host for Local Files Mode

Use this procedure for configuring TCP/IP on a host that runs in local files mode.

1 **Assume the Primary Administrator role, or become superuser**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Change to the `/etc` directory.**

3 **Verify that the correct host name is set in the `/etc/nodename` file.**

When you specify the host name of a system during Solaris installation, that host name is entered into the `/etc/nodename` file. Make sure that the node name entry is the correct host name for the system.

4 **Verify that an `/etc/hostname.`*interface* file exists for each network interface on the system.**

For file syntax and basic information about the `/etc/hostname.`*interface* file, refer to "Basics for Administering Physical Interfaces" on page 146.

The Solaris installation program requires you to configure at least one interface during installation. The first interface that you configure automatically becomes the *primary network interface*. The installation program creates an `/etc/hostname.`*interface* file for the primary network interface and any other interfaces that you optionally configure at installation time.

If you configured additional interfaces during installation, verify that each interface has a corresponding `/etc/hostname.`*interface* file. You do not need to configure more than one interface during Solaris installation. However, if you later want to add more interfaces to the system, you must manually configure them.

For steps for manually configuring interfaces, refer to "Administering Interfaces in Solaris 10 3/05" on page 137 or "How to Configure a Physical Interface After System Installation" on page 150, for releases starting with Solaris 10 1/06.

5 **For Solaris 10 11/06 and earlier releases, verify that the entries in the `/etc/inet/ipnodes` file are current.**

The Solaris 10 installation program creates the `/etc/inet/ipnodes` file. This file contains the node name and IPv4 address, and IPv6 address, if appropriate, of every interface that is configured during installation.

Use the following format for entries in the `/etc/inet/ipnodes` file:

*IP-address node-name nicknames...*

*nicknames* are additional names by which an interface is known.

6 **Verify that the entries in the `/etc/inet/hosts` file are current.**

The Solaris installation program creates entries for the primary network interface, loopback address, and, if applicable, any additional interfaces that were configured during installation.

a. **Make sure that the existing entries in `/etc/inet/hosts` are current.**

b. **(Optional) Add the IP addresses and corresponding names for any network interfaces that were added to the local host after installation.**

c. **(Optional) Add the IP address or addresses of the file server, if the `/usr` file system is NFS mounted.**

7 **Type the host's fully qualified domain name in the `/etc/defaultdomain` file.**

For example, suppose host `tenere` was part of the domain `deserts.worldwide.com`. Therefore, you would type `deserts.worldwide.com` in `/etc/defaultdomain`. See "`/etc/defaultdomain` File" on page 237 for more information.

8 **Type the router's name in the `/etc/defaultrouter` file.**

See "`/etc/defaultrouter` File" on page 237 for information about this file.

9 **Type the name of the default router and its IP addresses in the `/etc/inet/hosts` file.**

Additional routing options are available, as discussed in "How to Configure Hosts for Network Client Mode" on page 104. You can apply these options to a local files mode configuration.

10 **Add the network mask for your network, if applicable:**

- If the host gets its IP address from a DHCP server, you do not have to specify the network mask.
- If you have set up a NIS server on the same network as this client, you can add `netmask` information into the appropriate database on the server.

- For all other conditions, do the following:

**a. Type the network number and the netmask in the `/etc/inet/netmasks` file.**

Use the following format:

```
network-number netmask
```

For example, for the Class C network number `192.168.83`, you would type:

```
192.168.83.0    255.255.255.0
```

For CIDR addresses, convert the network prefix into the equivalent dotted decimal representation. Network prefixes and their dotted decimal equivalents can be found in Table 2–3. For example, use the following to express the CIDR network prefix `192.168.3.0/22`.

```
192.168.3.0 255.255.252.0
```

**b. Change the lookup order for netmasks in `/etc/nsswitch.conf`, so that local files are searched first:**

```
netmasks:   files nis
```

**11 Reboot the system.**

## ▼ How to Set Up a Network Configuration Server

Information for setting up installation servers and boot servers is found in *Solaris Express Installation Guide: Basic Installations*.

**1 Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Change to the root (`/`) directory of the prospective network configuration server.**

**3 Turn on the `in.tftpd` daemon by creating the directory `/tftpboot`:**

```
# mkdir /tftpboot
```

This command configures the system as a TFTP, bootparams, and RARP server.

**4 Create a symbolic link to the directory.**

```
# ln -s /tftpboot/. /tftpboot/tftpboot
```

5 **Enable the `tftp` line in the `/etc/inetd.conf` file.**

Check that the entry reads as follows:

```
tftp dgram udp6 wait root /usr/sbin/in.tftpd in.tftpd -s /tftpboot
```

This line prevents `in.tftpd` from retrieving any file other than the files that are located in `/tftpboot`.

6 **Edit the `hosts` database.**

Add the host names and IP addresses for every client on the network.

7 **Edit the `ethers` database.**

Create entries for every host on the network that runs in network client mode.

8 **Edit the `bootparams` database.**

See "`bootparams` Database" on page 249. Use the wildcard entry or create an entry for every host that runs in network client mode.

9 **Convert the `/etc/inetd.conf` entry into a Service Management Facility (SMF) service manifest, and enable the resulting service:**

```
# /usr/sbin/inetconv
```

10 **Verify that `in.tftpd` is working correctly.**

```
# svcs network/tftp/udp6
```

You should receive output resembling the following:

```
STATE          STIME    FMRI
online         18:22:21 svc:/network/tftp/udp6:default
```

**More Information**

## Administering the `in.tftpd`Daemon

The `in.tftpd` daemon is managed by the Service Management Facility. Administrative actions on `in.tftpd`, such as enabling, disabling, or restarting, can be performed using the `svcadm` command. Responsibility for initiating and restarting this service is delegated to `inetd`. Use the `inetadm` command to make configuration changes and to view configuration information for `in.tftpd`. You can query the service's status by using the `svcs` command. For an overview of the Service Management Facility, refer to Chapter 15, "Managing Services (Overview)," in *System Administration Guide: Basic Administration*.

# Configuring Network Clients

Network clients receive their configuration information from network configuration servers. Therefore, before you configure a host as a network client you must ensure that at least one network configuration server is set up for the network.

## ▼ How to Configure Hosts for Network Client Mode

Do the following procedure on each host to be configured in network client mode.

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Search the `/etc` directory for the `nodename` file.**

If such a file exists, delete it.

Eliminating `/etc/nodename` causes the system to use the `hostconfig` program to obtain the host name, domain name, and router addresses from the network configuration server. See "Configuring Systems on the Local Network " on page 99.

3 **Create the `/etc/hostname.`*interface* file, if it does not exist.**

Ensure that the file is empty. An empty `/etc/hostname.`*interface* file causes the system to acquire the IPv4 address from the network configuration server.

4 **Ensure that the `/etc/inet/hosts` file contains only the `localhost` name and IP address of the loopback network interface.**

```
# cat /etc/inet/hosts
# Internet host table
#
127.0.0.1       localhost
```

The IPv4 loopback interface has the IP address `127.0.0.1`.

For more information, see "Loopback Address" on page 238. The file should not contain the IP address and host name for the local host (primary network interface).

5 **Check for the existence of an `/etc/defaultdomain` file.**

If such a file exists, delete it.

The `hostconfig` program automatically sets the domain name. To override the domain name that is set by `hostconfig`, type the substitute domain name in the `/etc/defaultdomain` file.

6 **Ensure that the search paths in the client's `/etc/nsswitch.conf` file reflect the name service requirements for your network.**

## How to Change the IPv4 Address and Other Network Configuration Parameters

This procedure explains how to modify the IPv4 address, host name, and other network parameters on a previously installed system. Use the procedure for modifying the IP address of a server or networked standalone system. The procedure does not apply to network clients or appliances. The steps create a configuration that persists across reboots.

---

**Note –** The instructions apply specifically to changing the IPv4 address of the primary network interface. To add another interface to the system, refer to "How to Configure a Physical Interface After System Installation" on page 150.

---

In almost all cases, the following steps use traditional IPv4 dotted decimal notation to specify the IPv4 address and subnet mask. Alternatively, you can use CIDR notation to specify the IPv4 address in all the applicable files in this procedure. For an introduction to CIDR notation, see "IPv4 Addresses in CIDR Format" on page 56.

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **For Solaris 10 11/06 and earlier releases only, modify the IP address in the `/etc/inet/ipnodes` file or equivalent `ipnodes` database.**

Use the following syntax for each IP address that you add to the system:

*IP-address host-name, nicknames*
*IP-address interface-name, nicknames*

The first entry should contain the IP address of the primary network interface and the host name of the system. You can optionally add nicknames for the host name. When you add additional physical interfaces to a system, create entries in `/etc/inet/ipnodes` for the IP addresses and associated names of those interfaces.

3 **If the system's host name must change, modify the host name entry in the `/etc/nodename` file.**

4 **Modify the IP address and, if applicable, the host name in the `/etc/inet/hosts` file or equivalent `hosts` database.**

**5 Modify the IP address in the `/etc/hostname.`*interface* file for the primary network interface.**

You can use any of the following as the entry for the primary network interface in the `/etc/hostname`*interface* file:

- IPv4 address, expressed in traditional dotted decimal format

  Use the following syntax:

  *IPv4 address*
  `(Optional)` *subnet mask*

  Here is an example:

  ```
  # vi hostname.eri0
  10.0.2.5
  netmask + 255.0.0.0
  ```

  The netmask entry is optional. If you do not specify it, the default netmask is assumed.

- IPv4 address, expressed in CIDR notation, if appropriate for your network configuration.

  *IPv4 address/network prefix*

  Here is an example:

  ```
  # vi hostname.eri0
  10.0.2.5/8
  ```

  The CIDR prefix designates the appropriate netmask for the IPv4 address. For example, the /8 above indicates the netmask 255.0.0.0.

- Host name.

  To use the system's host name in the `/etc/hostname.`*interface* file, be sure that the host name and associated IPv4 address are also in the `hosts` database.

**6 If the subnet mask has changed, modify the subnet entries in the following files:**

- `/etc/netmasks`
- (Optional) `/etc/hostname.`*interface*

**7 If the subnet address has changed, change the IP address of the default router in `/etc/defaultrouter` to that of the new subnet's default router.**

**8 Reboot the system.**

```
# reboot -- -r
```

**Example 5–1** Modifying the IPv4 Address and Other Network Parameters to Persist Across Reboots

This example shows how to change the following network parameters of a system that is moved to another subnet:

- IP address for the primary network interface `eri0` changes from `10.0.0.14` to `192.168.55.14`.
- Host name changes from `myhost` to `mynewhostname`.
- Netmask changes from `255.0.0.0` to `255.255.255.0`.
- Default router address changes to `192.168.55.200`.

Check the system's current status:

```
# hostname
myhost
# ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
eri0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.0.0.14 netmask ff000000 broadcast 10.255.255.255
        ether 8:0:20:c1:8b:c3
```

Next, change the system's host name and the IP address of `eri0` in the appropriate files:

```
# vi /etc/nodename
mynewhostname
In Solaris 10 11/06 and earlier Solaris 10 releases
# vi /etc/inet/ipnodes
192.168.55.14   mynewhostname      #moved system to 192.168.55 net
# vi /etc/inet/hosts
#
# Internet host table
#
127.0.0.1       localhost
192.168.55.14   mynewhostname        loghost
# vi /etc/hostname.eri0
192.168.55.14
netmask + 255.255.255.0
```

Finally, change the netmask and the IP address of the default router.

```
# vi /etc/netmasks.
.
.
192.168.55.0    255.255.255.0
# vi /etc/defaultrouter
```

```
192.168.55.200        #moved system to 192.168.55 net
#
```

After making these changes, reboot the system.

```
# reboot -- -r
```

Verify that the configuration you just set is maintained after the reboot:

```
# hostname
mynewhostname
# ifconfig -a

lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
eri0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.55.14 netmask ffffff00 broadcast 10.255.255.255
        ether 8:0:20:c1:8b:c3
```

**Example 5–2** Changing the IP Address and Host Name For the Current Session

This example shows how to change a host's name, IP address of the primary network interface, and subnet mask for the current session only. If you reboot, the system reverts to its previous IP address and subnet mask. The IP address for the primary network interface `eri0` changes from `10.0.0.14` to `192.168.34.100`.

```
# ifconfig -alo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
eri0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.0.0.14 netmask ff000000 broadcast 10.255.255.255
        ether 8:0:20:c1:8b:c3
# ifconfig eri0 192.168.34.100 netmask 255.255.255.0 broadcast + up
# vi /etc/nodename
mynewhostname

# ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
eri0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.34.100 netmask ffffff00 broadcast 10.255.255.255
        ether 8:0:20:c1:8b:c3
# hostname
mynewhostname
```

**Example 5–3** Changing the IPv4 Address for the Current Session, Using CIDR Notation

This example shows how to change a host name and IP address for the current session only, using CIDR notation. If you reboot, the system reverts to its previous IP address and subnet mask. The IP address for the primary network interface, `eri0`, changes from `10.0.0.14` to `192.168.6.25/27`.

```
# ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
eri0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.0.0.14 netmask ff000000 broadcast 10.255.255.255
        ether 8:0:20:c1:8b:c3
# ifconfig eri0 192.168.6.25/27 broadcast + up
# vi /etc/nodename
mynewhostname
# ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
eri0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.06.25 netmask ffffffe0 broadcast 10.255.255.255
        ether 8:0:20:c1:8b:c3
# hostname
mynewhostname
```

When you use CIDR notation for the IPv4 address, you do not have to specify the netmask. `ifconfig` uses the network prefix designation to determine the netmask. For example, for the `192.168.6.0/27` network, `ifconfig` sets the netmask `ffffffe0`. If you had used the more common `/24` prefix designation, the resulting netmask is `ffffff00`. Using the `/24` prefix designation is the equivalent of specifying the netmask `255.255.255.0` to `ifconfig` when configuring a new IP address.

**See Also** To change the IP address of an interface other than the primary network interface, refer to *System Administration Guide: Basic Administration* and "How to Configure a Physical Interface After System Installation" on page 150.

# Packet Forwarding and Routing on IPv4 Networks

This section contains procedures and examples that show how to configure forwarding and routing for routers and hosts on IPv4 networks.

*Packet forwarding* is the basic method for sharing information across systems on a network. Packets are transferred between a source interface and a destination interface, usually on two different systems. When you issue a command or send a message to a nonlocal interface, your system forwards those packets onto the local network. The interface with the destination IP

address that is specified in the packet headers then retrieves the packets from the local network. If the destination address is not on the local network, the packets are then forwarded to the next adjacent network, or *hop*. By default, packet forwarding is automatically configured when you install the Solaris OS.

*Routing* is the process by which systems decide where to send a packet. Routing protocols on a system "discover" the other systems on the local network. When the source system and the destination system are on the same local network, the path that packets travel between them is called a *direct route*. If a packet must travel at least one hop beyond its source system, the path between the source system and destination system is called an *indirect route*. The routing protocols learn the path to a destination interface and retain data about known routes in the system's *routing table*.

*Routers* are specially configured systems with multiple physical interfaces that connect the router to more than one local network. Therefore, the router can forward packets beyond the home LAN, regardless of whether the router runs a routing protocol. For more information about how routers forward packets, refer to "Planning for Routers on Your Network" on page 65.

*Routing protocols* handle routing activity on a system and, by exchanging routing information with other hosts, maintain known routes to remote networks. Both routers and hosts can run routing protocols. The routing protocols on the host communicate with routing daemons on other routers and hosts. These protocols assist the host in determining where to forward packets. When network interfaces are enabled, the system automatically communicates with the routing daemons. These daemons monitor routers on the network and advertise the routers' addresses to the hosts on the local network. Some routing protocols, though not all, also maintain statistics that you can use to measure routing performance. Unlike packet forwarding, you must explicitly configure routing on a Solaris system.

This section contains tasks for administering packet forwarding and routing on IPv4 routers and hosts. For information about routing on an IPv6-enabled network, refer to "Configuring an IPv6 Router" on page 176.

## Routing Protocols Supported by the Solaris OS

Routing protocols are classified as interior gateway protocols (IGPs), exterior gateway protocols (EGPs), or a combination of both. *Interior gateway protocols* exchange routing information between routers on networks under common administrative control. In the network topology shown in Figure 5–3, the routers run an IGP for exchanging routing information. *Exterior gateway protocols* enable the router that connects the local internetwork to an external network to exchange information with another router on the external network. For example, the router that connects a corporate network to an ISP runs an EGP to exchange routing information with its router counterpart at the ISP. Border Gateway Protocol (BGP) is a popular EGP that is used for carrying routing information between different organizations and IGPs.

The following table provides information about the Solaris routing protocols and the location of each protocol's associated documentation.

**TABLE 5–1** Solaris Routing Protocols

| Protocol | Associated Daemon | Description | For Instructions |
|---|---|---|---|
| Routing Information Protocol (RIP) | `in.routed` | IGP that routes IPv4 packets and maintains a routing table | "How to Configure an IPv4 Router" on page 116 |
| Internet Control Message Protocol (ICMP) Router Discovery | `in.routed` | Used by hosts to discover the presence of a router on the network | "How to Enable Static Routing on a Single-Interface Host" on page 128 and "How to Enable Dynamic Routing on a Single-Interface Host" on page 130 |
| Routing Information Protocol, next generation (RIPng) Protocol | `in.ripngd` | IGP that routes IPv6 packets and maintains a routing table | "How to Configure an IPv6-Enabled Router" on page 177 |
| Neighbor Discovery (ND) Protocol | `in.ndpd` | Advertises the presence of an IPv6 router and discovers the presence of IPv6 hosts on a network | "Configuring an IPv6 Interface" on page 171 |

The Solaris 10 OS also supports the Open Source Quagga routing protocol suite. These protocols are available from the SFW consolidation disk, though they are not part of the main Solaris distribution. The following table lists the Quagga protocols:

**TABLE 5–2** OpenSolaris Quagga Protocols

| Protocol | Daemon | Description |
|---|---|---|
| RIP protocol | `ripd` | IPv4 distance vectoring IGP that routes IPv4 packets and advertises its routing table to neighbors. |
| RIPng | `ripngd` | IPv6 distance vectoring IGP. Routes IPv6 packets and maintains a routing table. |
| Open Shortest Path First (OSPF) protocol | `ospfd` | IPv4 link state IGP for packet routing and high availability networking |
| Border Gateway Protocol (BGP) | `bgpd` | IPv4 and IPv6 EGP for routing across administrative domains. |

The following figure shows an autonomous system that uses the Quagga routing protocols:

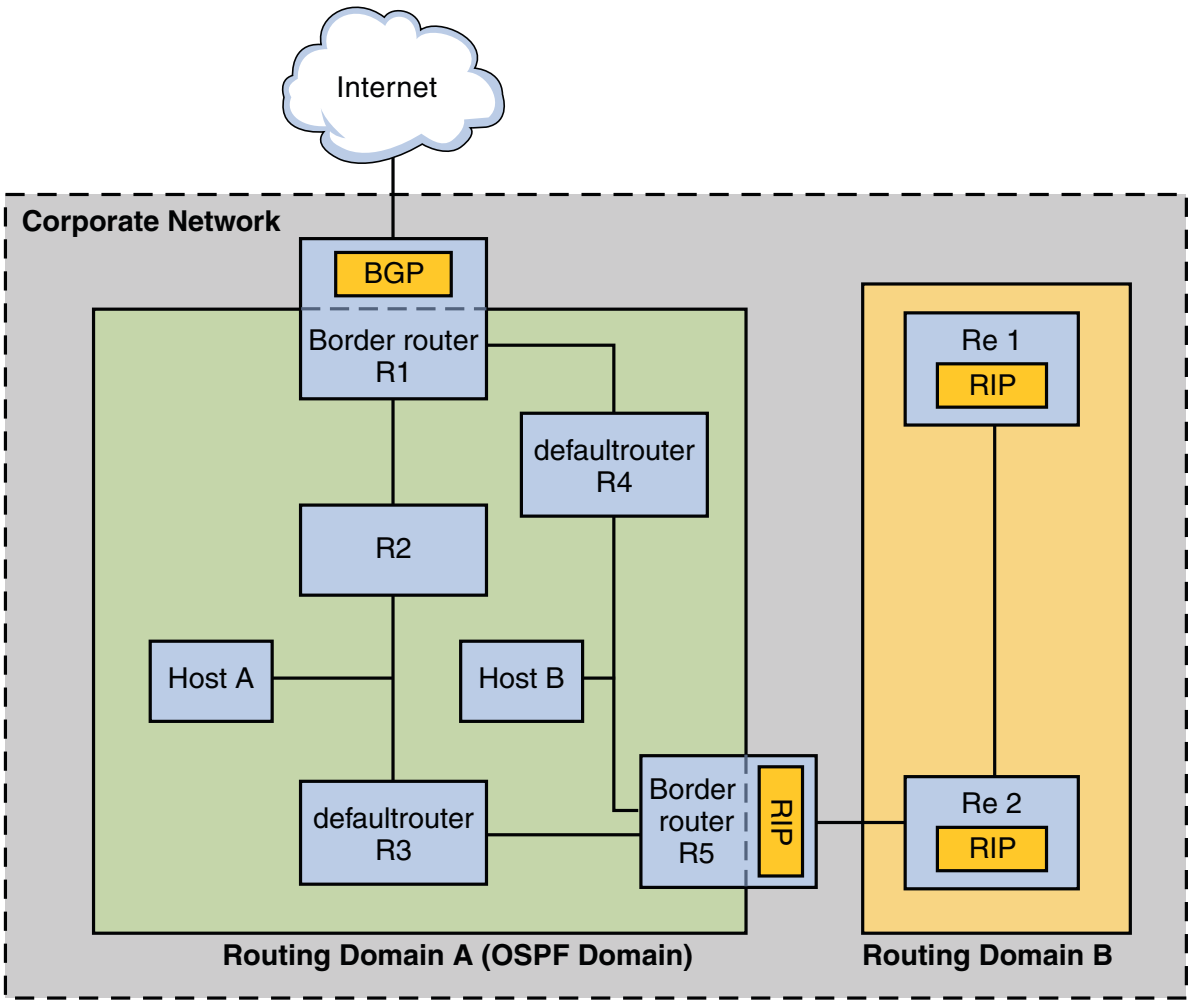



**FIGURE 5–2** Corporate Network That Runs Quagga Protocols

The figure shows a corporate network autonomous system that is subdivided into two routing domains, A and B. A *routing domain* is an internetwork with a cohesive routing policy, either for administrative purposes or because the domain uses a single routing protocol. Both domains in the figure run routing protocols from the Quagga protocol suite.

Routing Domain A is an OSPF domain, which is administered under a single OSPF domain ID. All systems within this domain run OSPF as their interior gateway protocol. In addition to internal hosts and routers, Domain A includes two border routers.

Border router R1 connects the Corporate Network to an ISP and ultimately the Internet. To facilitate communications between the Corporate Network and the outside world, R1 runs BGP

over its externally facing network interface. The border router R5 connects Domain A with Domain B. All systems on Domain B are administered with RIP as their interior gateway protocol. Therefore, border router R5 must run OSPF on the Domain A facing interface and RIP on the Domain B facing interface.

For more information on the Quagga protocols, refer to the Open Solaris Quagga (`http://opensolaris.org/os/project/quagga/`). For configuration procedures for these protocols, go to the Open Source Quagga (`http://www.quagga.net`) web pages.

## IPv4 Autonomous System Topology

Sites with multiple routers and networks typically administer their network topology as a single routing domain, or *autonomous system (AS)* . The following figure shows a typical network topology that would be considered a small AS. This topology is referenced in the examples throughout this section.

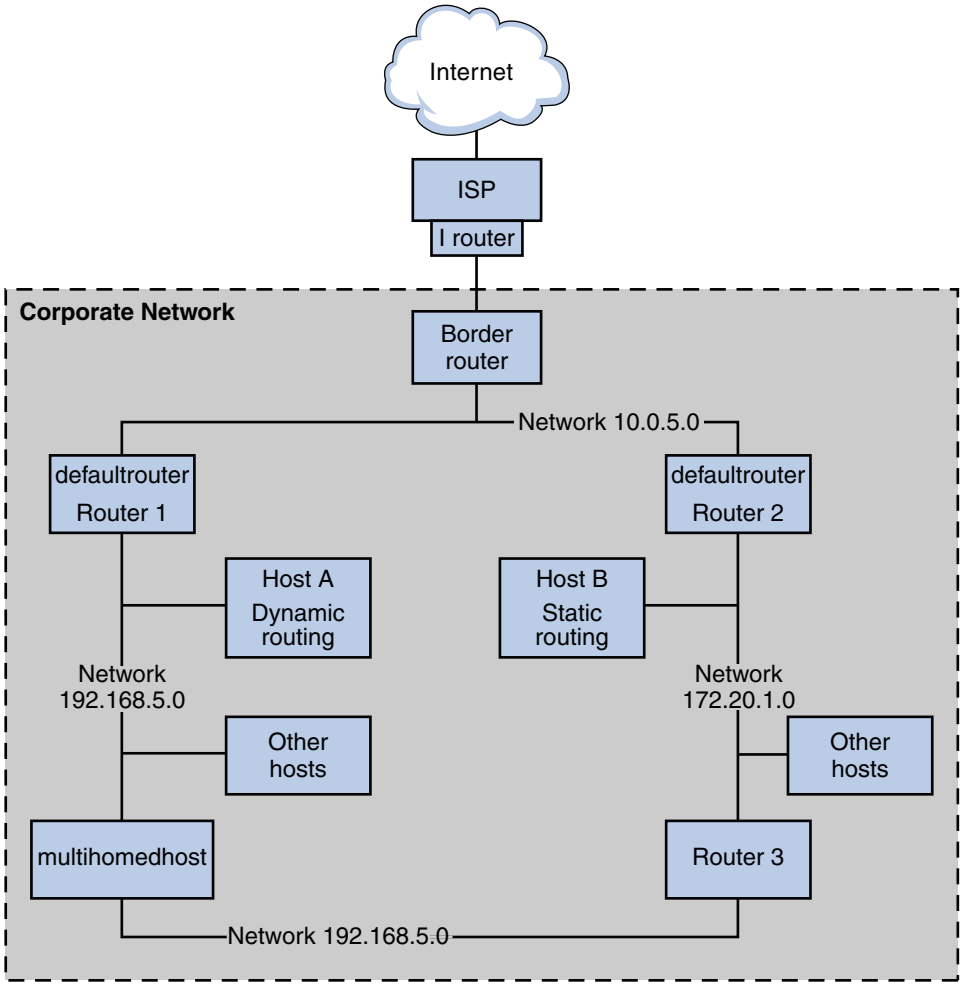



**FIGURE 5–3** Autonomous System With Multiple IPv4 Routers

The figure shows an AS that is divided into three local networks, `10.0.5.0`, `172.20.1.0`, and `192.168.5`. Four routers share packet-forwarding and routing responsibilities. The AS includes the following types of systems:

- *Border routers* connect an AS to an external network, such as the Internet. Border routers interconnect with networks external to the IGP running on the local AS. A border router can run an EGP, such as Border Gateway Protocol (BGP), to exchange information with external routers, for example, the routers at the ISP. In Figure 5–3, the border router's interfaces connect to internal network `10.0.5.0` and to a high-speed router to a service provider.

For information on configuring a border router, refer to the Open Source Quagga documentation (`http://www.quagga.net/docs/docs-info.php#SEC72`)for BGP information.

If you plan to use BGP to connect your AS to the Internet, you should obtain an autonomous system number (ASN) from the Internet Registry for your locale. Regional registries, such as the American Registry for Internet Numbers (ARIN), offer guidelines on how to obtain an ASN. For example, the ARIN Number Resource Policy Manual (`http://www.arin.net/policy/nrpm.html#five`) contains instructions for getting an ASN for autonomous systems in the United States and Canada. Alternatively, your ISP might be able to obtain an ASN for you.

- *Default routers* maintain routing information about all the systems on the local network. These routers typically run IGPs such as RIP. In Figure 5–3, Router 1s interfaces are connected to internal network `10.0.5.0` and internal network `192.168.5`. Router 1 also serves as the default router for `192.168.5`. Router 1 maintains routing information for all systems on `192.168.5` and routes to other routers, such as the border router. Router 2s interfaces connect to internal network `10.0.5.0` and internal network `172.20.1`.

  For an example of configuring a default router, refer to Example 5–4.

- *Packet-forwarding routers* forward packets but do not run routing protocols. This type of router receives packets from one of its interfaces that is connected to a single network. These packets are then forwarded through another interface on the router to another local network. In Figure 5–3, Router 3 is a packet-forwarding router with connections to networks `172.20.1` and `192.168.5`.

- *Multihomed hosts* have two or more interfaces that are connected to the same network segment. A multihomed host can forward packets, which is the default for all systems that run the Solaris OS. Figure 5–3 shows a multihomed host with both interfaces connected to network `192.168.5`. For an example of configuring a multihomed host, refer to Example 5–6.

- *Single interface hosts* rely on the local routers, not only for packet forwarding but also for receiving valuable configuration information. Figure 5–3 includes Host A on the `192.168.5` network, which implements dynamic routing, and Host B on the `172.20.1` network, which implements static routing. To configure a host to run dynamic routing, refer to "How to Enable Dynamic Routing on a Single-Interface Host" on page 130. To configure a host to run static routing, refer to "How to Enable Static Routing on a Single-Interface Host" on page 128.

# Configuring an IPv4 Router

This section contains a procedure and example for configuring an IPv4 router. To configure an IPv6-enabled router, refer to "How to Configure an IPv6-Enabled Router" on page 177.

Because a router provides the interface between two or more networks, you must assign a unique name and IP address to each of the router's physical network interfaces. Thus, each

router has a host name and an IP address that are associated with its primary network interface, in addition to a minimum of one more unique name and IP address for each additional network interface.

You can also use the following procedure to configure a system with only one physical interface (by default, a host) to be a router. You might configure a single interface system as a router if the system serves as one endpoint on a PPP link, as explained in "Planning a Dial-up PPP Link" in *System Administration Guide: Network Services*.

---

**Note –** You can configure all interfaces of a router during Solaris system installation. For instructions, see *Solaris Express Installation Guide: Basic Installations*.

---

## ▼ How to Configure an IPv4 Router

The following instructions assume that you are configuring interfaces for the router after installation.

**Before You Begin** After the router is physically installed on the network, configure the router to operate in local files mode, as described in "How to Configure a Host for Local Files Mode" on page 100. This configuration ensures that routers boot if the network configuration server is down.

**1 On the system to be configured as a router, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Starting in the Solaris 10 1/06 release, use the `dladm show-link` command to determine which interfaces are physically installed on the router.**

```
# dladm show-link
```

The following example output from `dladm show-link` indicates that a `qfe` NIC with four interfaces and two `bge` interfaces are physically available on the system.

```
qfe0            type: legacy    mtu: 1500       device: qfe0
qfe1            type: legacy    mtu: 1500       device: qfe1
qfe2            type: legacy    mtu: 1500       device: qfe0
qfe3            type: legacy    mtu: 1500       device: qfe1
bge0            type: non-vlan  mtu: 1500       device: bge0
bge1            type: non-vlan  mtu: 1500       device: bge1
```

**3 Review which interfaces on the router were configured and plumbed during installation.**

```
# ifconfig -a
```

The following example output from `ifconfig -a` shows that the interface `qfe0` was configured during installation. This interface is on the `172.16.0.0` network. The remaining interfaces on the `qfe` NIC, `qfe1` - `qfe3`, and the `bge` interfaces have not been configured.

```
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 172.16.26.232 netmask ffff0000 broadcast 172.16.26.255
        ether 0:3:ba:11:b1:15
```

**4 Configure and plumb another interface.**

```
# ifconfig interface plumb up
```

For example, for `qfe1`, you would type:

```
# ifconfig qfe1 plumb up
```

---

**Note –** Interfaces that are explicitly configured with the `ifconfig` command do not persist across reboots.

---

**5 Assign an IPv4 address and a netmask to the interface.**

---

**Caution –** You can configure an IPv4 routers to receive its IP address through DHCP, but this is recommended only for very experienced DHCP system administrators.

---

```
# ifconfig interface IPv4-address netmask+netmask
```

For example, to assign the IP address `192.168.84.3` to `qfe1`, do either of the following:

- Using traditional IPv4 notation, type the following:

  ```
  # ifconfig qfe1 192.168.84.3 netmask + 255.255.255.0
  ```

- Using CIDR notation, type the following:

  ```
  # ifconfig qfe1 192.168.84.3/24
  ```

  The prefix `/24` automatically assigns the `255.255.255.0` netmask to `qfe1`. For a table of CIDR prefixes and their dotted-decimal netmask equivalents, refer to Figure 2–2.

**6 (Optional) To ensure that the interface configuration persists across reboots, create an `/etc/hostname.`*interface* file for each additional physical interface .**

For example, you would create the `/etc/hostname.qfe1` and `/etc/hostname.qfe2` files. Then you would type the host name `timbuktu` in `/etc/hostname.qfe1` file and host name

`timbuktu-201` in `/etc/hostname.qfe1`. For more information about configuring single interfaces, refer to "How to Configure a Physical Interface After System Installation" on page 150.

Be sure to do a configuration reboot after creating this file:

```
# reboot -- -r
```

**7 Add the host name and IP address of each interface to the `/etc/inet/hosts` file.**

For example:

```
172.16.26.232      deadsea        #interface for network 172.16.0.0
192.168.200.20     timbuktu       #interface for network 192.168.200
192.168.201.20     timbuktu-201   #interface for network 192.168.201
192.168.200.9      gobi
192.168.200.10     mojave
192.168.200.110    saltlake
192.168.200.12     chilean
```

The interfaces `timbuktu` and `timbuktu-201` are on the same system. Notice that the network address for `timbuktu-201` is different from the network interface for `timbuktu`. The difference exists because the physical network media for network `192.168.201` is connected to the `timbuktu-201` network interface while the media for network `192.168.200` is connected to the `timbuktu` interface.

**8 For Solaris 10 11/06 and earlier releases of Solaris 10 only, add the IP address and host name of each new interface into the `/etc/inet/ipnodes` file or equivalent `ipnodes` database.**

For example:

```
vi /etc/inet/ipnodes
172.16.26.232      deadsea        #interface for network 172.16.0.0
192.168.200.20     timbuktu       #interface for network 192.168.200
192.168.201.20     timbuktu-201   #interface for network 192.168.201
```

**9 If the router is connected to any subnetted network, add the network number and the netmask to the `/etc/inet/netmasks` file.**

- For traditional IPv4 address notation, such as `192.168.83.0`, you would type:

  ```
  192.168.83.0    255.255.255.0
  ```

- For CIDR addresses, use the dotted-decimal version of the prefix in the entry in the `/etc/inet/netmask` file. Network prefixes and their dotted-decimal equivalents can be found in Figure 2–2. For example, you would use the following entry in `/etc/netmasks` to express the CIDR network prefix `192.168.3.0/22`:

  ```
  192.168.3.0 255.255.252.0
  ```

**10 Enable IPv4 packet forwarding on the router.**

Use either of the following commands to enable packet forwarding:

- Use the `routeadm` command, as follows:

```
# routeadm -e ipv4-forwarding -u
```

- Use the following service management facility (SMF) command:

```
# svcadm enable ipv4-forwarding
```

At this point, the router can forward packets beyond the local network. The router also supports *static routing*, a process where you can manually add routes to the routing table. If you plan to use static routing on this system, then router configuration is complete. However, you need to maintain routes in the system routing table. For information on adding routes, see "Configuring Routes" on page 122 and the `route`(1M) man page.

**11 (Optional) Start a routing protocol.**

The routing daemon `/usr/sbin/in.routed` automatically updates the routing table, a process that is known as *dynamic routing*. Turn on the default IPv4 routing protocols in either of the following ways:

- Use the `routeadm` command, as follows:

```
# routeadm -e ipv4-routing -u
```

- Use the following SMF command to start a routing protocol such as RIP.

```
# svcadm enable route:default
```

  The SMF FMRI associated with the `in.routed` daemon is `svc:/network/routing/route`.

For information about the `routeadm` command, see the `routeadm`(1M) man page.

**Example 5–4** Configuring the Default Router for a Network

This example shows how to upgrade a system with more than one interface to become a default router. The goal is to make Router 2, which is shown in Figure 5–3, the default router for network `172.20.1.0`. Router 2 contains two wired network connections, one connection to network `172.20.1.0` and one to network `10.0.5.0`. The example assumes that the router operates in local files mode, as described in "How to Configure a Host for Local Files Mode" on page 100.

After becoming superuser or assuming an equivalent role, you would determine out the status of the system's interfaces.Starting with Solaris 10 1/06, you can use the `dladm` command as follows:

```
# dladm show-link
ce0             type: legacy    mtu: 1500       device: ce0
bge0            type: non-vlan  mtu: 1500       device: bge0
bge1            type: non-vlan  mtu: 1500       device: bge1

# ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
ce0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 172.20.1.10 netmask ffff0000 broadcast 172.20.10.100
        ether 8:0:20:c1:1b:c6
```

The output of `dladm show-link` indicates that three links are available on the system. Only the `ce0` interface has been plumbed. You would begin default router configuration by physically connecting the `bge0` interface to the `10.0.5.0` network. Then, you would plumb the interface and make it persist across reboots.

```
# ifconfig bge0 plumb up
# ifconfig bge0 10.0.5.10
# ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
ce0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 172.20.1.10 netmask ffff0000 broadcast 172.255.255.255
        ether 8:0:20:c1:1b:c6
bge0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.0.5.10 netmask ff000000 broadcast 10.255.255.255
        ether 8:0:20:e5:95:c4
 # vi /etc/hostname.bge0
10.0.5.10
255.0.0.0
```

Reboot the system, using the reconfiguration boot command:

```
# reboot -- -r
```

Continue by configuring the following network databases with information about the newly plumbed interface and the network to which it is connected:

```
# vi /etc/inet/hosts
127.0.0.1       localhost
172.20.1.10        router2        #interface for network 172.20.1
10.0.5.10          router2-out    #interface for network 10.0.5
# vi /etc/inet/netmasks
172.20.1.0    255.255.0.0
10.0.5.0      255.0.0.0
```

Finally, use SMF to enable packet forwarding and then enable the `in.routed` routing daemon.

```
# svcadm enable ipv4-forwarding
# svcadm enable route:default
```

Now IPv4 packet forwarding and dynamic routing through RIP are enabled on Router 2. However, the default router configuration for network `172.20.1.0` is not yet complete. You would need to do the following:

- Modify each host on `172.10.1.10` so that the host gets its routing information from the new default router. For more information, refer to "How to Enable Static Routing on a Single-Interface Host" on page 128.
- Define a static route to the border router in the routing table of Router 2. For more details, refer to " Routing Tables and Routing Types" on page 121.

# Routing Tables and Routing Types

Both routers and hosts maintain a *routing table*. The routing daemon on each system updates the table with all known routes. The system's kernel reads the routing table before forwarding packets to the local network. The routing table lists the IP addresses of networks that the system knows about, including the system's local, default network. The table also lists the IP address of a gateway system for each known network. The *gateway* is a system that can receive outgoing packets and forward them one hop beyond the local network. The following is a simple routing table for a system on an IPv4-only network:

```
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
default              172.20.1.10          UG        1    532   ce0
224.0.0.0            10.0.5.100           U         1      0   bge0
10.0.0.0             10.0.5.100           U         1      0   bge0
127.0.0.1            127.0.0.1            UH        1     57   lo0
```

You can configure two types of routing on a Solaris system: static and dynamic. You can configure either or both routing types on a single system. A system that implements *dynamic routing* relies on routing protocols, such as RIP for IPv4 networks, and RIPng for IPv6 networks, to maintain its routing tables. A system that runs only *static routing* does not rely on a routing protocol for routing information and for updating the routing table. Instead, you must maintain the system's known routes manually through the `route` command. For complete details, refer to the `route`(1M) man page.

When you configure routing for the local network or autonomous system, consider which type of routing to support on particular routers and hosts.

| Routing Type | Best Used on |
|---|---|
| Static | Small networks, hosts that get their routes from a default router, and default routers that only need to know about one or two routers on the next few hops. |
| Dynamic | Larger internetworks, routers on local networks with many hosts, and hosts on large autonomous systems. Dynamic routing is the best choice for systems on most networks. |
| Combined static and dynamic | Routers that connect a statically routed network and a dynamically routed network, and border routers that connect an interior autonomous system with external networks. Combining both static and dynamic routing on a system is a common practice. |

The AS that is shown is Figure 5–3 combines both static and dynamic routing.

## Configuring Routes

To implement dynamic routing for an IPv4 network, use the `routeadm` or `svcadm` command to start the `in.routed` routing daemon. For instructions, see "How to Configure an IPv4 Router" on page 116. Dynamic routing is the preferred strategy for most networks and autonomous systems. However, your network topology or a particular system on your network might require static routing. In that case, you must manually edit the system routing table to reflect the known route to the gateway. The next procedure shows how to add a static route.

---

**Note –** Two routes to the same destination does not automatically cause the system to do load balancing or failover. If you need these capabilities, use IPMP, as explained in Chapter 30, "Introducing IPMP (Overview)."

---

## ▼ How to Add a Static Route to the Routing Table

**1 View the current state of the routing table.**

Use your regular user account to run the following form of the `netstat` command:

```
% netstat -rn
```

Your output would resemble the following:

```
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
192.168.5.125        192.168.5.10          U      1   5879   ipge0
224.0.0.0            198.168.5.10          U      1  0       ipge0
default              192.168.5.10          UG     1  91908
127.0.0.1            127.0.0.1             UH     1  811302   lo0
```

**2 Assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**3 (Optional) Flush the existing entries in the routing table.**

```
# route flush
```

**4 Add a route that persists across system reboots.**

**# route -p add -net** *network-address* **-gateway** *gateway-address*

| | |
|---|---|
| `-p` | Creates a route that must persist across system reboots. If you want the route to prevail only for the current session, do not use the `-p` option. |
| `add` | Indicates that you are about to add the following route. |
| `-net` *network-address* | Specifies that the route goes to the network with the address in *network-address*. |
| `-gateway` *gateway-address* | Indicates that the gateway system for the specified route has the IP address *gateway-address*. |

**Example 5–5** Adding a Static Route to the Routing Table

The following example shows how to add a static route to a system. The system is Router 2, the default router for the `172.20.1.0` network that is shown in Figure 5–3. In Example 5–4, Router 2 is configured for dynamic routing. To better serve as the default router for the hosts on network `172.20.1.0`, Router 2 additionally needs a static route to the AS's border router, `10.0.5.150`.

To view the routing table on Router 2, you would do the following:

```
# netstat -rn
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
default              172.20.1.10          UG        1    249 ce0
224.0.0.0            172.20.1.10          U         1      0 ce0
10.0.5.0             10.0.5.20            U         1     78 bge0
127.0.0.1            127.0.0.1            UH        1     57 lo0
```

The routing table indicates two routes that Router 2 knows about. The default route uses Router 2's `172.20.1.10` interface as its gateway. The second route, `10.0.5.0`, was discovered by the `in.routed` daemon running on Router 2. The gateway for this route is Router 1, with the IP address `10.0.5.20`.

To add a second route to network `10.0.5.0`, which has its gateway as the border router, you would do the following:

```
# route -p add -net 10.0.5.0/24 -gateway 10.0.5.150/24
add net 10.0.5.0: gateway 10.0.5.150
```

Now the routing table has a route for the border router, which has the IP address `10.0.5.150/24`.

```
# netstat -rn
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
default              172.20.1.10          UG        1    249 ce0
224.0.0.0            172.20.1.10          U         1      0 ce0
10.0.5.0             10.0.5.20            U         1     78 bge0
10.0.5.0             10.0.5.150           U         1    375 bge0
127.0.0.1            127.0.0.1            UH        1     57 lo0
```

# Configuring Multihomed Hosts

In the Solaris OS, a system with more than one interface is considered a *multihomed host*. A multihomed host does not forward IP packets. However, you can configure a multihomed host to run routing protocols. You typically configure the following types of systems as multihomed hosts:

- NFS servers, particularly those servers that function as large data centers, can be attached to more than one network in order to share files among a large pool of users. These servers do not need to maintain routing tables.
- Database servers can have multiple network interfaces to provide resources to a large pool of users, just like NFS servers.
- Firewall gateways are systems that provide the connection between a company's network and public networks such as the Internet. Administrators set up firewalls as a security measure. When configured as a firewall, the host does not pass packets between the networks that are attached to the host's interfaces. However, the host can still provide standard TCP/IP services, such as `ssh` to authorized users.

---

**Note –** When multihomed hosts have different types of firewalls on any of their interfaces, take care to avoid unintentional disruption of the host's packets. This problem arises particularly with stateful firewalls. One solution might be to configure stateless firewalling. For more information about firewalls, refer to “Firewall Systems” in *System Administration Guide: Security Services* or the documentation for your third-party firewall.

---

## How to Create a Multihomed Host

1. **On the prospective multihomed host, assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Configure and plumb each additional network interface that was not configured as part of the Solaris OS installation.**

   Refer to "How to Configure a Physical Interface After System Installation" on page 150.

3. **Verify that IP forwarding is not enabled on the multihomed host.**

   ```
   # routeadm
   ```

   The `routeadm` command without options reports the state of the routing daemons. The following output from `routeadm` shows that IPv4 forwarding is enabled:

   ```
      Configuration   Current             Current
                        Option   Configuration        System State
   ---------------------------------------------------------------
                   IPv4 routing   disabled             disabled
                   IPv6 routing   disabled             disabled
                 IPv4 forwarding  enabled              disabled
                 IPv6 forwarding  disabled             disabled

                 Routing services   "route:default ripng:default"
   ```

4. **Turn off packet forwarding, if it is enabled on the system.**

   Use either of the following commands:

   - For the `routeadm` command, type the following:

     ```
     # routeadm -d ipv4-forwarding -u
     ```

   - To use SMF, type the following:

     ```
     # svcadm disable ipv4-forwarding
     ```

5. **(Optional) Turn on dynamic routing for the multihomed host.**

   Use either of the following commands to enable the `in.routed` daemon:

   - For the `routeadm` command, type the following:

```
# routeadm -e ipv4-routing -u
```

- To use SMF, type the following:

```
#svcadm enable route:default
```

### Example 5–6 Configuring a Multihomed Host

The following example shows how to configure the multihomed host that is shown in Figure 5–3. In the example, the system has the host name `hostc`. This host has two interfaces, which are both connected to network `192.168.5.0` .

To begin, you would display the status of the system's interfaces.

```
# dladm show-link

hme0            type: legacy    mtu: 1500       device: hme0
qfe0            type: legacy    mtu: 1500       device: qfe0
qfe1            type: legacy    mtu: 1500       device: qfe1
qfe2            type: legacy    mtu: 1500       device: qfe2
qfe3            type: legacy    mtu: 1500       device: qfe3
# ifconfig -a
# ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
hme0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
      inet 192.168.5.82 netmask ff000000 broadcast 192.255.255.255
      ether 8:0:20:c1:1b:c6
```

The `dladm show-link` command reports that `hostc` has two interfaces with a total of five possible links. However, only `hme0` has been plumbed. To configure `hostc` as a multihomed host, you must add `qfe0` or another link on the `qfe` NIC. First, you would physically connect the `qfe0` interface to the `192.168.5.0` network. Then you would plumb the `qfe0` interface, and make the interface persist across reboots.

```
# ifconfig qf0 plumb up
# ifconfig qfe0 192.168.5.85
# ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
hme0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.5.82 netmask ff0000 broadcast 192.255.255.255
        ether 8:0:20:c1:1b:c6
qfe0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.5.85 netmask ff000000 broadcast 192.255.255.255
        ether 8:0:20:e1:3b:c4
```

```
 # vi /etc/hostname.qfe0
192.168.5.85
255.0.0.0
```

Reboot the system, using the reconfiguration command:

```
# reboot -- -r
```

Next, you would add the `qfe0` interface to the `hosts` database:

```
# vi /etc/inet/hosts
127.0.0.1           localhost
192.168.5.82        host3    #primary network interface for host3
192.168.5.85        host3-2  #second interface
```

Then, you would check the state of packet forwarding and routing on `host3`:

```
# routeadm
              Configuration   Current             Current
                     Option   Configuration       System State
---------------------------------------------------------------
                IPv4 routing   enabled             enabled
                IPv6 routing   disabled            disabled
              IPv4 forwarding  enabled             enabled
              IPv6 forwarding  disabled            disabled

            Routing services   "route:default ripng:default"
```

The `routeadm` command reports that dynamic routing through the `in.routed` daemon and packet forwarding are currently enabled. However, you would need to disable packet forwarding:

```
# svcadm disable ipv4-forwarding
```

You can also use the `routeadm` commands as shown in "How to Create a Multihomed Host" on page 125 to turn off packet forwarding. When packet forwarding is disabled, `host3` becomes a multihomed host.

# Configuring Routing for Single-Interface Systems

Single-interface hosts need to implement some form of routing. If the host is to obtain its routes from one or more local default routers, then you must configure the host to use static routing. Otherwise, dynamic routing is recommended for the host. The following procedures contain the instructions for enabling both routing types.

## How to Enable Static Routing on a Single-Interface Host

This procedure enables static routing on a single-interface host. Hosts that use static routing do not run a dynamic routing protocol such as RIP. Instead, the host must rely on the services of a default router for routing information. The figure "IPv4 Autonomous System Topology" on page 113 shows several default routers and their client hosts. If you supplied the name of a default router when you installed a particular host, that host is already configured to use static routing.

---

**Note –** You can also use the following procedure to configure static routing on a multihomed host.

---

For information about the `/etc/defaultrouter` file, see "`/etc/defaultrouter` File" on page 237. For information about static routing and the routing table, refer to " Routing Tables and Routing Types" on page 121.

1. **On the single interface host, assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Verify whether the `/etc/defaultrouter` file is present on the host.**

   ```
   # cd /etc
   # ls | grep defaultrouter
   ```

3. **Open a text editor to create or modify the `/etc/defaultrouter` file**

4. **Add an entry for the default router.**

   ```
   # vi  /etc/defaultrouter
   router-IP
   ```

   where *router-IP* indicates the IP address of the default router for the host to use.

5. **Verify that routing and packet forwarding are not running on the host.**

   ```
   # routeadm
      Configuration   Current              Current
                        Option   Configuration        System State
   ---------------------------------------------------------------
                  IPv4 routing   disabled             disabled
                  IPv6 routing   disabled             disabled
              IPv4 forwarding   disabled            disabled
              IPv6 forwarding   disabled             disabled
   ```

```
                Routing services   "route:default ripng:default"
```

6 **Add an entry for the default router in the local `/etc/inet/hosts` file.**

For information about configuring `/etc/inet/hosts`, refer to "How to Change the IPv4 Address and Other Network Configuration Parameters" on page 105.

**Example 5–7** Configuring a Default Router and Static Routing for a Single-Interface Host

The following example shows how to configure static routing for `hostb`, a single-interface host on the network `172.20.1.0` that is shown in Figure 5–3. `hostb` needs to use Router 2 as its default router.

First, you would log in to `hostb` as superuser, or assume an equivalent role. Then, you would determine whether the `/etc/defaultrouter` file is present on the host:

```
# cd /etc
# ls | grep defaultrouter
```

No response from `grep` indicates that you need to create the `/etc/defaultrouter` file.

```
# vi /etc/defaultrouter
172.20.1.10
```

The entry in the `/etc/defaultrouter` file is the IP address of the interface on Router 2, which is attached to the `172.20.1.0` network. Next, you verify whether the host currently enables packet forwarding or routing.

```
# routeadm
   Configuration   Current              Current
                     Option   Configuration        System State
---------------------------------------------------------------
                IPv4 routing   disabled             disabled
                IPv6 routing   disabled             disabled
              IPv4 forwarding  enabled              enabled
              IPv6 forwarding  disabled             disabled

             Routing services   "route:default ripng:default"
```

Packet forwarding is enabled for this particular host. You would turn it off as follows:

```
# svcadm disable ipv4-forwarding
```

Lastly, you would make sure that the host's `/etc/inet/hosts` file has an entry for the new default router.

```
# vi /etc/inet/hosts
127.0.0.1           localhost
172.20.1.18         host2    #primary network interface for host2
172.20.1.10         router2  #default router for host2
```

## How to Enable Dynamic Routing on a Single-Interface Host

Dynamic routing is the easiest way to manage routing on a host. Hosts that use dynamic routing run the routing protocols provided by the `in.routed` daemon for IPv4 or `in.ripngd` daemon for IPv6. Use the next procedure to enable IPv4 dynamic routing on a single interface host. For more information about dynamic routing, refer to "Packet Forwarding and Routing on IPv4 Networks" on page 109.

**1 On the host, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Verify whether the `/etc/defaultrouter` file exists.**

```
# cd /etc
# ls | grep defaultrouter
```

**3 If `/etc/defaultrouter` exists, delete any entry that you find there.**

An empty `/etc/defaultrouter` file forces the host to use dynamic routing.

**4 Verify whether packet forwarding and routing are enabled on the host.**

```
# routeadm
   Configuration   Current              Current
                     Option   Configuration        System State
---------------------------------------------------------------
                IPv4 routing   disabled             disabled
                IPv6 routing   disabled             disabled
              IPv4 forwarding  enabled              enabled
              IPv6 forwarding  disabled             disabled

             Routing services  "route:default ripng:default"
```

**5 If packet forwarding is enabled, turn it off**

Use either of the following commands:

- For the `routeadm` command, type the following:

  ```
  # routeadm -d ipv4-forwarding -u
  ```

- To use SMF, type the following:

```
# svcadm disable ipv4-forwarding
```

**6 Enable routing protocols on the host.**

Use either of the following commands:

- For the `routeadm` command, type the following:

```
# routeadm -e ipv4-routing -u
```

- To use SMF, type the following:

```
#svcadm enable route:default
```

Now IPv4 dynamic routing is enabled. The host's routing table is dynamically maintained by the `in.routed` daemon.

### Example 5–8 Running Dynamic Routing on a Single-Interface Host

The following example shows how to configure dynamic routing for `hosta`, a single-interface host on the network `192.168.5.0` that is shown in Figure 5–3. `hosta` currently uses Router 1 as its default router. However, `hosta` now needs to run dynamic routing.

First, you would log in to `hosta` as superuser or assume an equivalent role. Then, you would determine whether the `/etc/defaultrouter` file is present on the host:

```
# cd /etc
# ls | grep defaultrouter
defaultrouter
```

The response from `grep` indicates that a `/etc/defaultrouter` file exists for `hosta`.

```
# vi /etc/defaultrouter
192.168.5.10
```

The file has the entry `192.168.5.10`, which is the IP address for Router 1. You would delete this entry to enable static routing. Next, you would need to verify whether packet forwarding and routing are already enabled for the host.

```
# routeadm   Configuration   Current             Current
                     Option   Configuration        System State
---------------------------------------------------------------
               IPv4 routing   disabled             disabled
               IPv6 routing   disabled             disabled
            IPv4 forwarding   disabled             disabled
            IPv6 forwarding   disabled             disabled

           Routing services   "route:default ripng:default"
```

Both routing and packet forwarding are turned off for `hosta`. Turn on routing to complete the configuration of dynamic routing for `hosta`, as follows:

```
#svcadm enable route:default
```

# Monitoring and Modifying Transport Layer Services

The transport layer protocols TCP, SCTP, and UDP are part of the standard Solaris OS package. These protocols typically need no intervention to run properly. However, circumstances at your site might require you to log or modify services that run over the transport layer protocols. Then, you must modify the profiles for these services by using the Service Management Facility (SMF), which is described in Chapter 15, "Managing Services (Overview)," in *System Administration Guide: Basic Administration*.

The `inetd` daemon is responsible for starting standard Internet services when a system boots. These services include applications that use TCP, SCTP, or UDP as their transport layer protocol. You can modify existing Internet services or add new services using the SMF commands. For more information about `inetd`, refer to "`inetd` Internet Services Daemon" on page 244.

Operations that involve the transport layer protocols include:

- Logging of all incoming TCP connections
- Adding services that run over a transport layer protocol, using SCTP as an example
- Configuring the TCP wrappers facility for access control

For detailed information on the `inetd` daemon refer to the `inetd`(1M)man page.

## ▼ How to Log the IP Addresses of All Incoming TCP Connections

**1 On the local system, assume the Network Management role or become superuser.**

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**2 Set TCP tracing to enabled for all services managed by `inetd`.**

```
# inetadm -M tcp_trace=TRUE
```

## ▼ How to Add Services That Use the SCTP Protocol

The SCTP transport protocol provides services to application layer protocols in a fashion similar to TCP. However, SCTP enables communication between two systems, either or both of which can be multihomed. The SCTP connection is called an *association*. In an association, an application divides the data to be transmitted into one or more message streams, or *multi-streamed*. An SCTP connection can go to endpoints with multiple IP addresses, which is particularly important for telephony applications. The multihoming capabilities of SCTP are a security consideration if your site uses IP Filter or IPsec. Some of these considerations are described in the `sctp`(7P) man page.

By default, SCTP is included in the Solaris OS and does not require additional configuration. However, you might need to explicitly configure certain application layer services to use SCTP. Some example applications are `echo` and `discard`. The next procedure shows how to add an echo service that uses an SCTP one-to-one style socket.

---

**Note –** You can also use the following procedure to add services for the TCP and UDP transport layer protocols.

---

The following task shows how to add an SCTP `inet` service that is managed by the `inetd` daemon to the SMF repository. The task then shows how to use the Service Management Facility (SMF) commands to add the service.

- For information about SMF commands, refer to "SMF Command-Line Administrative Utilities" in *System Administration Guide: Basic Administration*.
- For syntactical information, refer to the man pages for the SMF commands, as cited in the procedure.
- For detailed information about SMF refer to the `smf`(5) man page.

**Before You Begin** Before you perform the following procedure, create a manifest file for the service. The procedure uses as an example a manifest for the `echo` service that is called `echo.sctp.xml`.

**1 Log in to the local system with a user account that has write privileges for system files.**

**2 Edit the `/etc/services` file and add a definition for the new service.**

Use the following syntax for the service definition.

*service-name* |*port/protocol* | *aliases*

**3 Add the new service.**

Go to the directory where the service manifest is stored and type the following:

```
# cd dir-name
# svccfg import service-manifest-name
```

For a complete syntax of `svccfg`, refer to the `svccfg`(1M) man page.

Suppose you want to add a new SCTP `echo` service using the manifest `echo.sctp.xml` that is currently located in the `service.dir` directory. You would type the following:

```
# cd service.dir
# svccfg import echo.sctp.xml
```

**4 Verify that the service manifest has been added:**

```
# svcs FMRI
```

For the *FMRI* argument, use the Fault Managed Resource Identifier (FMRI) of the service manifest. For example, for the SCTP `echo` service, you would use the following command:

```
# svcs svc:/network/echo:sctp_stream
```

Your output should resemble the following:

```
    STATE          STIME    FMRI
disabled       16:17:00 svc:/network/echo:sctp_stream
```

For detailed information about the `svcs` command, refer to the `svcs`(1) man page.

The output indicates that the new service manifest is currently disabled.

**5 List the properties of the service to determine if you must make modifications.**

```
# inetadm -l FMRI
```

For detailed information about the `inetadm` command, refer to the `inetadm`(1M) man page.

For example, for the SCTP `echo` service, you would type the following:

```
# inetadm -l svc:/network/echo:sctp_stream
SCOPE    NAME=VALUE
             name="echo"
             endpoint_type="stream"
             proto="sctp"
             isrpc=FALSE
             wait=FALSE
             exec="/usr/lib/inet/in.echod -s"
         .
         .
```

```
        default  tcp_trace=FALSE
          default  tcp_wrappers=FALSE
```

**6 Enable the new service:**

```
# inetadm -e FMRI
```

**7 Verify that the service is enabled:**

For example, for the new echo service, you would type the following:

```
# inetadm | grep sctp_stream
.
.
    enabled   online         svc:/network/echo:sctp_stream
```

### Example 5–9 Adding a Service That Uses the SCTP Transport Protocol

The following example shows the commands to use and the file entries required to have the echo service use the SCTP transport layer protocol.

```
$ cat /etc/services
.
.
echo            7/tcp
echo            7/udp
echo            7/sctp

# cd service.dir

    # svccfg import echo.sctp.xml

# svcs network/echo*
    STATE          STIME    FMRI
    disabled       15:46:44 svc:/network/echo:dgram
    disabled       15:46:44 svc:/network/echo:stream
    disabled       16:17:00 svc:/network/echo:sctp_stream

# inetadm -l svc:/network/echo:sctp_stream
    SCOPE    NAME=VALUE
             name="echo"
             endpoint_type="stream"
             proto="sctp"
             isrpc=FALSE
             wait=FALSE
             exec="/usr/lib/inet/in.echod -s"
             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
    default  tcp_trace=FALSE
    default  tcp_wrappers=FALSE

# inetadm -e svc:/network/echo:sctp_stream

# inetadm | grep echo
    disabled  disabled       svc:/network/echo:stream
    disabled  disabled       svc:/network/echo:dgram
    enabled   online         svc:/network/echo:sctp_stream
```

## How to Use TCP Wrappers to Control Access to TCP Services

The `tcpd` program implements *TCP wrappers*. TCP wrappers add a measure of security for service daemons such as `ftpd` by standing between the daemon and incoming service requests. TCP wrappers log successful and unsuccessful connection attempts. Additionally, TCP wrappers can provide access control, allowing or denying the connection depending on where the request originates. You can use TCP wrappers to protect daemons such as SSH, Telnet, and FTP. The `sendmail` application can also use TCP wrappers, as described in "Support for TCP Wrappers From Version 8.12 of sendmail" in *System Administration Guide: Network Services*.

**1 On the local system, assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Set TCP wrappers to enabled.**

```
# inetadm -M tcp_wrappers=TRUE
```

**3 Configure the TCP wrappers access control policy as described in the `hosts_access`(3) man page.**

This man page can be found in the `/usr/sfw/man` directory on the SFW CD-ROM, which is packaged along with the Solaris OS CD-ROM.

# Administering Interfaces in Solaris 10 3/05

This section contains the following tasks for administering physical interfaces:

- Adding physical interfaces after system installation
- Adding a virtual local area network (VLAN) to a network adapter

## What's New in This Section

This section contains information on configuring interfaces for users of the Solaris 10 3/05 OS only. If you are using an update to the Solaris 10 OS, refer to Chapter 6, "Administering Network Interfaces (Tasks)." For a complete listing of new Solaris features and a description of Solaris releases, refer to *Solaris Express Developer Edition What's New*.

## Configuring Physical Interfaces in Solaris 10 3/05

A Solaris OS-based system usually has two types of interfaces, physical and logical. *Physical interfaces* consist of a driver and a connector into which you plug network media, such as an Ethernet cable. *Logical interfaces* are logically configured onto existing physical interfaces, such as interfaces that are configured for tunnels or configured with IPv6 addresses. This section describes how to configure physical interfaces after installation. Instructions for configuring logical interfaces are included with tasks for features that require logical interfaces, for example, "How to Manually Configure IPv6 Over IPv4 Tunnels" on page 189.

Types of physical interfaces include interfaces that are built into the system and separately purchased interfaces. Each interface resides on a *network interface card* (NIC).

*Built-in NICs* are present on the system when it is purchased. An example of an interface on a built-in NIC is the *primary network interface*, such as `eri0` or `hme0`. You must configure the system's primary network interface at installation time.

NICs such as `eri` and `hme` have only one interface. However, many brands of NICs have multiple interfaces. A multiple interface NIC such as the `qfe` card has four interfaces, `qfe0` – `qfe3`. The Solaris installation program detects all interfaces present at installation and asks if you want to configure the interfaces. You can configure these interfaces at boot time or at a later date.

---

**Note –** NICs are also referred to as *network adapters*.

---

In addition to the built-in NICs, you can add separately purchased NICs to a system. You physically install a separately purchased NIC according to the manufacturer's instructions. Then, you need to configure the interfaces on the NIC so that the interfaces can be used for passing data traffic.

The following are reasons to configure additional interfaces on a system after installation:

- You want to upgrade the system to become a multihomed host. For more information about multihomed hosts, refer to "Configuring Multihomed Hosts" on page 124.
- You want to change the host to a router. For instructions on configuring routers, refer to "Configuring an IPv4 Router" on page 115.
- You want to add an interface to an IPMP group. For information about interfaces in an IPMP group, refer to "IPMP Interface Configurations" on page 724.

## ▼ How to Add a Physical Interface After Installation in Solaris 10 3/05 ONLY

**Before You Begin** Determine the IPv4 addresses that you want to use for the additional interfaces.

The physical interface to be configured must be present on the system. For information on installing separately purchased NIC hardware, refer to the manufacturers instructions that accompany the NIC.

The next procedure assumes that you have performed a reconfiguration boot after physically installing a new interface.

---

**Note –** The next procedure contains applies to users of the Solaris 10 3/05 OS only. If you are using an update to the Solaris 10 OS, refer to "How to Configure a Physical Interface After System Installation" on page 150.

---

**1 On the system with the interfaces to be configured, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Configure and plumb each interface.**

```
# ifconfig interface plumb up
```

For example, for `qfe0` you would type:

```
# ifconfig qfe0 plumb up
```

**Note –** Interfaces that are explicitly configured with the `ifconfig` command do not persist across a reboot.

**3 Assign an IPv4 address and netmask to the interface.**

```
# ifconfig interface IPv4-address netmask+netmask
```

For example, for `qfe0` you would type:

```
# ifconfig qfe0 10.0.0.32 netmask + 255.255.255.0
```

**4 Verify that the newly configured interfaces are plumbed and configured, or "`UP`."**

```
# ifconfig -a
```

Check the status line for each interface that is displayed. Ensure that the output contains an `UP` flag on the status line, for example:

```
qfe0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
```

**5 (Optional) To make the interface configuration persist across reboots, perform the following steps:**

**a. Create an `/etc/hostname.`*interface* file for each interface to be configured.**

For example, to add a `qfe0` interface, you would create the following file:

```
# vi /etc/hostname.qfe0
```

**b. Edit the `/etc/hostname.`*interface* file.**

At a minimum, add the IPv4 address of the interface to the file. You can also add a netmask and other configuration information to the file.

**Note –** To add an IPv6 address to an interface, refer to "Modifying an IPv6 Interface Configuration for Hosts and Servers" on page 181

**c. Add entries for the new interfaces into the `/etc/inet/hosts` file.**

**d. Perform a reconfiguration boot.**

```
# reboot -- -r
```

**e. Verify that the interface you created in the `/etc/hostname.`*interface* file has been configured.**

```
# ifconfig -a
```

**Example 5–10** Configuring an Interface After System Installation

The following example shows how to add two interfaces, `qfe0` and `qfe1`. These interfaces are attached to the same network as the primary network interface, `hme0`. Note that this interface configuration exists until you reboot the system. For an example that shows how to make interface configurations persist across reboots, see Example 6–2. However, the `dladm` command that is used in that example is only available starting with the Solaris 10 1/06 OS.

```
# ifconfig qfe0 plumb up
# ifconfig qfe1 plumb up
# ifconfig qfe0 10.0.0.32 netmask 255.0.0.0
# ifconfig qfe1 10.0.0.33 netmask 255.0.0.0

ifconfig -a
# ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
hme0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.0.0.14 netmask ff000000 broadcast 10.255.255.255
        ether 8:0:20:c1:8b:c3
qfe0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 10.0.0.32 netmask ff000000 broadcast 10.255.255.255
        ether 8:0:20:c8:f4:1d
qfe1: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 10.0.0.33 netmask ff000000 broadcast 10.255.255.255
        ether 8:0:20:c8:f4:1e
```

**See Also**

- To configure an IPv6 address onto an interface, refer to "How to Enable an IPv6 Interface for the Current Session" on page 172.
- To set up failover detection and failback for interfaces using Network Multipathing (IPMP), refer to Chapter 31, "Administering IPMP (Tasks)."

## ▼ How to Remove a Physical Interface in Solaris 10 3/05 ONLY

**Note –** The next procedure contains applies to users of the Solaris 10 3/05 OS only. If you are using an update to the Solaris 10 OS, refer to "How to Remove a Physical Interface" on page 153.

**1 On the system with the interface to be removed, assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Remove the physical interface.**

Use the following form of the `ifconfig` command:

```
# ifconfig interface unplumb down
```

For example, you would remove the interface `eri1` as follows:

```
# ifconfig eri1 unplumb down
```

# Configuring VLANs in Solaris 10 3/05 ONLY

**Note –** This section contains information on configuring VLANs for users of the Solaris 10 3/05 OS only. If you are using an update to the Solaris 10 OS, refer to "Administering Virtual Local Area Networks" on page 156.

Virtual local area networks (VLANs) are commonly used to split up groups of network users into manageable broadcast domains, to create logical segmentation of work groups, and to enforce security policies among each logical segment. With multiple VLANs on an adapter, a server with a single adapter can have a logical presence on multiple IP subnets. By default, 512 VLANs can be defined for each VLAN-aware adapter on your server.

If your network does not require multiple VLANs, you can use the default configuration, in which case no further configuration is necessary.

For an overview of VLANs, refer to "Overview of VLAN Topology" on page 156.

VLANs can be created according to various criteria, but each VLAN must be assigned a VLAN tag or VLAN ID (VID). The VID is a 12-bit identifier between 1 and 4094 that identifies a unique VLAN. For each network interface (for example, `ce0`, `ce1`, `ce2`, and so on) 512 possible VLANs can be created. Because IP subnets are commonly used, use IP subnets when setting up a VLAN network interface. This means that each VID assigned to a VLAN interface of a physical network interface belongs to different subnets.

Tagging an Ethernet frame requires the addition of a tag header to the frame. The header is inserted immediately following the destination MAC address and the source MAC address. The tag header consists of two bytes of the Ethernet Tag Protocol Identifier (TPID, 0x8100) and two bytes of Tag Control Information (TCI). The following figure shows the Ethernet Tag Header format.

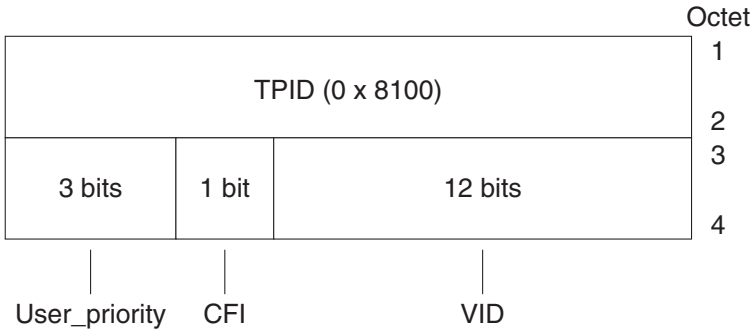



**FIGURE 5–4** Ethernet Tag Header Format

## How To Configure Static VLANs in Solaris 10 3/05 ONLY

**Note –** This procedure contains information on configuring VLANs for users of the Solaris 10 3/05 OS only. If you are using an update to the Solaris 10 OS, refer to "How to Configure a VLAN" on page 160

**1 Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Determine the type of interfaces in use on your system.**

The network adapter on your system might not be referred to by the letters `ce`, which is required for a VLAN.

```
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4>
mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4>
mtu 1500 index 2
        inet 129.156.200.77 netmask ffffff00 broadcast
129.156.200.255
```

**3 Create one `hostname.ce`*num* file (`hostname6.ce`*num* file for IPv6) for each VLAN that will be configured for each adapter on the server.**

Use the following naming format that includes both the VID and the physical point of attachment (PPA):

VLAN logical PPA = 1000 * *VID* + *Device PPA* `ce123000 = 1000*123 + 0`

For example: `hostname.ce123000`

VLAN logical PPA = 1000 * *VID* + *Device PPA* `ce11000 = 1000*11 + 0`

For example: `hostname.ce11000`

This format limits the maximum number of PPAs (instances) you can configure to 1000 in the `/etc/path_to_inst` file.

For example, on a server with the Sun Gigabit Ethernet/P 3.0 adapter having an instance of 0, that belongs to two VLANs with VIDs 123 and 224, you would use `ce123000` and `ce224000`, respectively, as the two VLAN PPAs.

**4 Configure a VLAN virtual device:**

For example, you could use the following examples of `ifconfig`:

```
# ifconfig ce123000 plumb up
# ifconfig ce224000 plumb up
```

The output of `ifconfig -a` on a system with VLAN devices `ce123000` and `ce224000` should resemble the following:

```
# ifconfig -a
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
hme0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 129.144.131.91 netmask ffffff00 broadcast 129.144.131.255
        ether 8:0:20:a4:4f:b8
ce123000: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 199.199.123.3 netmask ffffff00 broadcast 199.199.123.255
        ether 8:0:20:a4:4f:b8
ce224000: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 199.199.224.3 netmask ffffff00 broadcast 199.199.224.255
        ether 8:0:20:a4:4f:b8
```

**5 On the switch, set VLAN tagging and VLAN ports to coincide with the VLANs you have set up on the server.**

Using the examples in Step 4, you would set up VLAN ports 123 and 224 on the switch or VLAN ports 10 and 11.

Refer to the documentation that came with your switch for specific instructions for setting VLAN tagging and ports.

◆ ◆ ◆ CHAPTER 6

# Administering Network Interfaces (Tasks)

This chapter contains tasks and information about network interfaces:


- "Interface Administration (Task Map)" on page 146
- "Basics for Administering Physical Interfaces" on page 146
- "Administering Individual Network Interfaces" on page 148
- "Administering Virtual Local Area Networks" on page 156
- "Overview of Link Aggregations" on page 161


## What's New in Administering Network Interfaces

The information in this chapter describes interface configuration starting with the Solaris 10 1/06 release. If you are using the original release of Solaris 10, 3/05, refer to "Administering Interfaces in Solaris 10 3/05" on page 137. For a complete listing of new Solaris features and a description of Solaris releases, refer to *Solaris Express Developer Edition What's New*.

In Solaris 10 1/06, the following new features were introduced:

- The new `dladm` command for viewing interface status is introduced in "How to Configure a Physical Interface After System Installation" on page 150.
- VLAN support has extended to GLDv3 interfaces, as explained in "Administering Virtual Local Area Networks" on page 156.
- Link aggregation support is introduced in "Overview of Link Aggregations" on page 161.

In Solaris 10 8/07, the `/etc/inet/ipnodes` becomes obsolete. Use `/etc/inet/ipnodes` only for earlier Solaris 10 releases, as explained in the individual procedures.

# Interface Administration (Task Map)

| Task | Description | For Instructions |
|---|---|---|
| Check the status of interfaces on a system. | List all interfaces on the system and check which interfaces are already plumbed. | "How to Obtain Interface Status" on page 149 |
| Add a single interface after system installation. | Change a system to a multihomed host or router by configuring another interface. | "How to Configure a Physical Interface After System Installation" on page 150 |
| SPARC: Check that the MAC address of an interface is unique. | Ensure that the interface is configured with its factory-installed MAC address, rather than the system MAC address (SPARC only). | "SPARC: How to Ensure That the MAC Address of an Interface Is Unique" on page 154 |
| Plan for a virtual local area network (VLAN). | Perform required planning tasks prior to creating a VLAN. | "How to Plan a VLAN Configuration" on page 159 |
| Configure a VLAN. | Create and modify VLANs on your network. | "How to Configure a VLAN" on page 160 |
| Plan for aggregations. | Design your aggregation and perform required planning tasks prior to configuring aggregations. | "Overview of Link Aggregations" on page 161 |
| Configure an aggregation. | Perform various tasks related to link aggregations. | "How to Create a Link Aggregation" on page 165 |
| Plan for and configure an IPMP group. | Configure failover and failback for interfaces that are members of an IPMP group. | "How to Plan for an IPMP Group" on page 735<br>"How to Configure an IPMP Group With Multiple Interfaces" on page 737 |

# Basics for Administering Physical Interfaces

*Network interfaces* provide the connection between a system and a network. A Solaris OS-based system can have two types of interfaces, physical and logical. *Physical interfaces* consist of a software driver and a connector into which you connect network media, such as an Ethernet cable. Physical interfaces can be grouped for administrative or availability purposes. *Logical interfaces* are configured onto existing physical interfaces, usually for adding addresses and creating tunnel endpoints on the physical interfaces.

**Note –** Logical network interfaces are described in the tasks where they are used: IPv6 tasks, IPMP tasks, DHCP tasks, and others.

Most computer systems have at least one physical interface that is *built-in* by the manufacturer on the main system board. Some systems can also have more than one built-in interface.

In addition to built-in interfaces, you can add separately purchased interfaces to a system. A separately purchased interface is known as a *network interface card* (NIC). You physically install a NIC according to the manufacturer's instructions.

**Note –** NICs are also referred to as *network adapters*.

During system installation, the Solaris installation program detects any interfaces that are physically installed and displays each interface's name. You must configure at least one interface from the list of interfaces. The first interface to be configured during installation becomes the *primary network interface*. The IP address of the primary network interface is associated with the configured host name of the system, which is stored in the `/etc/nodename` file. However, you can configure any additional interfaces during installation or later.

## Network Interface Names

Each physical interface is identified by a unique device name. Device names have the following syntax:

```
<driver-name><instance-number>
```

Driver names on Solaris systems could include `ce`, `hme`, `bge`, `e1000g` and many other driver names. The variable *instance-number* can have a value from zero to *n*, depending on how many interfaces of that driver type are installed on the system.

For example, consider a 100BASE-TX Fast Ethernet interface, which is often used as the primary network interface on both host systems and server systems. Some typical driver names for this interface are `eri`, `qfe`, and `hme`. When used as the primary network interface, the Fast Ethernet interface has a device name such as `eri0` or `qfe0`.

NICs such as `eri` and `hme` have only one interface. However, many brands of NICs have multiple interfaces. For example, the Quad Fast Ethernet (`qfe`) card has four interfaces, `qfe0` through `qfe3`.

## Plumbing an Interface

An interface must be *plumbed* before it can pass traffic between the system and the network. The plumbing process involves associating an interface with a device name. Then, streams are set up so that the interface can be used by the IP protocol. Both physical interfaces and logical interfaces must be plumbed. Interfaces are plumbed either as part of the boot sequence or explicitly, with the appropriate syntax of the `ifconfig` command.

When you configure an interface during installation, the interface is automatically plumbed. If you decide during installation not to configure the additional interfaces on the system, those interfaces are not plumbed.

## Solaris OS Interface Types

Starting with the Solaris 10 1/06 release, the Solaris OS supports the following two types of interfaces:

- **Legacy interfaces** – These interfaces are DLPI interfaces and GLDv2 interfaces. Some legacy interface types are `eri`, `qfe`, and `ce`. When you check interface status with the `dladm show-link` command, these interfaces are reported as "legacy."
- **Non-VLAN interfaces** – These interfaces are GLDv3 interfaces.

---

**Note –** Currently GLDv3 is supported on the following interface types: `bge`, `xge`, and `e1000g`.

---

# Administering Individual Network Interfaces

After Solaris installation, you might configure or administer interfaces on a system for the following purposes:

- To upgrade the system to become a multihomed host. For more information, refer to "Configuring Multihomed Hosts" on page 124.
- To change a host to a router. For instructions on configuring routers, refer to "Configuring an IPv4 Router" on page 115.
- To configure interfaces as part of a VLAN. For more information, refer to "Administering Virtual Local Area Networks" on page 156.
- To configure interfaces as members of an aggregation. For more information, refer to "Overview of Link Aggregations" on page 161.
- To add an interface to an IPMP group. For instructions on configuring an IPMP group, refer to "Configuring IPMP Groups" on page 735

This section contains information about configuring individual network interfaces, starting with the Solaris 10 1/06 release. Refer to the following sections for information about configuring interfaces into one of the following groupings:

- For configuring interfaces into a VLAN, refer to "Administering Virtual Local Area Networks" on page 156.
- For configuring interfaces into an aggregation, refer to "Overview of Link Aggregations" on page 161.
- For configuring interfaces as members of IPMP groups, refer to "Configuring IPMP Groups" on page 735.

## ▼ How to Obtain Interface Status

Starting with Solaris 10 1/06, this procedure explains how to determine which interfaces are currently available on a system and their status. This procedure also shows which interfaces are currently plumbed. If you are using the earlier Solaris 10 3/05, refer to "How to Get Information About a Specific Interface" on page 205.

**1 On the system with the interfaces to be configured, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Determine which interfaces are currently installed on your system.**

```
# dladm show-link
```

This step uses the `dladm` command, which is explained in detail in the `dladm`(1M) man page. This command reports on all the interface drivers that it finds, regardless of whether the interfaces are currently configured.

**3 Determine which interfaces on the system are currently plumbed.**

```
# ifconfig -a
```

The `ifconfig` command has many additional functions, including plumbing an interface. For more information, refer to the `ifconfig`(1M) man page.

### Example 6–1 Obtaining the Status of an Interface with the `dladm` command

The next example shows the status display of the `dladm` command.

```
# dladm show-link

ce0             type: legacy    mtu: 1500       device: ce0
```

```
ce1             type: legacy    mtu: 1500       device: ce1
bge0            type: non-vlan  mtu: 1500       device: bge0
bge1            type: non-vlan  mtu: 1500       device: bge1
bge2            type: non-vlan  mtu: 1500       device: bge2
```

The output of `dladm show-link` indicates that four interface drivers are available for the local host. Both the `ce` and the `bge` interfaces can be configured for VLANs. However, only the GLDV3 interfaces with a type of `non-VLAN` can be used for link aggregations.

The next example shows the status display of the `ifconfig -a` command.

```
# ifconfig -a

lo0: flags=2001000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
ce0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.84.253 netmask ffffff00 broadcast 192.168.84.255
        ether 0:3:ba:7:84:5e
bge0: flags=1004843 <UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500 index 2
        inet 10.8.57.39 netmask ffffff00 broadcast 10.8.57.255
        ether 0:3:ba:29:fc:cc
```

The output of the `ifconfig -a` command displays statistics for only two interfaces, `ce0` and `bge0`. This output shows that only `ce0` and `bge0` have been plumbed and are ready for use by network traffic. These interfaces can be used in a VLAN. Because `bge0` has been plumbed, you can no longer use this interface in an aggregation.

## ▼ How to Configure a Physical Interface After System Installation

Use the next procedure for configuring interfaces. If you are using the Solaris 10 3/05 release, use the procedure "How to Add a Physical Interface After Installation in Solaris 10 3/05 ONLY" on page 138.

**Before You Begin**

- Determine the IPv4 addresses that you want to use for the additional interfaces.
- Ensure that the physical interface to be configured has been physically installed onto the system. For information about installing separately purchased NIC hardware, refer to the manufacturer's instructions that accompany the NIC.
- If you have just installed the interface, perform a reconfiguration boot before proceeding with the next task.

1. **On the system with the interfaces to be configured, assume the Primary Administrator role or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Determine which interfaces are currently installed on the system.**

   ```
   # dladm show-link
   ```

3. **Configure and plumb each interface.**

   ```
   # ifconfig interface plumb up
   ```

   For example, for `qfe0` you would type:

   ```
   # ifconfig qfe0 plumb up
   ```

   **Note –** Interfaces that are explicitly configured with the `ifconfig` command do not persist across a reboot.

4. **Assign an IPv4 address and netmask to the interface.**

   ```
   # ifconfig interface IPv4-address netmask+netmask
   ```

   For example, for `qfe0` you would type:

   ```
   # ifconfig qfe0 192.168.84.3 netmask + 255.255.255.0
   ```

   **Note –** You can specify an IPv4 address in either traditional IPv4 notation or CIDR notation.

5. **Verify that the newly configured interfaces are plumbed and configured, or "`UP`."**

   ```
   # ifconfig -a
   ```

   Check the status line for each interface that is displayed. Ensure that the output contains an `UP` flag on the status line, for example:

   ```
   qfe0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
   ```

6. **(Optional) To make the interface configuration persist across reboots, perform the following steps:**

   a. **Create an `/etc/hostname.`*interface* file for each interface to be configured.**

      For example, to add a `qfe0` interface, you would create the following file:

      ```
      # vi /etc/hostname.qfe0
      ```

**b. Edit the `/etc/hostname.`*interface* file.**

At a minimum, add the IPv4 address of the interface to the file. You can use traditional IPv4 notation or CIDR notation to specify the IP address of the interface. You can also add a netmask and other configuration information to the file.

---

**Note –** To add an IPv6 address to an interface, refer to "Modifying an IPv6 Interface Configuration for Hosts and Servers" on page 181

---

**c. For Solaris 10 11/06 and earlier releases of Solaris 10, add entries for the new interfaces into the `/etc/inet/ipnodes` file.**

**d. Add entries for the new interfaces into the `/etc/inet/hosts` file.**

**e. Perform a reconfiguration boot.**

```
# reboot -- -r
```

**f. Verify that the interface you created in the `/etc/hostname.`*interface* file has been configured.**

```
# ifconfig -a
```

For examples, refer to Example 6–2.

**Example 6–2** Adding Persistent Interface Configurations

The example shows how to configure the interfaces `qfe0` and `qfe1` to a host. These interfaces remain persistent across reboots.

```
# dladm show-link
eri0              type: legacy    mtu: 1500       device: eri0
qfe0              type: legacy    mtu: 1500       device: qfe0
qfe1              type: legacy    mtu: 1500       device: qfe1
qfe2              type: legacy    mtu: 1500       device: qfe2
qfe3              type: legacy    mtu: 1500       device: qfe3
bge0              type: non-vlan  mtu: 1500       device: bge0

# vi /etc/hostname.qfe0
192.168.84.3
netmask + 255.255.255.0
# vi /etc/hostname.qfe1
192.168.84.72
netmask + 255.255.255.0
# vi /etc/inet/hosts
# Internet host table
#
127.0.0.1       localhost
```

```
10.0.0.14       myhost
192.168.84.3       interface-2
192.168.84.72       interface-3
For Solaris 10 11/06 and earlier releases:# vi /etc/inet/ipnodes
10.0.0.14 myhost
192.168.84.3       interface-2
192.168.84.72       interface-3
```

At this point, you would reboot the system.

```
# reboot -- -r
```

After the system boots, you would then verify the interface configuration.

```
ifconfig -a
# ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
eri0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.0.0.14 netmask ff000000 broadcast 10.255.255.255
        ether 8:0:20:c1:8b:c3
qfe0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.84.3 netmask ffffff00 broadcast 192.255.255.255
        ether 8:0:20:c8:f4:1d
qfe1: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 192.168.84.72 netmask ffffff00 broadcast 10.255.255.255
        ether 8:0:20:c8:f4:1e
```

**See Also**

- To configure an IPv6 address onto an interface, refer to "How to Enable an IPv6 Interface for the Current Session" on page 172.
- To set up failover detection and failback for interfaces by using IP Network Multipathing (IPMP), refer to Chapter 31, "Administering IPMP (Tasks)."

## ▼ How to Remove a Physical Interface

Use this procedure for removing a physical interface. If you are using the earlier Solaris 10 3/05, refer to "How to Remove a Physical Interface in Solaris 10 3/05 ONLY" on page 140.

**1 On the system with the interface to be removed, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Remove the physical interface.**

```
# ifconfig interface unplumb down
```

For example, to remove the interface `qfe1`, you would type:

```
# ifconfig qfe1 unplumb down
```

## ▼ SPARC: How to Ensure That the MAC Address of an Interface Is Unique

Use this procedure for configuring MAC addresses.

Some applications require every interface on a host to have a unique MAC addresses. However, every SPARC based system has a system-wide MAC address, which by default is used by all interfaces. Here are two situations where you might want to configure the factory-installed MAC addresses for the interfaces on a SPARC system.

- For link aggregations, you should use the factory-set MAC addresses of the interfaces in the aggregation configuration.
- For IPMP groups, each interface in the group must have a unique MAC address. These interfaces must use their factory-installed MAC addresses.

The EEPROM parameter `local-mac-address?` determines whether all interfaces on a SPARC system use the system-wide MAC address or their unique MAC address. The next procedure shows how to use the `eeprom` command to check the current value of `local-mac-address?` and change it, if necessary.

1 **On the system with the interfaces to be configured, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Determine whether all interfaces on the system currently use the system-wide MAC address.**

```
# eeprom local-mac-address?
local-mac-address?=false
```

In the example, the response to the `eeprom` command, `local-mac-address?=false`, indicates that all interfaces do use the system-wide MAC address. The value of `local-mac-address?=false` must be changed to `local-mac-address?=true` before the interfaces can become members of an IPMP group. You should also change `local-mac-address?=false` to `local-mac-address?=true` for aggregations.

3 **If necessary, change the value of `local-mac-address?` as follows:**

```
# eeprom local-mac-address?=true
```

When you reboot the system, the interfaces with factory-installed MAC addresses now use these factory settings, rather than the system-wide MAC address. Interfaces without factory-set MAC addresses continue to use the system-wide MAC address.

4 **Check the MAC addresses of all the interfaces on the system.**

Look for cases where multiple interfaces have the same MAC address. In this example, all interfaces use the system-wide MAC address `8:0:20:0:0:1`.

```
ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
     inet 127.0.0.1 netmask ff000000
hme0: flags=1004843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
     inet 10.0.0.112 netmask ffffff80 broadcast 10.0.0.127
     ether 8:0:20:0:0:1
ce0: flags=1004843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
     inet 10.0.0.114 netmask ffffff80 broadcast 10.0.0.127
     ether 8:0:20:0:0:1
ce1: flags=1004843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
     inet 10.0.0.118 netmask ffffff80 broadcast 10.0.0.127
     ether 8:0:20:0:0:1
```

---

**Note –** Continue to the next step only if more than one network interface still has the same MAC address. Otherwise, go on to the final step.

---

5 **If necessary, manually configure the remaining interfaces so that all interfaces have unique MAC address.**

Specify a unique MAC address in the `/etc/hostname.`*interface* file for the particular interface.

In the example in Step 4, you would need to configure `ce0` and `ce1` with locally administered MAC addresses. For example, to reconfigure `ce1` with the locally administered MAC address `06:05:04:03:02`, you would add the following line to `/etc/hostname.ce1`:

```
ether 06:05:04:03:02
```

---

**Note –** To prevent any risk of manually configured MAC addresses conflicting with other MAC addresses on your network, you must always configure *locally administered* MAC addresses, as defined by the IEEE 802.3 standard.

---

You also can use the `ifconfig ether` command to configure an interface's MAC address for the current session. However, any changes made directly with `ifconfig` are not preserved across reboots. Refer to the `ifconfig`(1M) man page for details.

6 **Reboot the system.**

# Administering Virtual Local Area Networks

**Note –** If you are using the earlier Solaris 3/05, refer to "Configuring VLANs in Solaris 10 3/05 ONLY" on page 141.

A *virtual local area network (VLAN)* is a subdivision of a local area network at the data link layer of the TCP/IP protocol stack. You can create VLANs for local area networks that use switch technology. By assigning groups of users to VLANs, you can improve network administration and security for the entire local network. You can also assign interfaces on the same system to different VLANs.

Consider dividing your local network into VLANs if you need to do the following:

- Create a logical division of workgroups.

  For example, suppose all hosts on a floor of a building are connected on one switched-based local network. You could create a separate VLAN for each workgroup on the floor.
- Enforce differing security policies for the workgroups.

  For example, the security needs of a Finance department and an Information Technologies department are quite different. If systems for both departments share the same local network, you could create a separate VLAN for each department. Then, you could enforce the appropriate security policy on a per-VLAN basis.
- Split workgroups into manageable broadcast domains.

  The use of VLANs reduces the size of broadcast domains and improves network efficiency.

## Overview of VLAN Topology

Switched LAN technology enables you to organize the systems on a local network into VLANs. Before you can divide a local network into VLANs, you must obtain switches that support VLAN technology. You can configure all ports on a switch to serve a single VLAN or multiple VLANs, depending on the VLAN topology design. Each switch manufacturer has different procedures for configuring the ports of a switch.

The following figure shows a local area network that has the subnet address `192.168.84.0`. This LAN is subdivided into three VLANs, Red, Yellow, and Blue.

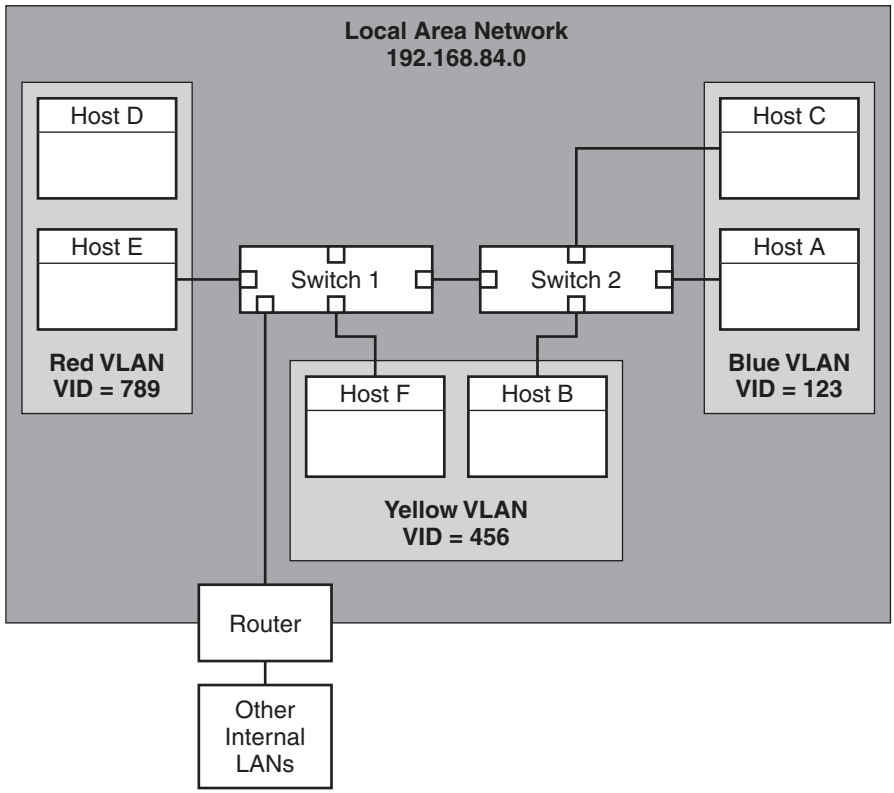



**FIGURE 6–1** Local Area Network With Three VLANs

Connectivity on LAN `192.168.84.0` is handled by Switches 1 and 2. The Red VLAN contains systems in the Accounting workgroup. The Human Resources workgroup's systems are on the Yellow VLAN. Systems of the Information Technologies workgroup are assigned to the Blue VLAN.

## VLAN Tags and Physical Points of Attachment

Each VLAN in a local area network is identified by a VLAN tag, or *VLAN ID (VID)*. The VID is assigned during VLAN configuration. The VID is a 12-bit identifier between 1 and 4094 that provides a unique identity for each VLAN. In Figure 6–1, the Red VLAN has the VID 789, the Yellow VLAN has the VID 456, and the Blue VLAN has the VID 123.

When you configure switches to support VLANs, you need to assign a VID to each port. The VID on the port must be the same as the VID assigned to the interface that connects to the port, as shown in the following figure.

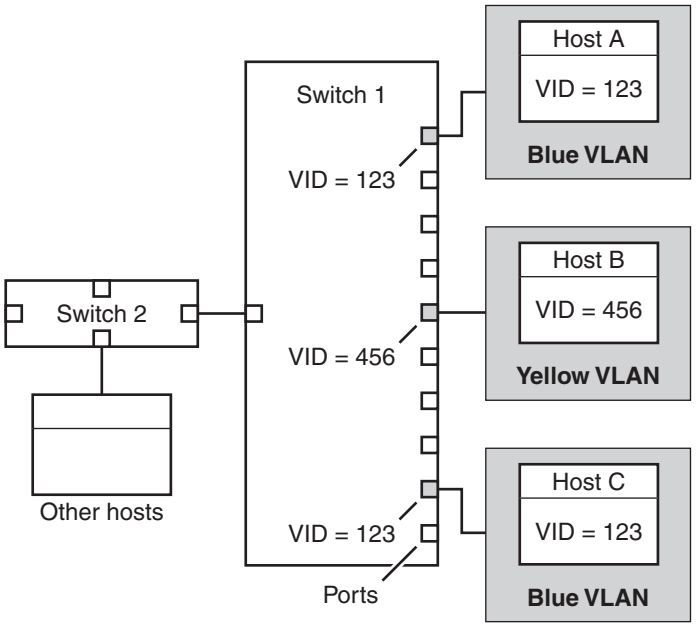



**FIGURE 6–2** Switch Configuration for a Network with VLANs

In this figure, the primary network interfaces of the three hosts connect to Switch 1. Host A is a member of the Blue VLAN. Therefore, Host A's interface is configured with the VID 123. This interface connects to Port 1 on Switch 1, which is then configured with the VID 123. Host B is a member of the Yellow VLAN with the VID 456. Host B's interface connects to Port 5 on Switch 1, which is configured with the VID 456. Finally, Host C's interface connects to Port 9 on Switch 1. The Blue VLAN is configured with the VID 123.

During VLAN configuration, you have to specify the *physical point of attachment*, or *PPA*, of the VLAN. You obtain the PPA value by using this formula:

*driver-name* `+ VID * 1000 +` *device-instance*

Note that the *device-instance* number must be less than 1000.

For example, you would create the following PPA for a `ce1` interface to be configured as part of VLAN 456:

```
ce + 456 * 1000 + 1= ce456001
```

# Planning for VLANs on a Network

Use the following procedure to plan for VLANs on your network.

### ▼ How to Plan a VLAN Configuration

1 **Examine the local network topology and determine where subdivision into VLANs is appropriate.**

For a basic example of such a topology, refer to Figure 6–1.

2 **Create a numbering scheme for the VIDs, and assign a VID to each VLAN.**

---

**Note –** A VLAN numbering scheme might already exist on the network. If so, you must create VIDs within the existing VLAN numbering scheme.

---

3 **On each system, determine which interfaces will be members of a particular VLAN.**

a. **Determine which interfaces are configured on a system.**

```
# dladm show-link
```

b. **Identify which VID will be associated with each data link on the system.**

c. **Create PPAs for each interface to be configured with a VLAN.**

All interfaces on a system do not necessarily have to be configured on the same VLAN.

4 **Check the connections of the interfaces to the network's switches.**

Note the VID of each interface and the switch port where each interface is connected.

5 **Configure each port of the switch with the same VID as the interface to which it is connected.**

Refer to the switch manufacturer's documentation for configuration instructions.

## Configuring VLANs

---

**Note –** If you are using the earlier Solaris 10 3/05, refer to "Configuring VLANs in Solaris 10 3/05 ONLY" on page 141.

---

The Solaris OS now supports VLANs on the following interface types:

- `ce`
- `bge`
- `xge`
- `e1000g`

Of the legacy interface types, only the `ce` interface can become a member of a VLAN. You can configure interfaces of different types in the same VLAN.

## How to Configure a VLAN

If you are using Solaris 10 3/05, use the procedure "How To Configure Static VLANs in Solaris 10 3/05 ONLY" on page 142.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Determine the types of interfaces in use on your system.**

   ```
   # dladm show-link
   ```

   The output shows the available interface types:

   ```
   ce0             type: legacy    mtu: 1500       device: ce0
   ce1             type: legacy    mtu: 1500       device: ce1
   bge0            type: non-vlan  mtu: 1500       device: bge0
   bge1            type: non-vlan  mtu: 1500       device: bge1
   bge2            type: non-vlan  mtu: 1500       device: bge2
   ```

3. **Configure an interface as part of a VLAN.**

   **# ifconfig** *interface-PPA* **plumb** *IP-address* **up**

   For example, you would use the following command to configure the interface `ce1` with a new IP address `10.0.0.2` into a VLAN with the VID 123:

   ```
   # ifconfig ce123001 plumb 10.0.0.2 up
   ```

4. **(Optional) To make the VLAN settings persist across reboots, create a `hostname.`*interface-PPA* file for each interface that is configured as part of a VLAN.**

   **# cat hostname.***interface-PPA*
   *IPv4-address*

5. **On the switch, set VLAN tagging and VLAN ports to correspond with the VLANs that you have set up on the system.**

**Example 6–3** Configuring a VLAN

This example shows how to configure devices `bge1` and `bge2` into a VLAN with the VID 123.

```
# dladm show-link
ce0             type: legacy    mtu: 1500       device: ce0
ce1             type: legacy    mtu: 1500       device: ce1
bge0            type: non-vlan  mtu: 1500       device: bge0
bge1            type: non-vlan  mtu: 1500       device: bge1
```

```
bge2            type: non-vlan  mtu: 1500       device: bge2
# ifconfig bge123001 plumb 10.0.0.1 up
# ifconfig bge123002 plumb 10.0.0.2 up
# cat hostname.bge123001
  10.0.0.1
# cat hostname.bge123002
  10.0.0.2
# ifconfig -a

lo0: flags=2001000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL>
     mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
bge123001: flags=201000803 <UP,BROADCAST,MULTICAST,IPv4,CoS> mtu 1500 index 2
        inet 10.0.0.1 netmask ff000000 broadcast 10.255.255.255
        ether 0:3:ba:7:84:5e
bge123002: flags=201000803 <UP,BROADCAST,MULTICAST,IPv4,CoS> mtu 1500 index 3
        inet 10.0.0.2 netmask ff000000 broadcast 10.255.255.255
        ether 0:3:ba:7:84:5e
ce0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 4
        inet 192.168.84.253 netmask ffffff00 broadcast 192.168.84.255
        ether 0:3:ba:7:84:5e
# dladm show-link
ce0             type: legacy    mtu: 1500       device: ce0
ce1             type: legacy    mtu: 1500       device: ce1
bge0            type: non-vlan  mtu: 1500       device: bge0
bge1            type: non-vlan  mtu: 1500       device: bge1
bge2            type: non-vlan  mtu: 1500       device: bge2
bge123001       type: vlan 123  mtu: 1500       device: bge1
bge123002       type: vlan 123  mtu: 1500       device: bge2
```

# Overview of Link Aggregations

**Note –** The original Solaris 10 release and earlier versions of Solaris do not support Link Aggregations. To create link aggregations for these earlier Solaris releases, use Sun Trunking, as described in the *Sun Trunking 1.3 Installation and Users Guide.*

The Solaris OS supports the organization of network interfaces into link aggregations. A *link aggregation* consists of several interfaces on a system that are configured together as a single, logical unit. Link aggregation, also referred to as *trunking*, is defined in the IEEE 802.3ad Link Aggregation Standard (`http://www.ieee802.org/3/index.html`).

The IEEE 802.3ad Link Aggregation Standard provides a method to combine the capacity of multiple full-duplex Ethernet links into a single logical link. This link aggregation group is then treated as though it were, in fact, a single link.

The following are features of link aggregations:

- **Increased bandwidth** – The capacity of multiple links is combined into one logical link.
- **Automatic failover/failback** – Traffic from a failed link is failed over to working links in the aggregation.
- **Load balancing** – Both inbound and outbound traffic is distributed according to user selected load-balancing policies, such as source and destination MAC or IP addresses.
- **Support for redundancy** – Two systems can be configured with parallel aggregations.
- **Improved administration** – All interfaces are administered as a single unit.
- **Less drain on the network address pool** – The entire aggregation can be assigned one IP address.

## Link Aggregation Basics

The basic link aggregation topology involves a single aggregation that contains a set of physical interfaces. You might use the basic link aggregation in the following situations:

- For systems that run an application with distributed heavy traffic, you can dedicate an aggregation to that application's traffic.
- For sites with limited IP address space that nevertheless require large amounts of bandwidth, you need only one IP address for a large aggregation of interfaces.
- For sites that need to hide the existence of internal interfaces, the IP address of the aggregation hides its interfaces from external applications.

Figure 6–3 shows an aggregation for a server that hosts a popular web site. The site requires increased bandwidth for query traffic between Internet customers and the site's database server. For security purposes, the existence of the individual interfaces on the server must be hidden from external applications. The solution is the aggregation `aggr1` with the IP address `192.168.50.32`. This aggregation consists of three interfaces,`bge0` through `bge2`. These interfaces are dedicated to sending out traffic in response to customer queries. The outgoing address on packet traffic from all the interfaces is the IP address of `aggr1`, `192.168.50.32`.

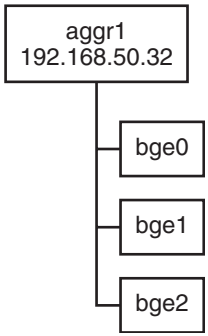



**FIGURE 6–3** Basic Link Aggregation Topology

Figure 6–4 depicts a local network with two systems, and each system has an aggregation configured. The two systems are connected by a switch. If you need to run an aggregation through a switch, that switch must support aggregation technology. This type of configuration is particularly useful for high availability and redundant systems.

In the figure, System A has an aggregation that consists of two interfaces, `bge0` and `bge1`. These interfaces are connected to the switch through aggregated ports. System B has an aggregation of four interfaces, `e1000g0` through `e1000g3`. These interfaces are also connected to aggregated ports on the switch.

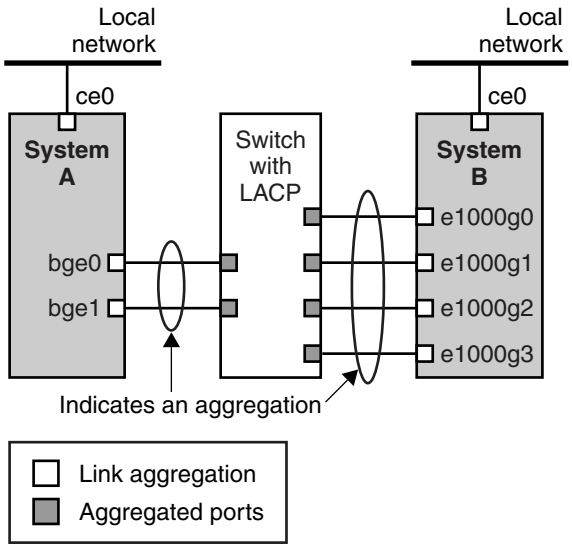



**FIGURE 6–4** Link Aggregation Topology With a Switch

## Back-to-Back Link Aggregations

The back-to-back link aggregation topology involves two separate systems that are cabled directly to each other, as shown in the following figure. The systems run parallel aggregations.

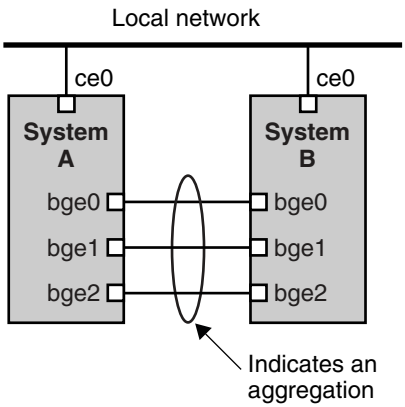



**FIGURE 6–5** Basic Back-to-Back Aggregation Topology

In this figure, device `bge0` on System A is directly linked to `bge0` on System B, and so on. In this way, Systems A and B can support redundancy and high availability, as well as high-speed communications between both systems. Each system also has interface `ce0` configured for traffic flow within the local network.

The most common application for back-to-back link aggregations is mirrored database servers. Both servers need to be updated together and therefore require significant bandwidth, high-speed traffic flow, and reliability. The most common use of back-to-back link aggregations is in data centers.

## Policies and Load Balancing

If you plan to use a link aggregation, consider defining a policy for outgoing traffic. This policy can specify how you want packets to be distributed across the available links of an aggregation, thus establishing load balancing. The following are the possible layer specifiers and their significance for the aggregation policy:

- **L2** – Determines the outgoing link by hashing the MAC (L2) header of each packet
- **L3** – Determines the outgoing link by hashing the IP (L3) header of each packet
- **L4** – Determines the outgoing link by hashing the TCP, UDP, or other ULP (L4) header of each packet

Any combination of these policies is also valid. The default policy is L4. For more information, refer to the `dladm`(1M) man page.

## Aggregation Mode and Switches

If your aggregation topology involves connection through a switch, you must note whether the switch supports the *link aggregation control protocol (LACP)*. If the switch supports LACP, you must configure LACP for the switch and the aggregation. However, you can define one of the following *modes* in which LACP is to operate:

- **Off mode** – The default mode for aggregations. LACP packets, which are called *LACPDUs* are not generated.
- **Active mode** – The system generates LACPDUs at regular intervals, which you can specify.
- **Passive mode** – The system generates an LACPDU only when it receives an LACPDU from the switch. When both the aggregation and the switch are configured in passive mode, they cannot exchange LACPDUs.

See the `dladm`(1M) man page and the switch manufacturer's documentation for syntax information.

## Requirements for Link Aggregations

Your link aggregation configuration is bound by the following requirements:

- You must use the `dladm` command to configure aggregations.
- An interface that has been plumbed cannot become a member of an aggregation.
- Interfaces must be of the GLDv3 type: `xge`, `e1000g`, and `bge`.
- All interfaces in the aggregation must run at the same speed and in full-duplex mode.
- You must set the value for MAC addresses to "true" in the EEPROM parameter `local-mac-address?` For instructions, refer to "SPARC: How to Ensure That the MAC Address of an Interface Is Unique" on page 154.

## ▼ How to Create a Link Aggregation

**Before You Begin**

**Note –** Link aggregation only works on full-duplex, point-to-point links that operate at identical speeds. Make sure that the interfaces in your aggregation conform to this requirement.

If you are using a switch in your aggregation topology, make sure that you have done the following on the switch:

- Configured the ports to be used as an aggregation
- If the switch supports LACP, configured LACP in either active mode or passive mode

**1 Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Determine which interfaces are currently installed on your system.**

```
# dladm show-link
```

**3 Determine which interfaces have been plumbed.**

```
# ifconfig -a
```

**4 Create an aggregation.**

```
# dladm create-aggr -d interface key
```

*interface* Represents the device name of the interface to become part of the aggregation.

*key* Is the number that identifies the aggregation. The lowest key number is 1. Zeroes are not allowed as keys.

For example:

```
# dladm create-aggr -d bge0 -d bge1 1
```

**5 Configure and plumb the newly created aggregation.**

```
# ifconfig aggrkey plumb IP-address up
```

For example:

```
# ifconfig aggr1  plumb 192.168.84.14 up
```

**6 Check the status of the aggregation you just created.**

```
# dladm show-aggr
```

You receive the following output:

```
key: 1 (0x0001) policy: L4      address: 0:3:ba:7:84:5e (auto)
          device   address           speed         duplex  link    state
          bge0     0:3:ba:7:84:5e    1000  Mbps    full    up      attached
          bge1     0:3:ba:7:84:5e    0     Mbps    unknown down    standby
```

The output shows that an aggregation with the key of 1 and a policy of L4 was created. Note that the interfaces are known by the MAC address `0:3:ba:7:84:5e`, which is the system MAC address.

**7 (Optional) Make the IP configuration of the link aggregation persist across reboots.**

**a. For link aggregations with IPv4 addresses, create an `/etc/hostname.aggr.`*key* file. For IPv6–based link aggregations, create an `/etc/hostname6.aggr.`*key* file.**

**b. Enter the IPv4 or IPv6 address of the link aggregation into the file.**

For example, you would create the following file for the aggregation that is created in this procedure:

```
# vi /etc/hostname.aggr1
192.168.84.14
```

**c. Perform a reconfiguration boot.**

```
# reboot -- -r
```

**d. Verify that the link aggregation configuration you entered in the `/etc/hostname.aggr`*key* file has been configured.**

```
# ifconfig -a
.
.
aggr1: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.84.14 netmask ff000000 broadcast 192.255.255.
```

**Example 6–4** Creating a Link Aggregation

This example shows the commands that are used to create a link aggregation with two devices, `bge0` and `bge1`, and the resulting output.

```
# dladm show-link
ce0             type: legacy    mtu: 1500       device: ce0
ce1             type: legacy    mtu: 1500       device: ce1
bge0            type: non-vlan  mtu: 1500       device: bge0
bge1            type: non-vlan  mtu: 1500       device: bge1
bge2            type: non-vlan  mtu: 1500       device: bge2
# ifconfig -a
lo0: flags=2001000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
ce0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.84.253 netmask ffffff00 broadcast 192.168.84.255
        ether 0:3:ba:7:84:5e
# dladm create-aggr -d bge0 -d bge1 1
# ifconfig aggr1 plumb 192.168.84.14 up
# dladm show-aggr
key: 1 (0x0001) policy: L4      address: 0:3:ba:7:84:5e (auto)
     device   address           speed         duplex  link    state
```

```
     bge0     0:3:ba:7:84:5e    1000  Mbps    full    up      attached
     bge1     0:3:ba:7:84:5e    0     Mbps    unknown down    standby

# ifconfig -a
lo0: flags=2001000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
ce0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.84.253 netmask ffffff00 broadcast 192.168.84.255
        ether 0:3:ba:7:84:5e
aggr1: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 3
        inet 192.168.84.14 netmask ff000000 broadcast 192.255.255.255
        ether 0:3:ba:7:84:5e
```

Note that the two interfaces that were used for the aggregation were not previously plumbed by `ifconfig`.

## How to Modify an Aggregation

This procedure shows how to make the following changes to an aggregation definition:

- Modifying the policy for the aggregation
- Changing the mode for the aggregation

**1 Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Modify the aggregation to change the policy.**

**# dladm modify-aggr -P***policy key*

*policy* Represents one or more of the policies L2, L3, and L4, as explained in "Policies and Load Balancing" on page 164.

*key* Is a number that identifies the aggregation. The lowest key number is 1. Zeroes are not allowed as keys.

**3 If LACP is running on the switch to which the devices in the aggregation are attached, modify the aggregation to support LACP.**

If the switch runs LACP in passive mode, be sure to configure active mode for your aggregation.

**# dladm modify-aggr -l** *LACP mode* **-t** *timer-value key*

`-l` *LACP mode* Indicates the LACP mode in which the aggregation is to run. The values are `active`, `passive`, and `off`.

| | |
|---|---|
| `-t` *timer-value* | Indicates the LACP timer value, either `short` or `long`. |
| *key* | Is a number that identifies the aggregation. The lowest key number is 1. Zeroes are not allowed as keys. |

**Example 6–5** Modifying a Link Aggregation

This example shows how to modify the policy of aggregation `aggr1` `vidfeed0` to L2 and then turn on active LACP mode.

```
# dladm modify-aggr -P L2 1
# dladm modify-aggr -l active -t short 1
# dladm show-aggr
key: 1 (0x0001) policy: L2      address: 0:3:ba:7:84:5e (auto)
     device   address           speed         duplex  link    state
     bge0     0:3:ba:7:84:5e    1000  Mbps    full    up      attached
     bge1     0:3:ba:7:84:5e    0     Mbps    unknown down    standby
```

## ▼ How to Remove an Interface From an Aggregation

**1 Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Remove an interface from the aggregation.**

```
# dladm remove-aggr -d interface
```

**Example 6–6** Removing Interfaces From an Aggregation

This example shows how to remove the interfaces of the aggregation `aggr1`.

```
# dladm show-aggr
key: 1 (0x0001) policy: L2      address: 0:3:ba:7:84:5e (auto)
     device   address           speed         duplex  link    state
     bge0     0:3:ba:7:84:5e    1000  Mbps    full    up      attached
     bge1     0:3:ba:7:84:5e    0     Mbps    unknown down    standby
# dladm remove-aggr -d bge1 1
# dladm show-aggr
key: 1 (0x0001) policy: L2      address: 0:3:ba:7:84:5e (auto)
     device    address           speed         duplex  link    state
     bge0      0:3:ba:7:84:5e    1000  Mbps    full    up      attached
```

## ▼ How to Delete an Aggregation

**1 Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Delete the aggregation.**

```
# dladm remove-aggr key
```

*key* Is a number that identifies the aggregation. The lowest key number is 1. Zeroes are not allowed as keys.

**Example 6–7** How to Delete an Aggregation

This example shows how to remove the aggregation `aggr1`.

```
# dladm show-aggr
key: 1 (0x0001) policy: L2      address: 0:3:ba:7:84:5e (auto)
     device   address           speed     duplex  link    state
          # dladm remove-aggr -d bge0 1
```

◆ ◆ ◆ CHAPTER 7

# Configuring an IPv6 Network (Tasks)

This chapter contains tasks for configuring IPv6 on a network. The following major topics are covered:


- "Configuring an IPv6 Interface" on page 171
- "Enabling IPv6 on an Interface (Task Map)" on page 172
- "Configuring an IPv6 Router" on page 176
- "Modifying an IPv6 Interface Configuration for Hosts and Servers" on page 181
- "Modifying an IPv6 Interface Configuration (Task Map)" on page 181
- "Configuring Tunnels for IPv6 Support" on page 189
- "Tasks for Configuring Tunnels for IPv6 Support (Task Map)" on page 188
- "Configuring Name Service Support for IPv6" on page 197


For an overview of IPv6 concepts, refer to Chapter 3, "Introducing IPv6 (Overview)." For IPv6 planning tasks, refer to Chapter 4, "Planning an IPv6 Network (Tasks)." . To find reference information about the tasks in this chapter, refer to Chapter 11, "IPv6 in Depth (Reference)."

## Configuring an IPv6 Interface

The initial step in IPv6 configuration is enabling IPv6 on an interface. You can enable IPv6 support during the Solaris 10 installation process or by configuring IPv6 on the interfaces of an installed system.

During the Solaris 10 installation process, you can enable IPv6 on one or more of a system's interfaces. After installation, the following IPv6-related files and tables are in place:

- Each interface that was enabled for IPv6 now has an associated `/etc/hostname6.`*interface* file, such as `hostname6.dmfe0`.
- For Solaris 10 11/06 and earlier releases, the `/etc/inet/ipnodes` file has been created. After installation, this file typically contains only the IPv6 and IPv4 loopback addresses.
- The `/etc/nsswitch.conf` file has been modified to accommodate lookups using IPv6 addresses.

- The IPv6 address selection policy table is created. This table prioritizes the IP address format to use for transmissions over an IPv6-enabled interface.

This section describes how to enable IPv6 on the interfaces of an installed system.

## Enabling IPv6 on an Interface (Task Map)

| Task | Description | For Instructions |
|---|---|---|
| Enable IPv6 on an interface on a system that has already been installed with the Solaris 10 OS. | Use this task for enabling IPv6 on an interface after the Solaris 10 OS has been installed. | "How to Enable an IPv6 Interface for the Current Session" on page 172 |
| Make the IPv6-enabled interface persist across reboots. | Use this task to make the IPv6 address of the interface permanent. | "How to Enable Persistent IPv6 Interfaces" on page 174 |
| Turn off IPv6 address autoconfiguration. | Use this task if you need to manually configure the interface ID portion of the IPv6 address. | "How to Turn Off IPv6 Address Autoconfiguration" on page 176 |

## ▼ How to Enable an IPv6 Interface for the Current Session

Begin your IPv6 configuration process by enabling IPv6 on the interfaces of all systems that will become IPv6 nodes. Initially, the interface obtains its IPv6 address through the autoconfiguration process, as described in "IPv6 Address Autoconfiguration" on page 81. You then can tailor the node's configuration based on its function in the IPv6 network, either as a host, server, or router.

---

**Note –** If the interface is on the same link as a router that currently advertises an IPv6 prefix, the interface obtains that site prefix as part of its autoconfigured addresses. For more information, refer to "How to Configure an IPv6-Enabled Router" on page 177.

---

The following procedure explains how to enable IPv6 for an interface that was added after Solaris 10 installation.

**Before You Begin** Complete the planning tasks for the IPv6 network, such as upgrading hardware and software, and preparing an addressing plan. For more information, see "IPv6 Planning (Task Maps)" on page 83.

1. **Log in to the prospective IPv6 node as Primary Administrator or as superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Enable IPv6 on an interface.**

   ```
   # ifconfig inet6 interface plumb up
   ```

3. **Start the IPv6 daemon`in.ndpd`.**

   ```
   # /usr/lib/inet/in.ndpd
   ```

---

**Note –** You can display the status of a node's IPv6-enabled interfaces by using the `ifconfig -a6` command.

---

**Example 7–1** Enabling an IPv6 Interface After Installation

This example shows how to enable IPv6 on the `qfe0` interface. Before you begin, check the status of all interfaces configured on the system.

```
# ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000863 <UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,IPv4> mtu 1500
           index 2
        inet 172.16.27.74 netmask ffffff00 broadcast 172.16.27.255
        ether 0:3:ba:13:14:e1
```

Only the `qfe0` interface is currently configured for this system. Enable IPv6 on this interface as follows:

```
# ifconfig inet6 qfe0 plumb up
# /usr/lib/inet/in.ndpd
# ifconfig -a6
lo0: flags=2000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841 <UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 0:3:ba:13:14:e1
        inet6 fe80::203:baff:fe13:14e1/10
```

The example shows the status of the system's interface before and after `qfe0`becomes IPv6-enabled. The `-a6` option of `ifconfig` shows just the IPv6 information for `qfe0` and the loopback interface. Note that the output indicates that only a link-local address was configured for `qfe0`, `fe80::203:baff:fe13:14e1/10`. This address indicates that as of yet no router on the node's local link advertises a site prefix.

After IPv6 is enabled, you can use the `ifconfig -a` command to display both IPv4 and IPv6 addresses for all interfaces on a system.

**Next Steps**

- To configure the IPv6 node as a router, go to "Configuring an IPv6 Router" on page 176.
- To maintain the IPv6 interface configuration across reboots, see "How to Enable Persistent IPv6 Interfaces" on page 174.
- To disable address autoconfiguration on the node, see "How to Turn Off IPv6 Address Autoconfiguration" on page 176.
- To tailor the node as a server, see the suggestions in "Administering IPv6-Enabled Interfaces on Servers" on page 187.

## ▼ How to Enable Persistent IPv6 Interfaces

This procedure explains how to enable IPv6 interfaces with autoconfigured IPv6 addresses that persist across subsequent reboots.

---

**Note –** If the interface is on the same link as a router that currently advertises an IPv6 prefix, the interface obtains that site prefix as part of its autoconfigured addresses. For more information, refer to "How to Configure an IPv6-Enabled Router" on page 177.

---

**1 Log in to the IPv6 node as Primary Administrator or as superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Create IPv6 addresses for interfaces that were added after installation.**

```
# touch /etc/hostname6.interface
```

**3 (Optional) Create an `/etc/inet/ndpd.conf` file that defines parameters for interface variables on the node.**

If you need to create temporary addresses for the host's interface, refer to "Using Temporary Addresses for an Interface" on page 181. For details about `/etc/inet/ndpd.conf`, refer to the `ndpd.conf`(4) man page and "`ndpd.conf` Configuration File" on page 263.

**4 Reboot the node.**

```
# reboot -- -r
```

The reboot process sends router discovery packets. If a router responds with a site prefix, the node can configure any interface with a corresponding `/etc/hostname6.`*interface* file with a global IPv6 address. Otherwise, the IPv6-enabled interfaces are configured solely with link-local addresses. Rebooting also restarts `in.ndpd` and other network daemons in IPv6 mode.

**Example 7–2** Making an IPv6 Interface Persist Across Reboots

This example shows how to make the IPv6 configuration for the `qfe0` interface persist across reboots. In this example, a router on the local link advertises the site prefix and subnet ID `2001:db8:3c4d:15/64`.

First, check the status of the system's interfaces.

```
# ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1000863 <UP,BROADCAST,NOTRAILERS,RUNNING,MULTICAST,IPv4> mtu 1500
           index 2
        inet 172.16.27.74 netmask ffffff00 broadcast 172.16.27.255
        ether 0:3:ba:13:14:e1

# touch /etc/hostname6.qfe0
# reboot -- -r
```

Verify that the IPv6 address you configured is still applied to the `qfe0` interface.

```
# ifconfig -a6
qfe0: flags=2000841 <UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
       ether 0:3:ba:13:14:e1
       inet6 fe80::203:baff:fe13:14e1/10
 qfe0:1: flags=2180841 <UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500
          index 2
        inet6 2001:db8:3c4d:15:203:baff:fe13:14e1/64
```

The output of `ifconfig -a6` shows two entries for `qfe0`. The standard `qfe0` entry includes the MAC address and the link-local address. A second entry, `qfe0:1`, indicates that a pseudo-interface was created for the additional IPv6 address on the `qfe0` interface. The new, global IPv6 address, `2001:db8:3c4d:15:203:baff:fe13:14e1/64`, includes the site prefix and subnet ID advertised by the local router.

**Next Steps**

- To configure the new IPv6 node as a router, go to "Configuring an IPv6 Router" on page 176.
- To disable address autoconfiguration on the node, see "How to Turn Off IPv6 Address Autoconfiguration" on page 176.
- To tailor the new node as a server, see the suggestions in "Administering IPv6-Enabled Interfaces on Servers" on page 187.

### ▼ How to Turn Off IPv6 Address Autoconfiguration

You normally should use address autoconfiguration to generate the IPv6 addresses for the interfaces of hosts and servers. However, sometimes you might want to turn off address autoconfiguration, especially if you want to manually configure a token, as explained in "Configuring an IPv6 Token" on page 185.

1. **Log in to the IPv6 node as Primary Administrator or as superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Create an `/etc/inet/ndpd.conf` file for the node.**

   The `/etc/inet/ndpd.conf` file defines interface variables for the particular node. This file should have the following contents in order to turn off address autoconfiguration for all of the server's interfaces:

   ```
   if-variable-name StatelessAddrConf false
   ```

   For details about `/etc/inet/ndpd.conf`, refer to the `ndpd.conf`(4) man page and "`ndpd.conf` Configuration File" on page 263.

3. **Update the IPv6 daemon with your changes.**

   ```
   # pkill -HUP in.ndpd
   ```

# Configuring an IPv6 Router

The first step in configuring IPv6 on a network is configuring IPv6 on a router. Router configuration involves a number of discrete tasks, which are described in this section. You might perform some or all of the tasks, depending on your site requirements.

## IPv6 Router Configuration (Task Map)

Perform the next tasks in the order that is shown in the table.

| Task | Description | For Instructions |
|---|---|---|
| 1. Ensure that you have completed the required prerequisites before you begin IPv6 configuration. | You must complete planning tasks and Solaris installation with IPv6 enabled interfaces before you configure an IPv6-enabled router. | Chapter 4, "Planning an IPv6 Network (Tasks)," and "Configuring an IPv6 Interface" on page 171. |

| Task | Description | For Instructions |
|---|---|---|
| 2. Configure a router. | Define the site prefix for the network. | "How to Configure an IPv6-Enabled Router" on page 177 |
| 3. Configure tunnel interfaces on the router. | Set up a manual tunnel or a 6to4 tunnel interface on the router. The local IPv6 network needs tunnels to communicate with other, isolated IPv6 networks. | ■ "How to Configure a 6to4 Tunnel" on page 192<br>■ "How to Manually Configure IPv6 Over IPv4 Tunnels" on page 189<br>■ "How to Manually Configure IPv6 Over IPv6 Tunnels" on page 190<br>■ "How to Configure IPv4 Over IPv6 Tunnels" on page 191 |
| 4. Configure the switches on the network. | If your network configuration includes switches, configure them for IPv6 at this point in the configuration process. | Refer to switch manufacturer's documentation. |
| 5. Configure any hubs on your network. | If your network configuration includes hubs, configure them for IPv6 at this point in the configuration process. | Refer to hub manufacturer's documentation. |
| 6. Configure the network name service for IPv6. | Configure your primary name service (DNS, NIS, or LDAP) to recognize IPv6 addresses after the router is configured for IPv6. | "How to Add IPv6 Addresses to DNS" on page 197 |
| 7. (Optional) Modify the addresses for the IPv6-enabled interfaces on hosts and servers. | After IPv6 router configuration, make further modifications on IPv6-enabled hosts and servers. | "Modifying an IPv6 Interface Configuration for Hosts and Servers" on page 181 |
| Configure applications to support IPv6 | Different applications might require different actions in order to support IPv6. | Refer to applications' documentation |

## ▼ How to Configure an IPv6-Enabled Router

This procedure assumes that all interfaces of the router were configured for IPv6 during Solaris installation.

**1 On the system that will become the IPv6 router, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Review which interfaces on the router were configured for IPv6 during installation.**

```
# ifconfig -a
```

Check the output to ensure that the interfaces that you wanted to configure for IPv6 are now plumbed with link-local addresses. The following sample command output of `ifconfig -a` shows the IPv4 and IPv6 addresses that were configured for the router's interfaces.

```
o0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
dmfe0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 172.16.26.232 netmask ffffff00 broadcast 172.16.26.255
        ether 0:3:ba:11:b1:15
dmfe1: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4 mtu 1500 index 3
        inet 172.16.26.220 netmask ffffff00 broadcast 172.16.26.255
        ether 0:3:ba:11:b1:16
lo0: flags=2000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
dmfe0: flags=2000841 <UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 0:3:ba:11:b1:15
        inet6 fe80::203:baff:fe11:b115/10
dmfe1: flags=2000841 <UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 3
        ether 0:3:ba:11:b1:16
        inet6 fe80::203:baff:fe11:b116/10
```

The output also shows that the primary network interface `dmfe0` and the additional interface `dmfe1` were configured during installation with the IPv6 link–local addresses `fe80::203:baff:fe11:b115/10` and `fe80::203:baff:fe11:b116/10`.

**3 Configure IPv6 packet forwarding on all interfaces of the router.**

For Solaris 10 11/03 and earlier releases, use the following command:

```
# routeadm -e ipv6-forwarding -u
```

Use either of the following to enable packet forwarding:

- Use the `routeadm` command, as follows:

  ```
  # routeadm -e ipv6-forwarding -u
  ```

- Use the following Service Management Facility (SMF) command, as follows:

  ```
  # svcadm enable ipv6-forwarding
  ```

**4 Start the routing daemon.**

The `in.ripngd` daemon handles IPv6 routing.

For Solaris 10 11/06 and earlier releases, start `in.ripngd` by typing the following command:

```
# routeadm -e ipv6-routing
# routeadm -u
```

Turn on IPv6 routing in either of the following ways:

- Use the `routeadm` command as follows:

  ```
  # routeadm -e ipv6-routing -u
  ```

- Use SMF to enable IPv6 routing:

  ```
  # svcadm enable ripng:default
  ```

For syntax information on the `routeadm` command, see the `routeadm`(1M) man page.

**5 Create the `/etc/inet/ndpd.conf` file.**

You specify the site prefix to be advertised by the router and other configuration information in `/etc/inet/ndpd.conf`. This file is read by the `in.ndpd` daemon, which implements the IPv6 Neighbor Discovery protocol.

For a list of variables and allowable values, refer to "`ndpd.conf` Configuration File" on page 263 and the `ndpd.conf`(4)man page.

**6 Type the following text into the `/etc/inet/ndpd.conf` file:**

```
ifdefault AdvSendAdvertisements true
prefixdefault AdvOnLinkFlag on AdvAutonomousFlag on
```

This text tells the `in.ndpd` daemon to send out router advertisements over all interfaces of the router that are configured for IPv6.

**7 Add additional text to the `/etc/inet/ndpd.conf` file to configure the site prefix on the various interfaces of the router.**

The text should have the following format:

`prefix` *global-routing-prefix*`:`*subnet ID*`/64` *interface*

The following sample `/etc/inet/ndpd.conf` file configures the router to advertise the site prefix `2001:0db8:3c4d::/48` over the interfaces `dmfe0` and `dmfe1`.

```
ifdefault AdvSendAdvertisements true
prefixdefault AdvOnLinkFlag on AdvAutonomousFlag on

if dmfe0 AdvSendAdvertisements 1
prefix 2001:0db8:3c4d:15::0/64 dmfe0

if dmfe1 AdvSendAdvertisements 1
prefix 2001:0db8:3c4d:16::0/64 dmfe1
```

**8 Reboot the system.**

The IPv6 router begins advertising on the local link any site prefix that is in the `ndpd.conf` file.

**Example 7–3** `ifconfig` Output Showing IPv6 Interfaces

The following example shows output from the `ifconfig -a` command such as you would receive after you finish the "Configuring an IPv6 Router" on page 176 procedure.

```
lo0: flags=1000849 <UP LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
dmfe0: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 172.16.15.232 netmask ffffff00 broadcast 172.16.26.255
        ether 0:3:ba:11:b1:15
dmfe1: flags=1000843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4 mtu 1500 index 3
        inet 172.16.16.220 netmask ffffff00 broadcast 172.16.26.255
        ether 0:3:ba:11:b1:16
lo0: flags=2000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
dmfe0: flags=2100841 <UP,RUNNING,MULTICAST,ROUTER,IPv6> mtu 1500 index 2
        ether 0:3:ba:11:b1:15
        inet6 fe80::203:baff:fe11:b115/10
dmfe0:1: flags=2180841 <UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500
          index 2
        inet6 2001:db8:3c4d:15:203:baff:fe11:b115/64
dmfe1: flags=2100841 <UP,RUNNING,MULTICAST,ROUTER,IPv6> mtu 1500 index 3
        ether 0:3:ba:11:b1:16
        inet6 fe80::203:baff:fe11:b116/10
dmfe1:1: flags=2180841 <UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6> mtu 1500
           index 3
        inet6 2001:db8:3c4d:16:203:baff:fe11:b116/64
```

In this example, each interface that was configured for IPv6 now has two addresses. The entry with the name of the interface, such as `dmfe0`, shows the link-local address for that interface. The entry with the form *interface:n*, such as `dmfe0:1`, shows a global IPv6 address. This address includes the site prefix that you configured in the `/etc/ndpd.conf` file, in addition to the interface ID.

**See Also**

- To configure any tunnels from the routers that you have identified in your IPv6 network topology, refer to "Configuring Tunnels for IPv6 Support" on page 189.
- For information about configuring switches and hubs on your network, refer to the manufacturer's documentation.
- To configure IPv6 hosts, refer to "Modifying an IPv6 Interface Configuration for Hosts and Servers" on page 181.
- To improve IPv6 support on servers, refer to "Administering IPv6-Enabled Interfaces on Servers" on page 187.

- For detailed information about IPv6 commands, files, and daemons, refer to "Solaris 10 IPv6 Implementation" on page 263.

# Modifying an IPv6 Interface Configuration for Hosts and Servers

This section explains how to modify the configuration of IPv6-enabled interfaces on nodes that are hosts or servers. In most instances, you should use address autoconfiguration for IPv6-enabled interfaces, as explained in "Stateless Autoconfiguration Overview" on page 81. However, you can modify the IPv6 address of an interface, if necessary, as explained in the tasks of this section.

## Modifying an IPv6 Interface Configuration (Task Map)

| Task | Description | For Instructions |
|---|---|---|
| Turn off IPv6 address autoconfiguration. | Use this task if you need to manually configure the interface ID portion of the IPv6 address. | "How to Turn Off IPv6 Address Autoconfiguration" on page 176 |
| Create a temporary address for a host. | Hide a host's interface ID by configuring a randomly created temporary address that is used as the lower 64 bits of the address. | "How to Configure a Temporary Address" on page 182 |
| Configure a token for the interface ID of a system. | Create a 64-bit token to be used as the interface ID in an IPv6 address. | "How to Configure a User-Specified IPv6 Token" on page 185 |

## Using Temporary Addresses for an Interface

An IPv6 *temporary address* includes a randomly generated 64-bit number as the interface ID, instead of an interface's MAC address. You can use temporary addresses for any interfaces on an IPv6 node that you want to keep anonymous. For example, you might want to use temporary addresses for the interfaces of a host that needs to access public web servers. Temporary addresses implement IPv6 privacy enhancements. These enhancements are described in RFC 3041, available at "Privacy Extensions for Stateless Address Autoconfiguration in IPv6" (`http://www.ietf.org/rfc/rfc3041.txt?number=3041`).

You enable a temporary address in the `/etc/inet/ndpd.conf` file for one or more interfaces, if needed. However, unlike standard, autoconfigured IPv6 addresses, a temporary address consists of the 64-bit subnet prefix and a randomly generated 64-bit number. This random

number becomes the interface ID segment of the IPv6 address. A link-local address is not generated with the temporary address as the interface ID.

Be aware that temporary addresses have a default *preferred lifetime* of one day. When you enable temporary address generation, you may also configure the following variables in the `/etc/inet/ndpd.conf` file:

| | |
|---|---|
| *valid lifetime*<br>TmpValidLifetime | Time span in which the temporary address exists, after which the address is deleted from the host. |
| *preferred lifetime*<br>TmpPreferredLifetime | Elapsed time before the temporary address is deprecated. This time span should be shorter than the valid lifetime. |
| *address regeneration* | Duration of time before the expiration of the preferred lifetime, during which the host should generate a new temporary address. |

You express the duration of time for temporary addresses as follows:

| | |
|---|---|
| *n* | *n* number of seconds, which is the default |
| *n* `h` | *n* number of hours (`h`) |
| *n* `d` | *n* number of days (`d`) |

## ▼ How to Configure a Temporary Address

**1 Log in to the IPv6 host as Primary Administrator or as superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 If necessary, enable IPv6 on the host's interfaces**

Refer to "How to Enable an IPv6 Interface for the Current Session" on page 172.

**3 Edit the `/etc/inet/ndpd.conf` file to turn on temporary address generation.**

- To configure temporary addresses on all interfaces of a host, add the following line to `/etc/inet/ndpd.conf`:

  ```
  ifdefault TmpAddrsEnabled true
  ```

- To configure a temporary address for a specific interface, add the following line to `/etc/inet/ndpd.conf`:

  ```
  if interface TmpAddrsEnabled true
  ```

4 **(Optional) Specify the valid lifetime for the temporary address.**

```
ifdefault TmpValidLifetime duration
```

This syntax specifies the valid lifetime for all interfaces on a host. The value for *duration* should be in seconds, hours, or days. The default valid lifetime is 7 days. You can also use `TmpValidLifetime` with the `if` *interface* keywords to specify the valid lifetime for a temporary address of a particular interface.

5 **(Optional) Specify a preferred lifetime for the temporary address, after which the address is deprecated.**

```
if interface TmpPreferredLifetime duration
```

This syntax specifies the preferred lifetime for the temporary address of a particular interface. The default preferred lifetime is one day. You can also use `TmpPreferredLifetime` with the `ifdefault` keyword to specify the preferred lifetime for the temporary addresses on all interfaces of a host.

---

**Note –** Default address selection gives a lower priority to IPv6 addresses that have been deprecated. If an IPv6 temporary address is deprecated, default address selection chooses a nondeprecated address as the source address of a packet. A nondeprecated address could be the automatically generated IPv6 address, or possibly, the interface's IPv4 address. For more information about default address selection, see "Administering Default Address Selection" on page 225.

---

6 **(Optional) Specify the lead time in advance of address deprecation, during which the host should generate a new temporary address.**

```
ifdefault TmpRegenAdvance duration
```

This syntax specifies the lead time in advance of address deprecation for the temporary addresses of all interfaces on a host. The default is 5 seconds.

7 **Change the configuration of the `in.ndpd` daemon.**

```
# pkill -HUP in.ndpd
# /usr/lib/inet/in.ndpd
```

8 **Verify that temporary addresses have been created by running the `ifconfig -a6` command, as shown in Example 7–5.**

The output from `ifconfig` should have the word `TEMPORARY` in the same line as the interface definition.

**Example 7–4** Temporary Address Variables in the `/etc/inet/ndpd.conf` File

The following example shows a segment of an `/etc/inet/ndpd.conf` file with temporary addresses enabled for the primary network interface.

```
ifdefault TmpAddrsEnabled true

ifdefault TmpValidLifetime 14d

ifdefault TmpPreferredLifetime 7d

ifdefault TmpRegenAdvance 6s
```

**Example 7–5** `ifconfig -a6` Command Output with Temporary Addresses Enabled

This example shows the output of the `ifconfig` command after temporary addresses are created.

```
# ifconfig -a6
lo0: flags=2000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
     inet6 ::1/128
hme0: flags=2000841 <UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
     ether 8:0:20:b9:4c:54
     inet6 fe80::a00:20ff:feb9:4c54/10
hme0:1: flags=2080841 <UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
     inet6 2001:db8:3c4d:15:a00:20ff:feb9:4c54/64
hme0:2: flags=802080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6,TEMPORARY> mtu 1500 index 2
      inet6 2001:db8:3c4d:15:7c37:e7d1:fc9c:d2cb/64
```

Note that the line following interface `hme0:2` includes the word `TEMPORARY`. This designation indicates that the address `2001:db8:3c4d:15:7c37:e7d1:fc9c:d2cb/64` has a temporary interface ID.

**See Also**

- To set up name service support for IPv6 addresses, see "Configuring Name Service Support for IPv6" on page 197.
- To configure IPv6 addresses for a server, see "How to Configure a User-Specified IPv6 Token" on page 185.
- To monitor activities on IPv6 nodes, see Chapter 8, "Administering a TCP/IP Network (Tasks)."

# Configuring an IPv6 Token

The 64-bit interface ID of an IPv6 address is also referred to as a *token*, as introduced in "IPv6 Addressing Overview" on page 74. During address autoconfiguration, the token is associated with the interface's MAC address. In most cases, nonrouting nodes, that is IPv6 hosts and servers, should use their autoconfigured tokens.

However, using autoconfigured tokens can be a problem for servers whose interfaces are routinely swapped as part of system maintenance. When the interface card is changed, the MAC address is also changed. Servers that depend on having stable IP addresses can experience problems as a result. Various parts of the network infrastructure, such as DNS or NIS, might have stored specific IPv6 addresses for the interfaces of the server.

To avoid address change problems, you can manually configure a token to be used as the interface ID in an IPv6 address. To create the token, you specify a hexadecimal number of 64 bits or less to occupy the interface ID portion of the IPv6 address. During subsequent address autoconfiguration, Neighbor Discovery does not create an interface ID that is based on the interface's MAC address. Instead, the manually created token becomes the interface ID. This token remains assigned to the interface, even when a card is replaced.

---

**Note –** The difference between user-specified tokens and temporary addresses is that temporary addresses are randomly generated, rather than explicitly created by a user.

---

## ▼ How to Configure a User-Specified IPv6 Token

The next instructions are particularly useful for servers whose interfaces are routinely replaced. They also are valid for configuring user-specified tokens on any IPv6 node.

**1 Assume the Primary Administrator role or become superuser on the node.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Verify that the interface you want to configure with a token is plumbed.**

An interface must be plumbed before you can configure a token for its IPv6 address.

```
# ifconfig -a6

qfe0: flags=2000841 <UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 0:3:ba:13:14:e1
        inet6 fe80::203:baff:fe13:14e1/10
```

This output shows that the network interface `qfe0` is plumbed and has the link-local address `fe80::203:baff:fe13:14e1/10`. This address was automatically configured during installation.

**3 Create one or more 64-bit hexadecimal numbers to be used as tokens for the node's interfaces. For examples of tokens, refer to "Link-Local Unicast Address" on page 78.**

**4 Configure each interface with a token.**

Use the following form of the `ifconfig` command for each interface to have a user-specified interface ID (token):

`ifconfig` *interface* `inet6 token` *address/64*

For example, you would use the following command to configure interface `qfe0` with a token:

```
# ifconfig qfe0 inet6 token ::1a:2b:3c:4d/64
```

Repeat this step for every interface that will have a user-specified token.

**5 (Optional) Make the new IPv6 address persist across reboots.**

**a. Edit or create an `/etc/hostname6.`*interface* file for each interface you configured with a token.**

**b. Add the following text at the bottom of each `/etc/hostname.6`*interface* file:**

**`token ::`***token-name*`/64`**

For example, you might add the following text to the bottom of an`/etc/hostname6.`*interface* file:

```
token ::1a:2b:3c:4d/64
```

After the system reboots, the token that you configured in an `/etc/hostname6.`*interface* file is applied to the interface's IPv6 address. This IPv6 address remains persistent across subsequent reboots.

**6 Update the IPv6 daemon with your changes.**

```
# pkill -HUP -in.ndpd
```

### Example 7–6 Configuring a User-Specified Token on an IPv6 Interface

In the following example, the interface `bge0:1` has an autoconfigured IPv6 address. The subnet prefix `2001:db8:3c4d:152:/64` is advertised by a router on the node's local link. The interface ID `2c0:9fff:fe56:8255` is generated from `bge0:1`'s MAC address.

```
# ifconfig -a6
lo0: flags=2002000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv6,VIRTUAL> mtu 8252 index 1
        inet6 ::1/128
bge0: flags=2100801 <UP,MULTICAST,IPv6> mtu 1500 index 5
        inet6 fe80::2c0:9fff:fe56:8255/10
        ether 0:c0:9f:56:82:55
bge0:1: flags=2180801 <UP, MULTICAST,ADDRCONF,IPv6>mtu 1500 index 5
        inet6 2001:db8:3c4d:152:c0:9fff:fe56:8255/64
# ifconfig bge0 inet6 token ::1a:2b:3c:4d/64
# vi /etc/hostname6.bge0
token ::1a:2b:3c:4d/64
# pkill -HUP -in.ndpd
# ifconfig -a6
lo0: flags=2002000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv6,VIRTUAL> mtu 8252 index 1
        inet6 ::1/128
bge0: flags=2100801 <UP,MULTICAST,IPv6> mtu 1500 index 5
        inet6 fe80::2c0:9fff:fe56:8255/10
        ether 0:c0:9f:56:82:55
bge0:1: flags=2180801 <UP, MULTICAST,ADDRCONF,IPv6>mtu 1500 index 5
        inet6 2001:db8:3c4d:152:1a:2b:3c:4d/64
```

After the token is configured, the global address on the second status line of `bge0:1` now has `1a:2b:3c:4d`configured for its interface ID.

**See Also**

- To update the name services with the IPv6 addresses of the server, see "Configuring Name Service Support for IPv6" on page 197.
- To monitor server performance, see Chapter 8, "Administering a TCP/IP Network (Tasks)."

# Administering IPv6-Enabled Interfaces on Servers

When you plan for IPv6 on a server, you must make a few decisions as you enable IPv6 on the server's interfaces. Your decisions affect the strategy to use for configuring the interface IDs, also known as *tokens*, of an interface's IPv6 address.

## ▼ How to Enable IPv6 on a Server's Interfaces

**Before You Begin**

The next procedure assumes the following:

- Solaris 10 OS is already installed on the server.
- You enabled IPv6 on the server's interfaces either during Solaris OS installation or later, using the procedures in "Configuring an IPv6 Interface" on page 171.

If applicable, upgrade the application software to support IPv6. Note that many applications that run on the IPv4 protocol stack also successfully run on IPv6. For more information, refer to "How to Prepare Network Services for IPv6 Support" on page 88.

1 **On the server, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Ensure that an IPv6 subnet prefix is configured on a router on the same link as the server.**

For more information, refer to "Configuring an IPv6 Router" on page 176.

3 **Use the appropriate strategy for the interface ID for the server's IPv6-enabled interfaces.**

By default, IPv6 address autoconfiguration uses the MAC address of an interface when creating the interface ID portion of the IPv6 address. If the IPv6 address of the interface is well known, swapping one interface for another interface can cause problems. The MAC address of the new interface will be different. During address autoconfiguration, a new interface ID is generated.

- For an IPv6-enabled interface that you do not plan to replace, use the autoconfigured IPv6 address, as introduced in "IPv6 Address Autoconfiguration" on page 81.
- For IPv6-enabled interfaces that must appear anonymous outside the local network, consider using a randomly generated token for the interface ID. For instructions and an example, refer to "How to Configure a Temporary Address" on page 182.
- For IPv6-enabled interfaces that you plan to swap on a regular basis, create tokens for the interface IDs. For instructions and an example, refer to "How to Configure a User-Specified IPv6 Token" on page 185.

# Tasks for Configuring Tunnels for IPv6 Support (Task Map)

| Task | Description | For Instructions |
|---|---|---|
| Manually configure IPv6 over IPv4 tunnels. | Manually creates an IPv6 tunnel over a IPv4 network, a solution for reaching remote IPv6 networks within a larger, mostly IPv4 enterprise network. | "How to Manually Configure IPv6 Over IPv4 Tunnels" on page 189 |
| Manually configure IPv6 over IPv6 tunnels. | Manually configures an IPv6 tunnel over an IPv6 network, typically used within a large enterprise network. | "How to Manually Configure IPv6 Over IPv6 Tunnels" on page 190 |

| Task | Description | For Instructions |
|---|---|---|
| Manually configure IPv4 over IPv6 tunnels. | Manually configures an IPv4 tunnel over an IPv6 network, useful for large networks with both IPv4 and IPv6 networks. | "How to Configure IPv4 Over IPv6 Tunnels" on page 191 |
| Automatically configure IPv6 over IPv4 tunnels (6to4 tunnels). | Create an automatic, 6to4 tunnel, a solution for reaching an external IPv6 site over the Internet. | "How to Configure a 6to4 Tunnel" on page 192 |
| Configure a tunnel between a 6to4 router and a 6to4 relay router. | Enables a tunnel to a 6to4 relay router by using the `6to4relay` command. | "How to Configure a 6to4 Tunnel to a 6to4 Relay Router" on page 195 |

# Configuring Tunnels for IPv6 Support

IPv6 networks are often isolated entities within the larger IPv4 world. Nodes on your IPv6 network might need to communicate with nodes on isolated IPv6 networks, either within your enterprise or remotely. Typically, you configure a tunnel between IPv6 routers, although IPv6 hosts can also function as tunnel endpoints. For tunnel planning information, refer to "Planning for Tunnels in the Network Topology" on page 89.

You can set up automatically or manually configured tunnels for the IPv6 network. The Solaris IPv6 implementation supports the following types of tunnel encapsulation:

- IPv6 over IPv4 tunnels
- IPv6 over IPv6 tunnels
- IPv4 over IPv6 tunnels
- 6to4 tunnels

For conceptual descriptions of tunnels, see "IPv6 Tunnels" on page 285.

## ▼ How to Manually Configure IPv6 Over IPv4 Tunnels

This procedure describes how to set up a tunnel from an IPv6 node to a remote IPv6 node over an IPv4 network.

**1 Log in to the local tunnel endpoint as Primary Administrator or as superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Create the `/etc/hostname6.ip.tun`*n* file.**

where *n* represents the tunnel number, beginning at zero for the first tunnel. Then, add entries by following these substeps:

a. **Add the tunnel source address and the tunnel destination address.**

`tsrc` *IPv4-source-address* `tdst` *IPv4-destination-address* `up`

b. **(Optional) Add a logical interface for the source IPv6 address and the destination IPv6 addresses.**

`addif` *IPv6-source-address IPv6-destination-address*

Omit this substep if you want the address autoconfigured for this interface. You do not need to configure link-local addresses for your tunnel.

3 **Reboot the system.**

4 **Repeat this task on the opposite endpoint of the tunnel.**

**Example 7–7** Entry in the `/etc/hostname6.ip.tun` File for a Manual, IPv6 Over IPv4 Tunnel

This sample `/etc/hostname6.ip.tun` file shows a tunnel for which global source addresses and global destination addresses are manually configured.

```
tsrc 192.168.8.20 tdst 192.168.7.19 up
addif 2001:db8:3c4d:8::fe12:528 2001:db8:3c4d:7:a00:20ff:fe12:1234 up
```

## ▼ How to Manually Configure IPv6 Over IPv6 Tunnels

This procedure describes how to set up a tunnel from an IPv6 node to a remote IPv6 node over an IPv6 network.

1 **Log in to the local tunnel endpoint as Primary Administrator or as superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Create the `/etc/hostname6.ip6.tun` *n* file.**

Use the values 0, 1, 2, and so on, for *n*. Then, add entries by following these substeps.

a. **Add the tunnel source address and the tunnel destination address.**

`tsrc` *IPv6-source-address* `tdst` *IPv6-destination-address*
*IPv6-packet-source-address IPv6-packet-destination-address* `up`

**b. (Optional) Add a logical interface for the source IPv6 address and destination IPv6 address.**

`addif` *IPv6-source-address IPv6-destination-address* `up`

Omit this step if you want the address autoconfigured for this interface. You do not need to configure link-local addresses for your tunnel.

**3 Reboot the system.**

**4 Repeat this procedure at the opposite endpoint of the tunnel.**

**Example 7–8** Entry in the `/etc/hostname6.ip6.tun` File for an IPv6 Over IPv6 Tunnel

This example shows the entry for an IPv6 over IPv6 tunnel.

```
tsrc 2001:db8:3c4d:22:20ff:0:fe72:668c tdst 2001:db8:3c4d:103:a00:20ff:fe9b:a1c3
fe80::4 fe80::61 up
```

## ▼ How to Configure IPv4 Over IPv6 Tunnels

This procedure explains how to configure a tunnel between two IPv4 hosts over an IPv6 network. You would use this procedure if your corporate network is heterogeneous, with IPv6 subnets that separate IPv4 subnets.

**1 Log in to the local IPv4 tunnel endpoint as Primary Administrator or as superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Create the `/etc/hostname.ip6.tun`*n* file.**

Use the values 0, 1, 2, and so on, for *n*. Then, add entries by following these steps:

**a. Add the tunnel source address and the tunnel destination address.**

`tsrc` *IPv6-source-address* `tdst` *IPv6-destination-address*

**b. (Optional) Add a logical interface for the source IPv6 address and destination IPv6 address.**

`addif` *IPv6-source-address IPv6-destination-address* `up`

**3 Reboot the local host.**

**4 Repeat this procedure at the opposite endpoint of the tunnel.**

**Example 7–9** Entry in the `/etc/hostname6.ip6.tun` for an IPv4 Over IPv6 Tunnel

This example shows the entry for an IPv4 over IPv6 tunnel.

```
tsrc 2001:db8:3c4d:114:a00:20ff:fe72:668c tdst 2001:db8:3c4d:103:a00:20ff:fe9b:a1c3
10.0.0.4 10.0.0.61 up
```

## How to Configure a 6to4 Tunnel

If your IPv6 network needs to communicate with a remote IPv6 network, consider using automatic, 6to4 tunnels. The process of configuring a 6to4 tunnel includes configuring the boundary router as a *6to4* router. The 6to4 router functions as the endpoint of a 6to4 tunnel between your network and an endpoint router at a remote IPv6 network.

**Before You Begin** Before you configure 6to4 routing on an IPv6 network, you must have done the following:

- Configured IPv6 on all appropriate nodes at the prospective 6to4 site, as described in "Modifying an IPv6 Interface Configuration for Hosts and Servers" on page 181.
- Selected at least one router with a connection to an IPv4 network to become the 6to4 router.
- Configured a globally unique IPv4 address for the prospective 6to4 router's interface to the IPv4 network. The IPv4 address must be static.

---

**Note –** Do not use a dynamically allocated IPv4 address, as described in Chapter 12, "About Solaris DHCP (Overview)." Global dynamically allocated addresses might change over time, which can adversely affect your IPv6 addressing plan.

---

**1 Log in to the prospective 6to4 router as Primary Administrator or as superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Configure a 6to4 pseudo-interface on the router by creating the `/etc/hostname6.ip.6to4tun0` file.**

- If you plan to use the recommended convention of subnet ID=0 and host ID=1, use the short format for `/etc/hostname6.ip.6to4tun0`:

  `tsrc` *IPv4-address* `up`

- If you plan to use other conventions for the subnet ID and host ID, use the long format for `/etc/hostname6.ip.6to4tun0`:

  `tsrc` *IPv4-address* `2002:`*IPv4-address*`:`*subnet-ID*`:`*interface-ID*`:/64 up`

The required parameters for `/etc/hostname6.ip.6to4tun0` follow:

`tsrc` Indicates that this interface is used as a tunnel source.

*IPv4-address* Specifies, in dotted-decimal format, the IPv4 address that is configured on the physical interface to become the 6to4 pseudo-interface.

The remaining parameters are optional. However, if you specify one optional parameter, you must specify all optional parameters.

`2002` Specifies the 6to4 prefix.

*IPv4–address* Specifies, in hexadecimal notation, the IPv4 address of the pseudo-interface.

*subnet-ID* Specifies, in hexadecimal notation, a subnet ID other than 0.

*interface-ID* Specifies an interface ID other than 1.

`/64` Indicates that the 6to4 prefix has a length of 64 bits.

`up` Configures the 6to4 interface as "up."

---

**Note –** Two IPv6 tunnels on your network cannot have the same source address and the same destination address. Packets are dropped as a result. This type of event can happen if a 6to4 router also performs tunneling through the `atun` command. For information about `atun`, refer to the `tun`(7M) man page.

---

**3 (Optional) Create additional 6to4 pseudo-interfaces on the router.**

Each prospective 6to4 pseudo-interface must have an already configured, globally unique IPv4 address.

**4 Reboot the 6to4 router.**

**5 Verify the status of the interface.**

```
# ifconfig ip.6to4tun0 inet6
```

If the interface is correctly configured, you receive output that is similar to the following:

```
ip.6to4tun0: flags=2200041<UP,RUNNING,NONUD,IPv6>mtu 1480 index 11
        inet tunnel src 111.222.33.44
        tunnel hop limit 60
        inet6 2002:6fde:212c:10:/64
```

**6 Edit the `/etc/inet/ndpd.conf` file to advertise 6to4 routing.**

For detailed information, refer to the `ndpd.conf`(4) man page.

**a. Specify the subnet to receive the advertisement in the first line.**

Create an `if` entry with the following format:

`if` *subnet-interface* `AdvSendAdvertisements 1`

For example, to advertise 6to4 routing to the subnet that is connected to interface `hme0`, replace *subnet-interface* with `hme0`.

```
if hme0 AdvSendAdvertisements 1
```

**b. Add the 6to4 prefix as the second line of the advertisement.**

Create a `prefix` entry with following format:

`prefix 2002:`*IPv4-address*`:`*subnet-ID*`::/64` *subnet-interface*

**7 Reboot the router.**

Alternatively, you can issue a `sighup` to the `/etc/inet/in.ndpd` daemon to begin sending router advertisements. The IPv6 nodes on each subnet to receive the 6to4 prefix now autoconfigure with new 6to4-derived addresses.

**8 Add the new 6to4-derived addresses of the nodes to the name service that is used at the 6to4 site.**

For instructions, go to "Configuring Name Service Support for IPv6" on page 197.

### Example 7–10 6to4 Router Configuration (Short Form)

The following is an example of the short form of `/etc/hostname6.ip.6to4tun0`:

```
# cat /etc/hostname6.ip.6to4tun0
tsrc 111.222.33.44 up
```

### Example 7–11 6to4 Router Configuration (Long Form)

Here is an example of the long form of `/etc/hostname6.ip.6to4tun0`:

```
# cat /etc/hostname6.ip.6to4tun0
tsrc 111.222.33.44 2002:6fde:212c:20:1/64 up
```

### Example 7–12 `ifconfig` Output Showing 6to4 Pseudo-Interface

The following sample shows output of the `ifconfig` command for a 6to4 pseudo-interface:

```
# ifconfig ip.6to4tun0 inet6
ip.6to4tun0: flags=2200041<UP,RUNNING,NONUD,IPv6> mtu 1480 index 11
        inet tunnel src 192.168.87.188
        tunnel hop limit 60
        inet6 2002:c0a8:57bc::1/64
```

**Example 7–13** 6to4 Advertisements in `/etc/inet/ndpd.conf`

The following sample `/etc/inet/ndpd.conf` file advertises 6to4 routing on two subnets:

```
if qfe0 AdvSendAdvertisements 1
prefix  2002:c0a8:57bc:10::/64 qfe0

if qfe1 AdvSendAdvertisements 1
prefix  2002:c0a8:57bc:2::/64 qfe1
```

**More Information** Configuring Multiple Routers at the 6to4 Site

For a multiple router site, the routers behind the 6to4 router might require further configuration to support 6to4. If your site uses RIP, you must configure on each non-6to4 router the static routes to the 6to4 router. If you use a commercial routing protocol, you do not need to create static routes to the 6to4 router.

## ▼ How to Configure a 6to4 Tunnel to a 6to4 Relay Router

**Caution –** Because of major security issues, by default, 6to4 relay router support is disabled in the Solaris OS. See "Security Issues When Tunneling to a 6to4 Relay Router" on page 232.

**Before You Begin** Before you enable a tunnel to a 6to4 relay router, you must have completed the following tasks:

- Configured a 6to4 router at your site, as explained in "How to Configure a 6to4 Tunnel" on page 192
- Reviewed the security issues that are involved in tunneling to a 6to4 relay router

**1 Log in to the 6to4 router as Primary Administrator or as superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Enable a tunnel to the 6to4 relay router by using either of the following formats:**

- Enable a tunnel to an anycast 6to4 relay router.

  ```
  # /usr/sbin/6to4relay -e
  ```

  The `-e` option sets up a tunnel between the 6to4 router and an anycast 6to4 relay router. Anycast 6to4 relay routers have the well-known IPv4 address `192.88.99.1`. The anycast

relay router that is physically nearest to your site becomes the endpoint for the 6to4 tunnel. This relay router then handles packet forwarding between your 6to4 site and a native IPv6 site.

For detailed information about anycast 6to4 relay routers, refer to RFC 3068, "An Anycast Prefix for 6to4 Relay Routers" (`ftp://ftp.rfc-editor.org/in-notes/rfc3068.txt`).

- Enable a tunnel to a specific 6to4 relay router.

  ```
  # /usr/sbin/6to4relay -e -a relay-router-address
  ```

  The `-a` option indicates that a specific router address is to follow. Replace *relay-router-address* with the IPv4 address of the specific 6to4 relay router with which you want to enable a tunnel.

The tunnel to the 6to4 relay router remains active until you remove the 6to4 tunnel pseudo-interface.

**3 Delete the tunnel to the 6to4 relay router, when the tunnel is no longer needed:**

```
# /usr/sbin/6to4relay -d
```

**4 (Optional) Make the tunnel to the 6to4 relay router persistent across reboots.**

Your site might have a compelling reason to have the tunnel to the 6to4 relay router reinstated each time the 6to4 router reboots. To support this scenario, you must do the following:

**a. Edit the `/etc/default/inetinit` file.**

The line that you need to modify is at the end of the file.

**b. Change the "NO" value in the line `ACCEPT6TO4RELAY=NO` to "YES."**

**c. (Optional) Create a tunnel to a specific 6to4 relay router that persists across reboots.**

For the parameter `RELAY6TO4ADDR`, change the address `192.88.99.1` to the IPv4 address of the 6to4 relay router that you want to use.

**Example 7–14** Getting Status Information About 6to4 Relay Router Support

You can use the `/usr/bin/6to4relay` command to find out whether support for 6to4 relay routers is enabled. The next example shows the output when support for 6to4 relay routers is disabled, as is the default in the Solaris OS:

```
# /usr/sbin/6to4relay
6to4relay: 6to4 Relay Router communication support is disabled.
```

When support for 6to4 relay routers is enabled, you receive the following output:

```
# /usr/sbin/6to4relay
6to4relay: 6to4 Relay Router communication support is enabled.
IPv4 destination address of Relay Router=192.88.99.1
```

# Configuring Name Service Support for IPv6

This section describes how to configure the DNS and NIS name services to support IPv6 services.

**Note –** LDAP supports IPv6 without requiring IPv6-specific configuration tasks.

For full details for administering DNS, NIS, and LDAP, refer to the *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

## ▼ How to Add IPv6 Addresses to DNS

1. **Log in to the primary or secondary DNS server as Primary Administrator or as superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Edit the appropriate DNS zone file by adding AAAA records for each IPv6-enabled node:**

   *host-name* `IN AAAA` *host-address*

3. **Edit the DNS reverse zone file and add PTR records:**

   *host-address* `IN PTR` *hostname*

   For detailed information on DNS administration, refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

**Example 7–15** DNS Reverse Zone File

This example shows an IPv6 address in the reverse zone file.

```
$ORIGIN    ip6.int.
8.2.5.0.2.1.e.f.f.f.9.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.2.0.0.0 \
    IN        PTR        vallejo.Eng.apex.COM.
```

## Adding IPv6 Addresses to NIS

In Solaris 10 11/06 and earlier releases, two maps were added for NIS : `ipnodes.byname` and `ipnodes.byaddr`. These maps contained both IPv4 and IPv6 host name and address associations. Tools that are aware of IPv6 used the `ipnodes` NIS maps. The `hosts.byname` and `hosts.byaddr` maps contained only IPv4 host name and address associations. These maps are unchanged so that they can facilitate existing applications. Administration of the `ipnodes` maps is similar to the administration of the `hosts.byname` and `hosts.byaddr` maps. For Solaris 10 11/06, it is important that when you update the *hosts* maps with IPv4 addresses, the *ipnode* maps are also updated with the same information.

---

**Note –** Subsequent releases of Solaris 10 do not use the `ipnodes` maps. The IPv6 functionality of the `ipnodes` maps is now maintained in the `hosts` maps.

---

For instructions on administering NIS maps, refer to Chapter 5, "Setting Up and Configuring NIS Service," in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

## ▼ How to Display IPv6 Name Service Information

You can use the `nslookup` command to display IPv6 name service information.

1. **Under your user account, run the `nslookup` command.**

   ```
   % /usr/sbin/nslookup
   ```

   The default server name and address appear, followed by the `nslookup` command's angle bracket prompt.

2. **View information about a particular host by typing the following commands at the angle bracket prompt:**

   ```
   >set q=any
   >host-name
   ```

3. **Type the following command to view only AAAA records:**

   ```
   >set q=AAAA
   hostname
   ```

4. **Quit the `nslookup` command by typing `exit`.**

**Example 7–16** Using `nslookup` to Display IPv6 Information

This example shows the results of `nslookup` in an IPv6 network environment.

```
%  /usr/sbin/nslookup
Default Server:  dnsserve.local.com
Address:  10.10.50.85
> set q=AAAA
> host85
Server:  dnsserve.local.com
Address:  10.10.50.85

host85.local.com      IPv6 address = 2::9256:a00:fe12:528
> exit
```

## How to Verify That DNS IPv6 PTR Records Are Updated Correctly

In this procedure, you use the `nslookup` command to display PTR records for DNS IPv6.

1. **Under your user account, run the `nslookup` command.**

   ```
   % /usr/sbin/nslookup
   ```

   The default server name and address display, followed by the `nslookup` command's angle bracket prompt.

2. **Type the following at the angle bracket prompt to see the PTR records:**

   ```
   >set q=PTR
   ```

3. **Quit the command by typing `exit`.**

**Example 7–17** Using `nslookup` to Display PTR Records

The following example shows the PTR record display from the `nslookup` command.

```
%  /usr/sbin/nslookup
Default Server:  space1999.Eng.apex.COM
Address:  192.168.15.78
> set q=PTR
> 8.2.5.0.2.1.e.f.f.f.0.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.0.2.0.0.0.ip6.int

8.2.5.0.2.1.e.f.f.f.0.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.0.2.0.0.0.ip6.int name =
vallejo.ipv6.Eng.apex.COM
```

```
ip6.int nameserver = space1999.Eng.apex.COM
> exit
```

## How to Display IPv6 Information Through NIS

In this procedure, you use the `ypmatch` command to display IPv6 information through NIS:

- **Under your user account, type the following to display IPv6 addresses in NIS:**

  `% ypmatch` *hostname* `hosts ipnodes.`*byname*

  The information about the specified *hostname* displays.

---

**Note –** Solaris releases after Solaris 11/06 no longer include the `ipnodes` maps. The IPv6 functionality of `ipnodes` is now maintained in the `hosts` maps.

---

**Example 7–18** IPv6 Addresses Output by the `ypmatch` Command

For Solaris 10 11/06 and earlier releases, the following sample shows the results of a `ypmatch` operation on the `ipnodes.byname` database.

```
% ypmatch farhost hosts ipnodes.byname
2001:0db8:3c4d:15:a00:20ff:fe12:5286       farhost
```

## How to Display IPv6 Information Independent of the Name Service

This procedure can be used for Solaris 10 11/06 and earlier releases only. For subsequent releases, you can perform the same operation on the `hosts` database.

- **Under your user account, type the following command:**

  `% getent ipnodes` *hostname*

  The information about the specified *host-name* is displayed.

**Example 7–19** Displaying IPv6 Information in the `ipnodes` Database

The following sample shows the output of the `getent` command:

```
% getent ipnodes vallejo
```

```
2001:0db8:8512:2:56:a00:fe87:9aba    myhost myhost
fe80::56:a00:fe87:9aba     myhost myhost
```

◆ ◆ ◆ CHAPTER 8

# Administering a TCP/IP Network (Tasks)

This chapter contains tasks for administering a TCP/IP network. The following topics are covered:


- "Major TCP/IP Administrative Tasks (Task Map)" on page 203
- "Monitoring the Interface Configuration With the `ifconfig` Command" on page 204
- "Monitoring Network Status With the `netstat` Command" on page 208
- "Probing Remote Hosts With the `ping` Command" on page 216
- "Administering and Logging Network Status Displays" on page 217
- "Displaying Routing Information With the `traceroute` Command" on page 220
- "Monitoring Packet Transfers With the `snoop` Command" on page 222
- "Administering Default Address Selection" on page 225


The tasks assume that you have an operational TCP/IP network at your site, either IPv4-only or dual-stack IPv4/IPv6. If you want to implement IPv6 at your site but have not done so, refer to following chapters for more information:

- To plan an IPv6 implementation, refer to Chapter 4, "Planning an IPv6 Network (Tasks)."
- To configure IPv6 and create a dual-stack network environment, refer to Chapter 7, "Configuring an IPv6 Network (Tasks)."

## Major TCP/IP Administrative Tasks (Task Map)

| Task | Description | For Information |
|---|---|---|
| Display configuration information about an interface. | Determine the current configuration of each interface on a system. | "How to Get Information About a Specific Interface" on page 205 |

| Task | Description | For Information |
|---|---|---|
| Display interface address assignments. | Determine the address assignments for all interfaces on the local system. | "How to Display Interface Address Assignments" on page 206 |
| Display statistics on a per-protocol basis. | Monitor the performance of the network protocols on a particular system. | "How to Display Statistics by Protocol" on page 209 |
| Display network status. | Monitor your system by displaying all sockets and routing table entries. The output includes the inet address family for IPv4 and inet6 address family for IPv6. | "How to Display the Status of Sockets" on page 212 |
| Display the status of network interfaces. | Monitor the performance of network interfaces, which is useful for troubleshooting transmission problems. | "How to Display Network Interface Status" on page 212 |
| Display packet transmission status. | Monitor the state of packets as they are sent over the wire. | "How to Display the Status of Transmissions for Packets of a Specific Address Type" on page 214 |
| Control the display output of IPv6-related commands. | Controls the output of the `ping`, `netstat`, `ifconfig`, and `traceroute` commands. Creates a file that is named `inet_type`. Sets the `DEFAULT_IP` variable in this file. | "How to Control the Display Output of IP-Related Commands" on page 217 |
| Monitor network traffic. | Displays all IP packets by using the `snoop` command. | "How to Monitor IPv6 Network Traffic" on page 225 |
| Trace all routes that are known to the network's routers. | Uses the `traceroute` command to show all routes. | "How to Trace All Routes" on page 221 |

# Monitoring the Interface Configuration With the `ifconfig` Command

You use the `ifconfig` command to manually assign IP addresses to interfaces and to manually configure interface parameters. In addition, the Solaris startup scripts run `ifconfig` to configure pseudo interfaces, such as 6to4 tunnel endpoints.

This book contains many tasks that use the various options of the versatile `ifconfig` command. For a complete description of this command, its options, and its variables, refer to the `ifconfig`(1M) man page. The basic syntax of `ifconfig` follows:

`ifconfig` *interface [protocol-family]*

## ▼ How to Get Information About a Specific Interface

Use the `ifconfig` command to determine basic information about the interfaces of a particular system. For example, a simple `ifconfig` query can tell you the following:

- Device names of all interfaces on a system
- All IPv4 and, if applicable, all IPv6 addresses that are assigned to the interfaces
- Whether these interfaces are currently configured

The following procedure shows how to use the `ifconfig` command to obtain basic configuration information about a system's interfaces.

**1 On the local host, assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Obtain information about a particular interface.**

```
# ifconfig interface
```

The output from the ifconfig command has the following format:

- Status line

  The first line in the `ifconfig` command output includes the interface name and status flags currently associated with the interface. Also, the status line includes the maximum transmission unit (MTU) that is configured for the particular interface and an index number. Use the status line to determine the current state of the interface.

- IP address information line

  The second line of the `ifconfig` output includes the IPv4 address or IPv6 address that is configured for the interface. For an IPv4 address, the configured netmask and broadcast address are also displayed.

- MAC address line

  When you run the `ifconfig` command as superuser or with a similar role, the `ifconfig` output contains a third line. For an IPv4 address, the third line shows the MAC address (Ethernet layer address) that is assigned to the interface. For an IPv6 address, the third line in the output shows the link-local address that the IPv6 `in.ndpd` daemon generates from the MAC address.

### Example 8–1 Basic Interface Information From the `ifconfig` Command

The following example shows how to obtain information about the `eri` interface on a particular host by using the `ifconfig` command.

```
# ifconfig eri
eri0: flags=863<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 1
      inet 10.0.0.112 netmask ffffff80 broadcast 10.8.48.127
      ether 8:0:20:b9:4c:54
```

The next table describes the variable information in an `ifconfig` query. The preceding output is used as an example.

| Variable | Screen Output | Description |
|---|---|---|
| Interface name | `eri0` | Indicates the device name of the interface whose status was requested in the `ifconfig` command. |
| Interface status | `flags=863<UP` | Displays the status of the interface, including any flags that are currently associated with the interface. Here you can determine whether the interface is currently initialized (`UP`) or not initialized (`DOWN`). |
| Broadcast status | `BROADCAST` | Indicates that the interface supports IPv4 broadcasts. |
| Transmission status | `RUNNING` | Indicates that the system is transmitting packets through the interface. |
| Multicast status | `MULTICAST, IPv4` | Shows that the interface supports multicast transmissions. The example interface supports IPv4 multicast transmissions. |
| Maximum transmission unit | `mtu 1500` | Shows that this interface has a maximum transfer size of 1500 octets. |
| IP address | `inet 10.0.0.112` | Displays the IPv4 or IPv6 address that is assigned to the interface. Example interface `eri0` has the IPv4 address `10.0.0.112`. |
| Netmask | `netmask ffffff80` | Displays the IPv4 netmask of the particular interface. Note that IPv6 addresses do not use netmasks. |
| MAC address | `ether 8:0:20:b9:4c:54` | Shows the interface's Ethernet layer address. |

## ▼ How to Display Interface Address Assignments

Routers and multihomed hosts have more than one interface and, often, more than one IP address assigned to each interface. You can use the `ifconfig` command to display all addresses that are assigned to the interfaces of a system. You can also use the `ifconfig` command to display only IPv4 or IPv6 address assignments. To additionally display the MAC addresses of the interfaces, you must first log in as superuser or assume the appropriate role.

For more information on the `ifconfig` command, see the `ifconfig`(1M) man page.

1. **On the local system, assume the Network Management role or become superuser.**

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Obtain information about all interfaces.**

   You can use variations of the `ifconfig -a` command to do the following:

   - View all addresses of all interfaces on the system.

     ```
     # ifconfig -a
     ```

   - View all IPv4 addresses that are assigned to a system's interfaces.

     ```
     # ifconfig -a4
     ```

   - If the local system is IPv6-enabled, display all IPv6 addresses that are assigned to a system's interfaces.

     ```
     ifconfig -a6
     ```

**Example 8–2** Displaying Addressing Information for All Interfaces

This example shows entries for a host with solely a primary network interface, `qfe0`. Nevertheless, the `ifconfig` output shows that three forms of addresses are currently assigned to `qfe0`: loopback (`lo0`), IPv4 (`inet`), and IPv6 (`inet6`). In the IPv6 section of the output, note that the line for interface `qfe0` displays the link-local IPv6 address. The second address for `qfe0` is displayed on the `qfe0:1` line.

```
% ifconfig -a
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1004843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.0.0.112 netmask ffffff80 broadcast 10.0.0.127
        ether 8:0:20:b9:4c:54
lo0: flags=2000849 <UP,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841 <UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b9:4c:54
        inet6 fe80::a00:20ff:feb9:4c54/10
qfe0:1: flags=2080841 <UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 2001:db8:3c4d:48:a00:20ff:feb9:4c54/64
```

**Example 8–3** Displaying Addressing Information for All IPv4 Interfaces

This example shows the IPv4 address that is configured for a multihomed host. You do not need to be logged in as superuser to run this form of the `ifconfig` command.

```
% ifconfig -a4
lo0: flags=1000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
qfe0: flags=1004843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.0.0.112 netmask ffffff80 broadcast 10.0.0.127
        ether 8:0:20:b9:4c:54
qfe1: flags=1004843 <UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 10.0.0.118 netmask ffffff80 broadcast 10.0.0.127
        ether 8:0:20:6f:5e:17
```

**Example 8–4** Displaying Addressing Information for All IPv6 Interfaces

This example shows only the IPv6 addresses that are configured for a particular host. You do not need to be logged in as superuser to run this form of the `ifconfig` command.

```
% ifconfig -a6
lo0: flags=2000849 <UP,LOOPBACK,RUNNING,MULTICAST,IPv6> mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841 <UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        ether 8:0:20:b9:4c:54
        inet6 fe80::a00:20ff:feb9:4c54/10
qfe0:1: flags=2080841 <UP,RUNNING,MULTICAST,ADDRCONF,IPv6> mtu 1500 index 2
        inet6 2001:db8:3c4d:48:a00:20ff:feb9:4c54/64
```

This output from `ifconfig` shows the following three types of IPv6 address forms that are assigned to the single interface of a host:

`lo0`
: IPv6 loopback address.

`inet6 fe80::a00:20ff:feb9:4c54/10`
: Link-local address that is assigned to the primary network interface.

`inet6 2001:db8:3c4d:48:a00:20ff:feb9:4c54/64`
: IPv6 address, including subnet prefix. The term `ADDRCONF` in the output indicates that this address was autoconfigured by the host.

# Monitoring Network Status With the `netstat` Command

The `netstat` command generates displays that show network status and protocol statistics. You can display the status of TCP, SCTP, and UDP endpoints in table format. You can also display routing table information and interface information.

The `netstat` command displays various types of network data, depending on the selected command-line option. These displays are the most useful for system administration. The basic syntax for `netstat` follows:

`netstat` [`-m`] [`-n`] [`-s`] [`-i` | `-r`] [`-f`*address-family*]

This section describes the most commonly used options of the `netstat` command. For a detailed description of all `netstat` options, refer to the `netstat`(1M) man page.

## ▼ How to Display Statistics by Protocol

The `netstat -s` option displays protocol statistics for the UDP, TCP, SCTP, ICMP, and IP protocols.

---

**Note –** You can us your Solaris user account to obtain output from the `netstat` command.

---

- **Display the protocol status.**

```
$ netstat -s
```

**Example 8–5** Network Protocol Statistics

The following example shows the output of the `netstat -s` command. Parts of the output have been truncated. The output can indicate areas where a protocol is having problems. For example, statistical information from ICMPv4 and ICMPv6 can indicate where the ICMP protocol has found errors.

```
RAWIP
        rawipInDatagrams    =  4701     rawipInErrors       =     0
        rawipInCksumErrs    =     0     rawipOutDatagrams   =     4
        rawipOutErrors      =     0

UDP
        udpInDatagrams      = 10091     udpInErrors         =     0
        udpOutDatagrams     = 15772     udpOutErrors        =     0

TCP     tcpRtoAlgorithm     =     4     tcpRtoMin           =   400
        tcpRtoMax           = 60000     tcpMaxConn          =    -1
        .
        .
        tcpListenDrop       =     0     tcpListenDropQ0     =     0
        tcpHalfOpenDrop     =     0     tcpOutSackRetrans   =     0

IPv4    ipForwarding        =     2     ipDefaultTTL        =   255
        ipInReceives        =300182     ipInHdrErrors       =     0
        ipInAddrErrors      =     0     ipInCksumErrs       =     0
        .
        .
        ipsecInFailed       =     0     ipInIPv6            =     0
```

```
        ipOutIPv6           =     3     ipOutSwitchIPv6     =     0

IPv6    ipv6Forwarding      =     2     ipv6DefaultHopLimit =   255
        ipv6InReceives      = 13986     ipv6InHdrErrors     =     0
        ipv6InTooBigErrors  =     0     ipv6InNoRoutes      =     0
        .
        .
        rawipInOverflows    =     0     ipv6InIPv4          =     0

       ipv6OutIPv4         =     0     ipv6OutSwitchIPv4   =     0

ICMPv4  icmpInMsgs          = 43593     icmpInErrors        =     0
        icmpInCksumErrs     =     0     icmpInUnknowns      =     0
        .
        .
        icmpInOverflows     =     0

ICMPv6  icmp6InMsgs         = 13612     icmp6InErrors       =     0
        icmp6InDestUnreachs =     0     icmp6InAdminProhibs =     0
        .
        .
        icmp6OutGroupQueries=     0     icmp6OutGroupResps  =     2
        icmp6OutGroupReds   =     0

IGMP:
      12287 messages received
          0 messages received with too few bytes
          0 messages received with bad checksum
      12287 membership queries received
SCTP  sctpRtoAlgorithm     =  vanj
      sctpRtoMin           =  1000
      sctpRtoMax           = 60000
      sctpRtoInitial       =  3000
      sctpTimHearBeatProbe =     2
      sctpTimHearBeatDrop  =     0
      sctpListenDrop       =     0
      sctpInClosed         =     0
```

## ▼ How to Display the Status of Transport Protocols

You can display the status of the transport protocols through the `netstat` command. For detailed information, refer to the `netstat`(1M) man page.

**1 Display the status of the TCP and SCTP transport protocols on a system.**

```
$ netstat
```

**2 Display the status of a particular transport protocol on a system.**

```
$ netstat -P transport-protocol
```

Values for the *transport-protocol* variable are `tcp`, `sctp`, or `udp`.

### Example 8–6 Displaying the Status of the TCP and SCTP Transport Protocols

This example shows the output of the basic `netstat` command. Note that IPv4-only information is displayed.

```
$ netstat

TCP: IPv4
   Local Address     Remote Address    Swind Send-Q  Rwind Recv-Q      State
----------------- -------------------- ----- ------  ----- ------     -------
lhost-1.login      abc.def.local.Sun.COM.980 49640      0     49640    0 ESTABLISHED
lhost-1.login      ghi.jkl.local.Sun.COM.1020 49640     1     49640    0 ESTABLISHED
remhost-1.1014     mno.pqr.remote.Sun.COM.nfsd 49640    0     49640    0 TIME_WAIT
SCTP:
Local Address    Remote Address  Swind  Send-Q  Rwind  Recv-Q StrsI/O  State
---------------- --------------  -----  ------ ------ ------  ------   -------
 *.echo            0.0.0.0            0       0 102400      0   128/1   LISTEN
 *.discard         0.0.0.0            0       0 102400      0   128/1   LISTEN
 *.9001            0.0.0.0            0       0 102400      0   128/1   LISTEN
```

### Example 8–7 Displaying the Status of a Particular Transport Protocol

This example shows the results when you specify the `-P` option of `netstat`.

```
$ netstat -P tcp

TCP: IPv4
   Local Address     Remote Address    Swind Send-Q  Rwind Recv-Q      State
----------------- -------------------- ----- ------  ----- ------     -------
lhost-1.login      abc.def.local.Sun.COM.980 49640      0     49640    0 ESTABLISHED
lhost.login        ghi.jkl.local.Sun.COM.1020 49640     1     49640    0 ESTABLISHED
remhost.1014       mno.pqr.remote.Sun.COM.nfsd 49640    0     49640    0 TIME_WAIT

TCP: IPv6
 Local Address    Remote Address        Swind Send-Q Rwind Recv-Q   State If
---------------- ---------------------- ------ ----- ------ ------------ -----
localhost.38983   localhost.32777       49152      0 49152      0 ESTABLISHED
localhost.32777   localhost.38983       49152      0 49152      0 ESTABLISHED
localhost.38986   localhost.38980       49152      0 49152      0 ESTABLISHED
```

## How to Display Network Interface Status

The i option of the netstat command shows the state of the network interfaces that are configured on the local system. With this option, you can determine the number of packets a system transmits and receives on each network.

- **Display the status of interfaces on the network.**

```
$ netstat -i
```

**Example 8–8** Network Interface Status Display

The next example shows the status of IPv4 and IPv6 packet flow through the host's interfaces.

For example, the input packet count (Ipkts) that is displayed for a server can increase each time a client tries to boot, while the output packet count (Opkts) remains steady. This outcome suggests that the server is seeing the boot request packets from the client. However, the server does not know to respond to them. This confusion might be caused by an incorrect address in the hosts, ipnodes, or ethers database.

However, if the input packet count is steady over time, then the machine does not see the packets at all. This outcome suggests a different type of failure, possibly a hardware problem.

```
Name  Mtu  Net/Dest      Address        Ipkts  Ierrs Opkts  Oerrs Collis Queue
lo0   8232 loopback      localhost      142    0     142    0     0      0
hme0  1500 host58        host58        1106302 0     52419  0     0      0

Name  Mtu  Net/Dest      Address                   Ipkts  Ierrs Opkts  Oerrs Collis
lo0   8252 localhost     localhost                  142    0     142    0     0
hme0  1500 fe80::a00:20ff:feb9:4c54/10 fe80::a00:20ff:feb9:4c54 1106305 0 52422 0  0
```

## How to Display the Status of Sockets

The -a option of the netstat command enables you to view the status of sockets on the local host.

- **Type the following to display the status of sockets and routing table entries:**

  You can use your user account to run this option of netstat.

```
% netstat -a
```

**Example 8–9** Displaying All Sockets and Routing Table Entries

The output of the netstat -a command shows extensive statistics. The following example shows portions of typical netstat -a output.

```
UDP: IPv4
   Local Address         Remote Address     State
-------------------- -------------------- -------
      *.bootpc                            Idle
host85.bootpc                             Idle
      *.*                                 Unbound
      *.*                                 Unbound
      *.sunrpc                            Idle
      *.*                                 Unbound
      *.32771                             Idle
      *.sunrpc                            Idle
      *.*                                 Unbound
      *.32775                             Idle
      *.time                              Idle
       .
       .
      *.daytime                           Idle
      *.echo                              Idle
      *.discard                           Idle

UDP: IPv6
   Local Address                      Remote Address                    State      If
-------------------------------- -------------------------------- ---------- -----
      *.*                                                        Unbound
      *.*                                                        Unbound
      *.sunrpc                                                   Idle
      *.*                                                        Unbound
      *.32771                                                    Idle
      *.32778                                                    Idle
      *.syslog                                                   Idle
      .
      .
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
      *.*                  *.*                0      0 49152      0 IDLE
localhost.4999             *.*                0      0 49152      0 LISTEN
      *.sunrpc             *.*                0      0 49152      0 LISTEN
      *.*                  *.*                0      0 49152      0 IDLE
      *.sunrpc             *.*                0      0 49152      0 LISTEN
      .
      .
      *.printer            *.*                0      0 49152      0 LISTEN
      *.time               *.*                0      0 49152      0 LISTEN
      *.daytime            *.*                0      0 49152      0 LISTEN
      *.echo               *.*                0      0 49152      0 LISTEN
      *.discard            *.*                0      0 49152      0 LISTEN
      *.chargen            *.*                0      0 49152      0 LISTEN
```

```
      *.shell              *.*                0      0 49152      0 LISTEN
      *.shell              *.*                0      0 49152      0 LISTEN
      *.kshell             *.*                0      0 49152      0 LISTEN
      *.login
       .
       .
            *.*               0      0 49152      0 LISTEN
  *TCP: IPv6
 Local Address            Remote Address        Swind Send-Q Rwind Recv-Q   State If
---------------------- ---------------------- ----- ------ ----- ------    ----
   *.*                      *.*                  0      0 49152      0      IDLE
   *.sunrpc                 *.*                  0      0 49152      0      LISTEN
   *.*                      *.*                  0      0 49152      0      IDLE
   *.32774                  *.*                  0      0 49152
```

## ▼ How to Display the Status of Transmissions for Packets of a Specific Address Type

Use the `-f` option of the `netstat` command to view statistics related to packet transmissions of a particular address family.

- **View statistics for transmissions of either IPv4 or IPv6 packets.**

  `$ netstat -f` *inet* | *inet6*

  To view IPv4 transmission information, type `inet` as the argument to `netstat -f`. Use `inet6` as the argument to `netstat -f` to view IPv6 information.

**Example 8–10** Status of IPv4 Packet Transmission

The following example shows output from the `netstat -f inet` command.

```
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
host58.734         host19.nfsd       49640      0 49640      0 ESTABLISHED
host58.38063       host19.32782      49640      0 49640      0 CLOSE_WAIT
host58.38146       host41.43601      49640      0 49640      0 ESTABLISHED
host58.996         remote-host.login 49640      0 49206      0 ESTABLISHED
```

**Example 8–11** Status of IPv6 Packet Transmission

The following example shows output from the `netstat -f inet6` command.

```
TCP: IPv6
 Local Address          Remote Address        Swind Send-Q Rwind Recv-Q   State    If
------------------ ------------------------ ----- ------ ----- ------ --------- -----
```

```
localhost.38065        localhost.32792       49152   0 49152      0    ESTABLISHED
localhost.32792        localhost.38065       49152   0 49152      0    ESTABLISHED
localhost.38089        localhost.38057       49152   0 49152      0    ESTABLISHED
```

## How to Display the Status of Known Routes

The `-r` option of the `netstat` command displays the routing table for the local host. This table shows the status of all routes that the host knows about. You can run this option of `netstat` from your user account.

- **Display the IP routing table.**

  ```
  $ netstat -r
  ```

**Example 8–12** Routing Table Output by the `netstat` Command

The following example shows output from the `netstat -r` command.

```
Routing Table: IPv4
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
host15               myhost               U         1  31059  hme0
10.0.0.14            myhost               U         1      0  hme0
default              distantrouter        UG        1      2  hme0
localhost            localhost            UH        42019361  lo0

Routing Table: IPv6
  Destination/Mask            Gateway                   Flags Ref   Use   If
--------------------------- --------------------------- ----- --- ------ -----
2002:0a00:3010:2::/64    2002:0a00:3010:2:1b2b:3c4c:5e6e:abcd U  1      0 hme0:1
fe80::/10                fe80::1a2b:3c4d:5e6f:12a2    U       1     23 hme0
ff00::/8                 fe80::1a2b:3c4d:5e6f:12a2    U       1      0 hme0
default                  fe80::1a2b:3c4d:5e6f:12a2    UG      1      0 hme0
localhost                localhost                    UH      9  21832 lo0
```

| Parameter | Description |
|---|---|
| `Destination`<br>`Destination/Mask` | Specifies the host that is the destination endpoint of the route. Note that the IPv6 routing table shows the prefix for a 6to4 tunnel endpoint (`2002:0a00:3010:2::/64`) as the route destination endpoint. |
| `Gateway` | Specifies the gateway to use for forwarding packets. |
| `Flags` | Indicates the current status of the route. The `U` flag indicates that the route is up. The `G` flag indicates that the route is to a gateway. |

| Parameter | Description |
|---|---|
| Use | Shows the number of packets sent. |
| Interface | Indicates the particular interface on the local host that is the source endpoint of the transmission. |

# Probing Remote Hosts With the ping Command

You can use the ping command to determine the status of a remote host. When you run ping, the ICMP protocol sends a datagram to the host that you specify, asking for a response. ICMP is the protocol responsible for error handling on a TCP/IP network. When you use ping, you can find out whether an IP connection exists for the specified remote host.

The following is the basic syntax of ping:

/usr/sbin/ping *host [timeout]*

In this syntax, *host* is the name of the remote host. The optional *timeout* argument indicates the time in seconds for the ping command to continue trying to reach the remote host. The default is 20 seconds. For additional syntax and options, refer to the ping(1M) man page.

## ▼ How to Determine if a Remote Host Is Running

- **Type the following form of the ping command:**

  $ **ping** *hostname*

  If host *hostname* is accepting ICMP transmissions, this message is displayed:

  *hostname* is alive

  This message indicates that *hostname* responded to the ICMP request. However, if *hostname* is down or cannot receive the ICMP packets, you receive the following response from the ping command:

  no answer from *hostname*

## ▼ How to Determine if a Host Is Dropping Packets

Use the -s option of the ping command to determine if a remote host is running but nevertheless losing packets.

- **Type the following form of the ping command:**

  $ **ping -s** *hostname*

**Example 8–13** `ping` Output for Detecting Packet Dropping

The `ping -s` *hostname* command continually sends packets to the specified host until you send an interrupt character or a time out occurs. The responses on your screen resemble the following:

```
& ping -s host1.domain8
PING host1.domain8 : 56 data bytes
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=0. time=1.67 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=1. time=1.02 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=2. time=0.986 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=3. time=0.921 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=4. time=1.16 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=5. time=1.00 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=5. time=1.980 ms

^C

----host1.domain8  PING Statistics----
7 packets transmitted, 7 packets received, 0% packet loss
round-trip (ms)  min/avg/max/stddev = 0.921/1.11/1.67/0.26
```

The packet-loss statistic indicates whether the host has dropped packets. If `ping` fails, check the status of the network that is reported by the `ifconfig` and `netstat` commands. Refer to "Monitoring the Interface Configuration With the `ifconfig` Command" on page 204 and "Monitoring Network Status With the `netstat` Command" on page 208.

# Administering and Logging Network Status Displays

The following tasks show how to check the status of the network by using well-known networking commands.

## ▼ How to Control the Display Output of IP-Related Commands

You can control the output of the `netstat` and `ifconfig` commands to display IPv4 information only, or both IPv4 and IPv6 information.

1. **Create the `/etc/default/inet_type` file.**
2. **Add one of the following entries to `/etc/default/inet_type`, as required for your network:**
   - To display IPv4 information only:

     ```
     DEFAULT_IP=IP_VERSION4
     ```

- To display both IPv4 and IPv6 information:

  ```
  DEFAULT_IP=BOTH
  ```

  Or

  ```
  DEFAULT_IP=IP_VERSION6
  ```

  For more information about the `inet_type` file, see the `inet_type`(4) man page.

---

**Note –** The `-4` and `-6` flags in the `ifconfig` command override the values set in the `inet_type` file. The `-f` flag in the `netstat` command also overrides the values set in the `inet_type` file.

---

**Example 8–14** Controlling Output to Select IPv4 and IPv6 Information

- When you specify the `DEFAULT_IP=BOTH` or `DEFAULT_IP=IP_VERSION6` variable in the `inet_type` file, you should have the following output:

```
% ifconfig -a
lo0: flags=1000849 mtu 8232 index 1
        inet 10.10.0.1 netmask ff000000
qfe0: flags=1000843 mtu 1500 index 2
        inet 10.46.86.54 netmask ffffff00 broadcast 10.46.86.255
        ether 8:0:20:56:a8
lo0: flags=2000849 mtu 8252 index 1
        inet6 ::1/128
qfe0: flags=2000841 mtu 1500 index 2
        ether 8:0:20:56:a8
        inet6 fe80::a00:fe73:56a8/10
qfe0:1: flags=2080841 mtu 1500 index 2
        inet6 2001:db8:3c4d:5:a00:fe73:56a8/64
```

- When you specify the `DEFAULT_IP=IP_VERSION4` or `DEFAULT_IP=IP_VERSION6` variable in the `inet_type` file, you should have the following output:

```
% ifconfig -a
lo0: flags=849 mtu 8232
        inet 10.10.0.1 netmask ff000000
qfe0: flags=843 mtu 1500
        inet 10.46.86.54 netmask ffffff00 broadcast 10.46.86.255
        ether 8:0:20:56:a8
```

## How to Log Actions of the IPv4 Routing Daemon

If you suspect a malfunction of `routed`, the IPv4 routing daemon, you can start a log that traces the daemon's activity. The log includes all packet transfers when you start the `routed` daemon.

1 **On the local host, assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Create a log file of routing daemon actions:**

```
# /usr/sbin/in.routed /var/log-file-name
```

**Caution –** On a busy network, this command can generate almost continuous output.

**Example 8–15** Network Log for the `in.routed` Daemon

The following example shows the beginning of the log that is created by the procedure "How to Log Actions of the IPv4 Routing Daemon" on page 219.

```
-- 2003/11/18 16:47:00.000000 --
Tracing actions started
RCVBUF=61440
Add interface lo0  #1   127.0.0.1      -->127.0.0.1/32
   <UP|LOOPBACK|RUNNING|MULTICAST|IPv4> <PASSIVE>
Add interface hme0 #2   10.10.48.112    -->10.10.48.0/25
    <UP|BROADCAST|RUNNING|MULTICAST|IPv4>
turn on RIP
Add    10.0.0.0        -->10.10.48.112      metric=0  hme0  <NET_SYN>
Add    10.10.48.85/25  -->10.10.48.112      metric=0  hme0  <IF|NOPROP>
```

## How to Trace the Activities of the IPv6 Neighbor Discovery Daemon

If you suspect a malfunction of the IPv6 `in.ndpd` daemon, you can start a log that traces the daemon's activity. This trace is displayed on the standard output until terminated. This trace includes all packet transfers when you start the `in.ndpd` daemon.

1 **Assume the Primary Administrator role, or become superuser, on the local IPv6 node.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Start a trace of the** `in.ndpd` **daemon.**

```
# /usr/lib/inet/in.ndpd -t
```

3 **Terminate the trace as needed by typing Control-C.**

**Example 8–16** Trace of the `in.ndpd` Daemon

The following output shows the beginning of a trace of `in.ndpd`.

```
# /usr/lib/inet/in.ndpd -t
Nov 18 17:27:28 Sending solicitation to  ff02::2 (16 bytes) on hme0
Nov 18 17:27:28         Source LLA: len 6 <08:00:20:b9:4c:54>
Nov 18 17:27:28 Received valid advert from fe80::a00:20ff:fee9:2d27 (88 bytes) on hme0
Nov 18 17:27:28         Max hop limit: 0
Nov 18 17:27:28         Managed address configuration: Not set
Nov 18 17:27:28         Other configuration flag: Not set
Nov 18 17:27:28         Router lifetime: 1800
Nov 18 17:27:28         Reachable timer: 0
Nov 18 17:27:28         Reachable retrans timer: 0
Nov 18 17:27:28         Source LLA: len 6 <08:00:20:e9:2d:27>
Nov 18 17:27:28         Prefix: 2001:08db:3c4d:1::/64
Nov 18 17:27:28                 On link flag:Set
Nov 18 17:27:28                 Auto addrconf flag:Set
Nov 18 17:27:28                 Valid time: 2592000
Nov 18 17:27:28                 Preferred time: 604800
Nov 18 17:27:28         Prefix: 2002:0a00:3010:2::/64
Nov 18 17:27:28                 On link flag:Set
Nov 18 17:27:28                 Auto addrconf flag:Set
Nov 18 17:27:28                 Valid time: 2592000
Nov 18 17:27:28                 Preferred time: 604800
```

# Displaying Routing Information With the `traceroute` Command

The `traceroute` command traces the route an IP packet follows to a remote system. For technical details about `traceroute`, see the `traceroute`(1M) man page.

You use the `traceroute` command to uncover any routing misconfiguration and routing path failures. If a particular host is unreachable, you can use `traceroute` to see what path the packet follows to the remote host and where possible failures might occur.

The `traceroute` command also displays the round trip time for each gateway along the path to the target host. This information can be useful for analyzing where traffic is slow between the two hosts.

## How to Find Out the Route to a Remote Host

- **Type the following to discover the route to a remote system:**

  `% traceroute` *destination-hostname*

  You can run this form of the `traceroute` command from your user account.

**Example 8–17** Using the `traceroute` Command to Show the Route to a Remote Host

The following output from the `traceroute` command shows the seven–hop path a packet follows from the local system `nearhost` to the remote system `farhost`. The output also shows the times for a packet to traverse each hop.

```
istanbul% traceroute farhost.faraway.com
    traceroute to farhost.faraway.com (172.16.64.39), 30 hops max, 40 byte packets
     1  frbldg7c-86 (172.16.86.1)  1.516 ms  1.283 ms  1.362 ms
     2  bldg1a-001 (172.16.1.211)  2.277 ms  1.773 ms  2.186 ms
     3  bldg4-bldg1 (172.16.4.42)  1.978 ms  1.986 ms  13.996 ms
     4  bldg6-bldg4 (172.16.4.49)  2.655 ms  3.042 ms  2.344 ms
     5  ferbldg11a-001 (172.16.1.236)  2.636 ms  3.432 ms  3.830 ms
     6  frbldg12b-153 (172.16.153.72)  3.452 ms  3.146 ms  2.962 ms
     7  sanfrancisco (172.16.64.39)  3.430 ms  3.312 ms  3.451 ms
```

## How to Trace All Routes

This procedure uses the `-a` option of the `traceroute` command to trace all routes.

- **Type the following command on the local system:**

  `% traceroute -a`*host-name*

  You can run this form of the `traceroute` command from your user account.

**Example 8–18** Tracing All Routes to a Dual-Stack Host

This example shows all possible routes to a dual-stack host.

```
% traceroute -a v6host.remote.com
traceroute: Warning: Multiple interfaces found; using 2::56:a0:a8 @ eri0:2
traceroute to v6host (2001:db8:4a3b::102:a00:fe79:19b0),30 hops max, 60 byte packets
 1  v6-rout86 (2001:db8:4a3b:56:a00:fe1f:59a1)  35.534 ms  56.998 ms *
 2  2001:db8::255:0:c0a8:717  32.659 ms  39.444 ms *
 3  farhost.faraway.COM (2001:db8:4a3b::103:a00:fe9a:ce7b)  401.518 ms  7.143 ms *
 4  distant.remote.com (2001:db8:4a3b::100:a00:fe7c:cf35)  113.034 ms  7.949 ms *
 5  v6host (2001:db8:4a3b::102:a00:fe79:19b0)  66.111 ms *  36.965 ms
```

```
traceroute to v6host.remote.com  (192.168.10.75),30 hops max,40 byte packets
 1  v6-rout86 (172.16.86.1)  4.360 ms  3.452 ms  3.479 ms
 2  flrmpj17u.here.COM (172.16.17.131)  4.062 ms  3.848 ms  3.505 ms
 3  farhost.farway.com (10.0.0.23)  4.773 ms *  4.294 ms
 4  distant.remote.com (192.168.10.104)  5.128 ms  5.362 ms *
 5  v6host  (192.168.15.85)  7.298 ms  5.444 ms *
```

# Monitoring Packet Transfers With the snoop Command

You can use the snoop command to monitor the state of data transfers. snoop captures network packets and displays their contents in the format that you specify. Packets can be displayed as soon as they are received, or saved to a file. When snoop writes to an intermediate file, packet loss under busy trace conditions is unlikely. snoop itself is then used to interpret the file.

To capture packets to and from the default interface in promiscuous mode, you must assume the Network Management role or become superuser. In summary form, snoop displays only the data that pertains to the highest-level protocol. For example, an NFS packet only displays NFS information. The underlying RPC, UDP, IP, and Ethernet frame information is suppressed but can be displayed if either of the verbose options is chosen.

Use snoop frequently and consistently to become familiar with normal system behavior. For assistance in analyzing packets, look for a recent white paper and RFC, and seek the advice of an expert in a particular area, such as NFS or NIS. For details on using snoop and its options, refer to the snoop(1M) man page.

## ▼ How to Check Packets From All Interfaces

1. **On the local host, assume the Network Management role or become superuser.**

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Print information about the interfaces that are attached to the system.**

   ```
   # ifconfig -a
   ```

   The snoop command normally uses the first non-loopback device, typically the primary network interface.

3. **Begin packet capture by typing snoop without arguments, as shown in Example 8–19.**

4. **Use Control-C to halt the process.**

**Example 8–19** Output From the snoop Command

The basic snoop command returns output that resembles the following, for a dual-stack host.

```
% snoop
Using device /dev/hme (promiscuous mode)
farhost.remote.com -> myhost       RLOGIN C port=993
    myhost -> farhost.remote.com   RLOGIN R port=993 Using device /dev/hme
router5.local.com -> router5.local.com ARP R 10.0.0.13, router5.local.com is
    0:10:7b:31:37:80
router5.local.com -> BROADCAST     TFTP Read "network-confg" (octet)
farhost.remote.com -> myhost       RLOGIN C port=993
    myhost ->   nisserve2          NIS C MATCH 10.0.0.64 in ipnodes.byaddr
nisserve2 ->    myhost             NIS R MATCH No such key
    blue-112 -> slave-253-2        NIS C MATCH 10.0.0.112 in ipnodes.byaddr
myhost -> DNSserver.local.com      DNS C 192.168.10.10.in-addr.arpa. Internet PTR ?
DNSserver.local.com  myhost        DNS R 192.168.10.10.in-addr.arpa. Internet PTR
   niserve2.
.
.
farhost.remote.com-> myhost        RLOGIN C port=993
    myhost -> farhost.remote.com   RLOGIN R port=993 fe80::a00:20ff:febb:
.
fe80::a00:20ff:febb:e09 -> ff02::9 RIPng R (5 destinations)
```

The packets that are captured in this output show a remote login section, including lookups to the NIS and DNS servers for address resolution. Also included are periodic ARP packets from the local router and advertisements of the IPv6 link-local address to `in.ripngd`.

## ▼ How to Capture snoop Output Into a File

1. **On the local host, assume the Network Management role or become superuser.**

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Capture a snoop session into a file.**

   `# snoop -o` *filename*

   For example:

```
# snoop /tmp/cap
Using device /dev/eri (promiscuous mode)
30 snoop: 30 packets captured
```

In the example, 30 packets have been captured in a file named /tmp/cap. The file can be in any directory with enough disk space. The number of packets that are captured is displayed on the command line, enabling you to press Control-C to abort at any time.

snoop creates a noticeable networking load on the host machine, which can distort the results. To see the actual results, run snoop from a third system.

3 **Inspect the** snoop **output captures file.**

```
# snoop -i filename
```

**Example 8–20** Contents of a snoop Output Captures File

The following output shows a variety of captures such as you might receive as output from the snoop -i command.

```
# snoop -i /tmp/cap
1   0.00000 fe80::a00:20ff:fee9:2d27 -> fe80::a00:20ff:fecd:4375
    ICMPv6 Neighbor advertisement
2   0.16198 farhost.com   -> myhost     RLOGIN C port=985
3   0.00008 myhost -> farhost.com       RLOGIN R port=985
10  0.91493    10.0.0.40 -> (broadcast)  ARP C Who is 10.0.0.40, 10.0.0.40 ?
34  0.43690 nearserver.here.com  -> 224.0.1.1  IP  D=224.0.1.1 S=10.0.0.40 LEN=28,
      ID=47453, TO =0x0, TTL=1
35  0.00034  10.0.0.40 -> 224.0.1.1    IP  D=224.0.1.1 S=10.0.0.40 LEN=28, ID=57376,
     TOS=0x0, TTL=47
```

## ▼ How to Check Packets Between an IPv4 Server and a Client

1 **Establish a** snoop **system off a hub that is connected to either the client or the server.**

The third system (the snoop system) checks all the intervening traffic, so the snoop trace reflects what is actually happening on the wire.

2 **On the** snoop **system, assume the Network Management role or become superuser.**

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

3 **Type** snoop **with options and save the output to a file.**

4 **Inspect and interpret the output.**

Refer to RFC 1761, Snoop Version 2 Packet Capture File Format (http://www.ietf.org/rfc/rfc1761.txt?number=1761) for details of the snoop capture file.

### ▼ How to Monitor IPv6 Network Traffic

You can use the `snoop` command to display only IPv6 packets.

1. **On the local node, assume the Network Management role or become superuser.**

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Capture IPv6 packets.**

   ```
   # snoop ip6
   ```

   For more information on the `snoop` command, see the `snoop`(1M) man page.

**Example 8–21** Displaying Only IPv6 Network Traffic

The following example shows typical output such as you might receive from running the `snoop ip6` command on a node.

```
# snoop ip6
fe80::a00:20ff:fecd:4374 -> ff02::1:ffe9:2d27 ICMPv6 Neighbor solicitation
fe80::a00:20ff:fee9:2d27 -> fe80::a00:20ff:fecd:4375 ICMPv6 Neighbor
      solicitation
fe80::a00:20ff:fee9:2d27 -> fe80::a00:20ff:fecd:4375 ICMPv6 Neighbor
      solicitation
fe80::a00:20ff:febb:e09 -> ff02::9      RIPng R (11 destinations)
fe80::a00:20ff:fee9:2d27 -> ff02::1:ffcd:4375 ICMPv6 Neighbor solicitation
```

# Administering Default Address Selection

The Solaris OS enables a single interface to have multiple IP addresses. For example, technologies, such as network multipathing (IPMP) enable multiple network interface cards (NICs) to connect to the same IP link layer. That link can have one or more IP addresses. Additionally, interfaces on IPv6-enabled systems have a link-local IPv6 address, at least one IPv6 routing address, and an IPv4 address for at least one interface.

When the system initiates a transaction, an application makes a call to the `getaddrinfo` socket. `getaddrinfo` discovers the possible address in use on the destination system. The kernel then prioritizes this list to find the best destination to use for the packet. This process is called *destination address ordering*. The Solaris kernel then selects the appropriate format for the source address, given the best destination address for the packet. The process is known as *address selection*. For more information on destination address ordering, see the `getaddrinfo`(3SOCKET) man page.

Both IPv4-only and dual-stack IPv4/IPv6 systems must perform default address selection. In most circumstances, you do not need to change the default address selection mechanisms. However, you might need to change the priority of address formats to support IPMP or to prefer 6to4 address formats, for example.

## How to Administer the IPv6 Address Selection Policy Table

The following procedure explains how to modify the address selection policy table. For conceptual information about IPv6 default address selection, refer to "`ipaddrsel` Command" on page 268.

**Caution –** Do not change the IPv6 address selection policy table, except for the reasons shown in the next task. You can cause problems on the network with a badly constructed policy table. Be sure to save a backup copy of the policy table, as is done in the next procedure.

**1 Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Review the current IPv6 address selection policy table.**

```
# ipaddrsel
# Prefix                 Precedence Label
::1/128                          50 Loopback
::/0                             40 Default
2002::/16                        30 6to4
::/96                            20 IPv4_Compatible
::ffff:0.0.0.0/96                10 IPv4
```

**3 Make a backup copy of the default address policy table.**

```
# cp /etc/inet/ipaddrsel.conf /etc/inet/ipaddrsel.conf.orig
```

**4 Use a text editor to add your customizations to `/etc/inet/ipaddrsel.conf`.**

Use the following syntax for entries in `/etc/inet/ipaddrsel`:

*prefix/prefix-length precedence label [# comment ]*

Here are some common modifications that you might want to make to your policy table:

- Give the highest priority to 6to4 addresses.

```
2002::/16                       50 6to4
::1/128                         45 Loopback
```

The 6to4 address format now has the highest priority, 50. Loopback, which previously had a 50 precedence, now has a 45 precedence. The other addressing formats remain the same.

- Designate a specific source address to be used in communications with a specific destination address.

```
::1/128                         50 Loopback
2001:1111:1111::1/128           40 ClientNet
2001:2222:2222::/48             40 ClientNet
::/0                            40 Default
```

This particular entry is useful for hosts with only one physical interface. Here `2001:1111:1111::1/128` is preferred as the source address on all packets that are bound for destinations within network `2001:2222:2222::/48`. The 40 priority gives higher precedence to the source address `2001:1111:1111::1/128` than to other address formats configured for the interface.

- Favor IPv4 addresses over IPv6 addresses.

```
::ffff:0.0.0.0/96               60 IPv4
::1/128                         50 Loopback
.
.
```

The IPv4 format `::ffff:0.0.0.0/96` has its precedence changed from the default 10 to 60, the highest priority in the table.

**5 Load the modified policy table into the kernel.**

```
ipaddrsel -f /etc/inet/ipaddrsel.conf
```

**6 If the modified policy table has problems, restore the default IPv6 address selection policy table.**

```
# ipaddrsel -d
```

## How to Modify the IPv6 Address Selection Table for the Current Session Only

When you edit the `/etc/inet/ipaddrsel.conf`, file, any modifications that you make persist across reboots. If you want the modified policy table to exist only in the current session, follow this procedure.

**1 Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Copy the contents of `/etc/inet/ipaddrsel` into *filename*, where *filename* represents a name of your choice.**

   ```
   # cp /etc/inet/ipaddrsel filename
   ```

3. **Edit the policy table in *filename* to your specifications.**

4. **Load the modified policy table into the kernel.**

   ```
   # ipaddrsel -f filename
   ```

   The kernel uses the new policy table until you reboot the system.

◆ ◆ ◆ CHAPTER 9

# Troubleshooting Network Problems (Tasks)

This chapter contains solutions for common problems that might occur on your network. The following topics are covered:


- "General Network Troubleshooting Tips" on page 229
- "Common Problems When Deploying IPv6" on page 231


## What's New in Troubleshooting Network Problems

In Solaris 10 8/07, the `/etc/inet/ipnodes` file becomes obsolete. Use `/etc/inet/ipnodes` only for earlier Solaris 10 releases, as explained in the individual procedures.

## General Network Troubleshooting Tips

One of the first signs of trouble on a network is a loss of communications by one or more hosts. If a host does not to come up at all the first time that the host is added to the network, the problem might be in one of the configuration files. The problem might also be a faulty network interface card. If a single host suddenly develops a problem, the network interface might be the cause. If the hosts on a network can communicate with each other but not with other networks, the problem could lie with the router. Or, the problem could be in another network.

You can use the `ifconfig` command to obtain information on network interfaces. Use the `netstat` command to display routing tables and protocol statistics. Third-party network diagnostic programs provide a number of troubleshooting tools. Refer to third-party documentation for information.

Less obvious are the causes of problems that degrade performance on the network. For example, you can use tools such as `ping` to quantify problems such as the loss of packets by a host.

# Running Basic Diagnostic Checks

If the network has problems, you can run a series of software checks to diagnose and fix basic, software-related problems.

## ▼ How to Perform Basic Network Software Checking

1 **On the local system, assume the Network Management role or become superuser.**

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Use the `netstat` command to display network information.**

For syntax and information about the `netstat` command, refer to "Monitoring Network Status With the `netstat` Command" on page 208 and the `netstat`(1M) man page.

3 **Check the `hosts` database (and, in Solaris 10 11/06 and previous releases, the `ipnodes` database, if you are using IPv6) to ensure that the entries are correct and current.**

For information about the `/etc/inet/hosts` database, refer to "`hosts` Database" on page 237 and the `hosts`(4) man page. For information about the `/etc/inet/ipnodes` database, refer to "`ipnodes` Database" on page 240 and the `ipnodes`(4) man page.

4 **If you are running the Reverse Address Resolution Protocol (RARP), check the Ethernet addresses in the `ethers` database to ensure that the entries are correct and current.**

5 **Try to connect to the local host by using the `telnet` command.**

For syntax and information about `telnet`, refer to the `telnet`(1) man page.

6 **Ensure that the network daemon `inetd` is running.**

```
# ps -ef | grep inetd
```

The following output verifies that the `inetd` daemon is running:

```
root 57 1 0 Apr 04 ? 3:19 /usr/sbin/inetd -s
```

7 **If IPv6 is enabled on your network, verify that the IPv6 daemon `in.ndpd` is running:**

```
# ps -ef | grep in.ndpd
```

The following output verifies that the `in.ndpd` daemon is running:

```
root 123  1 0  Oct 27 ?  0:03 /usr/lib/inet/in.ndpd
```

# Common Problems When Deploying IPv6

This section describes issues and problems that you might encounter while planning and deploying IPv6 at your site. For actual planning tasks, refer to Chapter 4, "Planning an IPv6 Network (Tasks)."

## IPv4 Router Cannot Be Upgraded to IPv6

If your existing equipment cannot be upgraded, you might have to purchase IPv6-ready equipment. Check the manufacturers' documentation for any equipment-specific procedures you might have to perform to support IPv6.

Certain IPv4 routers cannot be upgraded for IPv6 support. If this situation applies to your topology, physically wire an IPv6 router next to the IPv4 router. Then, you can tunnel from the IPv6 router over the IPv4 router. For tasks for configuring tunnels, refer to "Tasks for Configuring Tunnels for IPv6 Support (Task Map)" on page 188.

## Problems After Upgrading Services to IPv6

You might encounter the following situations when preparing services for IPv6 support:

- Certain applications, even after they are ported to IPv6, do not turn on IPv6 support by default. You might have to configure these applications to turn on IPv6.
- A server that runs multiple services, some of which are IPv4 only, and others that are both IPv4 and IPv6, can experience problems. Some clients might need to use both types of services, which leads to confusion on the server side.

## Current ISP Does Not Support IPv6

If you want to deploy IPv6 but your current ISP does not offer IPv6 addressing, consider the following alternatives to changing ISPs:

- Hire an ISP to provide a second line for IPv6 communications from your site. This solution is expensive.
- Get a *virtual ISP*. A virtual ISP provides your site with IPv6 connectivity but no link. Instead, you create a tunnel from your site, over your IPv4 ISP, to the virtual ISP.
- Use a 6to4 tunnel over your ISP to other IPv6 sites. For an address, use the registered IPv4 address of the 6to4 router as the public topology part of the IPv6 address.

## Security Issues When Tunneling to a 6to4 Relay Router

By nature, a tunnel between a 6to4 router and a 6to4 relay router is insecure. Security problems, such as the following, are inherent in such a tunnel:

- Though 6to4 relay routers do encapsulate and decapsulate packets, these routers do not check the data that is contained within the packets.
- Address spoofing is a major issue on tunnels to a 6to4 relay router. For incoming traffic, the 6to4 router is unable to match the IPv4 address of the relay router with the IPv6 address of the source. Therefore, the address of the IPv6 host can easily be spoofed. The address of the 6to4 relay router can also be spoofed.
- By default, no trust mechanism exists between 6to4 routers and 6to4 relay routers. Thus, a 6to4 router cannot identify whether the 6to4 relay router is to be trusted, or even if it is a legitimate 6to4 relay router. A trust relationship between the 6to4 site and the IPv6 destination must exist, or both sites leave themselves open to possible attacks.

These problems and other security issues that are inherent with 6to4 relay routers are explained in the Internet Draft, *Security Considerations for 6to4*. Generally, you should consider enabling support for 6to4 relay routers for the following reasons only:

- Your 6to4 site intends to communicate with a private, trusted IPv6 network. For example, you might enable 6to4 relay router support on a campus network that consists of isolated 6to4 sites and native IPv6 sites.
- Your 6to4 site has a compelling business reason to communicate with certain native IPv6 hosts.
- You have implemented the checks and trust models that are suggested in the Internet Draft, *Security Considerations for 6to4*.

## Known Issues With a 6to4 Router

The following known bugs affect 6to4 configuration:

- 4709338 – Need a RIPng implementation which recognizes static routes
- 4152864 – Configuring two tunnels with the same tsrc/tdst pair works

### Implementing Static Routes at the 6to4 Site (Bug ID 4709338)

The following issue occurs on 6to4 sites with routers that are internal to the 6to4 boundary router. When you configure the 6to4 pseudo-interface, the static route `2002::/16` is automatically added to the routing table on the 6to4 router. Bug 4709338 describes a limitation in the Solaris RIPng routing protocol that prevents this static route from being advertised to the 6to4 site.

Either of the following workarounds are available for Bug 4709338.

- Add the `2002::/16`static route to the routing tables of all intrasite routers within the 6to4 site.
- Use a routing protocol other than RIPng on the 6to4 site's internal router.

## Configuring Tunnels with the Same Source Address (Bug ID 4152864)

Bug ID 4152864 describes problems that occur when two tunnels are configured with the same tunnel source address, which is a serious issue for 6to4 tunnels.

**Caution –** Do not configure a 6to4 tunnel and an automatic tunnel (`atun`) with the same tunnel source address. For information about automatics and the `atun` command, refer to the `tun`(7M) man page.

◆ ◆ ◆ CHAPTER 10

# TCP/IP and IPv4 in Depth (Reference)

This chapter provides TCP/IP network reference information about network configuration files, including the types, their purpose, and the format of the file entries. The existing network databases are also described in detail. The chapter also shows how the structure of IPv4 addresses are derived, based on defined network classifications and subnet numbers.

This chapter contains the following information:


- "TCP/IP Configuration Files" on page 235
- "Network Databases and the `nsswitch.conf` File" on page 245
- "Routing Protocols in the Solaris OS" on page 253
- "Network Classes" on page 254


## What's New in TCP/IP and IPv4 in Depth

In the Solaris 10 8/07, the `/etc/inet/ipnodes` file becomes obsolete. Use `/etc/inet/ipnodes` only for earlier Solaris 10 releases, as explained in the individual procedures.

## TCP/IP Configuration Files

Each system on the network obtains its TCP/IP configuration information from the following TCP/IP configuration files and network databases:

- `/etc/hostname.`*interface* file
- `/etc/nodename` file
- `/etc/defaultdomain` file
- `/etc/defaultrouter` file (optional)
- `hosts` database
- In Solaris 10 11/06 and earlier releases, `ipnodes` database
- `netmasks` database (optional)

The Solaris installation program creates these files as part of the installation process. You can also edit the files manually, as explained in this section. The `hosts` and `netmasks` databases are two of the network databases read by the name services available on Solaris networks. "Network Databases and the `nsswitch.conf` File" on page 245 describes in detail the concept of network databases. In Solaris 10 11/06 and earlier releases, for information on the `ipnodes` file, see "`ipnodes` Database" on page 240.

## `/etc/hostname.`*interface* File

This file defines the physical network interfaces on the local host. At least one `/etc/hostname.`*interface* file should exist on the local system. The Solaris installation program creates an `/etc/hostname.`*interface* file for the first interface that is found during the installation process. This interface usually has the lowest device number, for example `eri0`, and is referred to as the *primary network interface*. If the installation programs finds additional interfaces, you optionally can configure them, as well, as part of the installation process.

If you add a new network interface to your system after installation, you must create an `/etc/hostname.`*interface* file for that interface, as explained in "How to Configure a Physical Interface After System Installation" on page 150. Also, for the Solaris software to recognize and use the new network interface, you need to load the interface's device driver into the appropriate directory. Refer to the documentation that comes with the new network interface for the appropriate *interface* name and device driver instructions.

The basic `/etc/hostname.`*interface* file contains one entry: the host name or IPv4 address that is associated with the network interface. The IPv4 address can be expressed in traditional dotted decimal format or in CIDR notation. If you use a host name as the entry for the `/etc/hostname.`*interface* file, that host name must also exist in the `/etc/inet/hosts` file.

For example, suppose `smc0` is the primary network interface for a system that is called `tenere`. The `/etc/hostname.smc0` file could have as its entry an IPv4 address in dotted decimal notation or in CIDR notation, or the host name `tenere`.

---

**Note –** IPv6 uses the `/etc/hostname6.`*interface* file for defining network interfaces. For more information, refer to "IPv6 Interface Configuration File" on page 267.

---

## `/etc/nodename` File

This file should contain one entry: the host name of the local system. For example, on system `timbuktu`, the file `/etc/nodename` would contain the entry `timbuktu`.

## /etc/defaultdomain File

This file should contain one entry: the fully qualified domain name of the administrative domain to which the local host's network belongs. You can supply this name to the Solaris installation program or edit the file at a later date. For more information on network domains, refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

## /etc/defaultrouter File

This file can contain an entry for each router that is directly connected to the network. The entry should be the name for the network interface that functions as a router between networks. The presence of the `/etc/defaultrouter` file indicates that the system is configured to support static routing.

## hosts Database

The `hosts` database contains the IPv4 addresses and host names of systems on your network. If you use the NIS or DNS name service, or the LDAP directory service, the `hosts` database is maintained in a database that is designated for host information. For example, on a network that runs NIS, the `hosts` database is maintained in the `hostsbyname` file.

If you use local files for the name service, the `hosts` database is maintained in the `/etc/inet/hosts` file. This file contains the host names and IPv4 addresses of the primary network interface, other network interfaces that are attached to the system, and any other network addresses that the system must check for.

---

**Note –** For compatibility with BSD-based operating systems, the `/etc/hosts` file is a symbolic link to `/etc/inet/hosts`.

---

### /etc/inet/hosts File Format

The `/etc/inet/hosts` file uses the basic syntax that follows. Refer to the `hosts`(4) man page for complete syntax information.

*IPv4-address hostname [nicknames] [#comment]*

| | |
|---|---|
| *IPv4-address* | Contains the IPv4 address for each interface that the local host must recognize. |
| *hostname* | Contains the host name that is assigned to the system at setup, plus the host names that are assigned to additional network interfaces that the local host must recognize. |
| *[nickname]* | Is an optional field that contains a nickname for the host. |

*[#comment]* Is an optional field for a comment.

## Initial `/etc/inet/hosts` File

When you run the Solaris installation program on a system, the program configures the initial `/etc/inet/hosts` file. This file contains the minimum entries that the local host requires. The entries include the loopback address, the host IPv4 address, and the host name.

For example, the Solaris installation program might create the following `/etc/inet/hosts` file for system `tenere` shown in Figure 5–1:

**EXAMPLE 10–1** `/etc/inet/hosts` File for System `tenere`

```
127.0.0.1     localhost         loghost    #loopback address
192.168.200.3   tenere                        #host name
```

### Loopback Address

In Example 10–1, the IPv4 address `127.0.0.1` is the *loopback address*. The loopback address is the reserved network interface that is used by the local system to allow interprocess communication. This address enables the host to send packets to itself. The `ifconfig` command uses the loopback address for configuration and testing, as explained in "Monitoring the Interface Configuration With the `ifconfig` Command" on page 204. Every system on a TCP/IP network must use the IP address `127.0.0.1` for IPv4 loopback on the local host.

### Host Name

The IPv4 address `192.168.200.1` and the name `tenere` are the address and host name of the local system. They are assigned to the system's primary network interface.

### Multiple Network Interfaces

Some systems have more than one network interface, because they are either routers or multihomed hosts. Each network interface that is attached to the system requires its own IP address and associated name. During installation, you must configure the primary network interface. If a particular system has multiple interfaces at installation time, the Solaris installation program also prompts you about these additional interfaces. You can optionally configure one or more additional interfaces at this time, or manually, at a later date.

After Solaris installation, you can configure additional interfaces for a router or multihomed host by adding interface information to the systems' `/etc/inet/hosts` file. For more information on configuring routers and multihomed hosts refer to "Configuring an IPv4 Router" on page 115 and "Configuring Multihomed Hosts" on page 124.

Example 10–2 shows the /etc/inet/hosts file for system timbuktu that is shown in Figure 5–1.

**EXAMPLE 10–2** /etc/inet/hosts File for System timbuktu

```
127.0.0.1        localhost     loghost
192.168.200.70   timbuktu      #This is the local host name
192.168.201.10   timbuktu-201  #Interface to network 192.9.201
```

With these two interfaces, timbuktu connects networks 192.168.200 and 192.168.201 as a router.

## How Name Services Affect the hosts Database

The NIS and DNS name services, and LDAP directory service, maintain host names and addresses on one or more servers. These servers maintain hosts databases that contain information for every host and router (if applicable) on the servers' network. Refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for more information about these services.

### When Local Files Provide the Name Service

On a network that uses local files for the name service, systems that run in local files mode consult their individual /etc/inet/hosts files for IPv4 addresses and host names of other systems on the network. Therefore, these system's /etc/inet/hosts files must contain the following:

- Loopback address
- IPv4 address and host name of the local system (primary network interface)
- IPv4 address and host name of additional network interfaces that are attached to this system, if applicable
- IPv4 addresses and host names of all hosts on the local network
- IPv4 addresses and host names of any routers that this system must know about, if applicable
- IPv4 address of any system your system wants to refer to by its host name

Figure 10–1 shows the /etc/inet/hosts file for system tenere. This system runs in local files mode. Notice that the file contains the IPv4 addresses and host names for every system on the 192.9.200 network. The file also contains the IPv4 address and interface name timbuktu-201. This interface connects the 192.9.200 network to the 192.9.201 network.

A system that is configured as a network client uses the local /etc/inet/hosts file for its loopback address and IPv4 address.

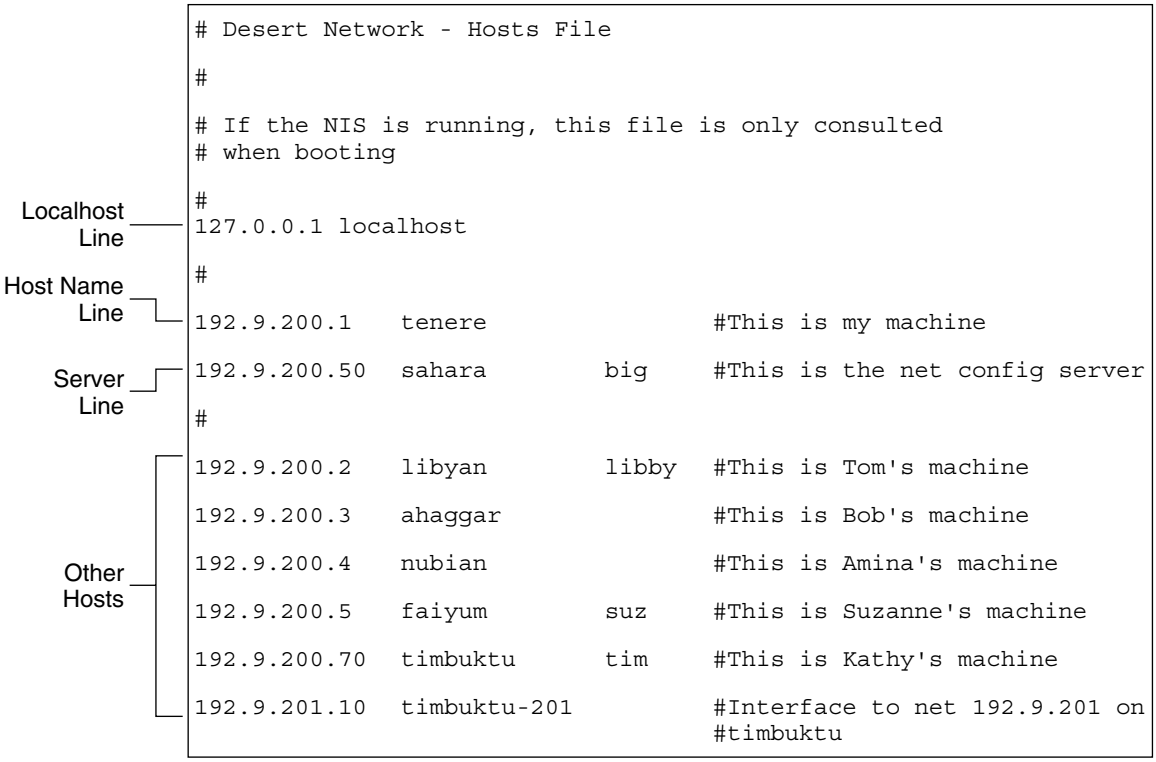



**FIGURE 10–1** `/etc/inet/hosts` File for a System Running in Local Files Mode

## `ipnodes` Database

**Note –** The `ipnodes` database is no longer included in releases after Solaris 10 11/06. In these subsequent releases, the IPv6 features of `ipnodes` migrate into the `hosts` database.

The `/etc/inet/ipnodes` file stores both IPv4 and IPv6 addresses. Moreover, you can store IPv4 addresses in either traditional dotted decimal or CIDR notation. This file serves as a local database that associates the names of hosts with their IPv4 and IPv6 addresses. Do not store host names and their addresses in static files, such as `/etc/inet/ipnodes`. However, for testing purposes, store IPv6 addresses in a file in the same way that IPv4 addresses are stored in `/etc/inet/hosts`. The `ipnodes` file uses the same format convention as the `hosts` file. For more information on `/etc/inet/hosts`, refer to "`hosts` Database" on page 237. See the `ipnodes`(4) man page for a description of the `ipnodes` file.

IPv6-enabled applications use the `/etc/inet/ipnodes` database. The existing `/etc/hosts` database, which contains only IPv4 addresses, remains the same to facilitate existing applications. If the `ipnodes` database does not exist, IPv6-enabled applications use the existing `hosts` database.

---

**Note –** If you need to add addresses, you must add IPv4 addresses to both the `hosts` and `ipnodes` files. You add IPv6 addresses to the `ipnodes` file only.

---

**EXAMPLE 10–3** `/etc/inet/ipnodes` File

You must group host name addresses by the host name, as shown in this example.

```
#
# Internet IPv6 host table
# with both IPv4 and IPv6 addresses
#
::1     localhost
2001:db8:3b4c:114:a00:20ff:fe78:f37c   farsite.com farsite farsite-v6
fe80::a00:20ff:fe78:f37c     farsite-11.com farsitell
192.168.85.87                    farsite.com farsite farsite-v4
2001:db8:86c0:32:a00:20ff:fe87:9aba   nearsite.com nearsite nearsite-v6
fe80::a00:20ff:fe87:9aba     nearsite-11.com nearsitell
10.0.0.177                   nearsite.com nearsite nearsite-v4 loghost
```

# `netmasks` Database

You need to edit the `netmasks` database as part of network configuration *only* if you have set up subnetting on your network. The `netmasks` database consists of a list of networks and their associated subnet masks.

---

**Note –** When you create subnets, each new network must be a separate physical network. You cannot apply subnetting to a single physical network.

---

## What Is Subnetting?

*Subnetting* is a method for maximizing the limited 32-bit IPv4 addressing space and reducing the size of the routing tables in a large internetwork. With any address class, subnetting provides a means of allocating a part of the host address space to network addresses, which lets you have more networks. The part of the host address space that is allocated to new network addresses is known as the *subnet number*.

In addition to making more efficient use of the IPv4 address space, subnetting has several administrative benefits. Routing can become very complicated as the number of networks grows. A small organization, for example, might give each local network a class C number. As the organization grows, the administration of a number of different network numbers could become complicated. A better idea is to allocate a few class B network numbers to each major division in an organization. For example, you could allocate one Class B network to Engineering, one Class B to Operations, and so on. Then, you could divide each class B network into additional networks, using the additional network numbers gained by subnetting. This division can also reduce the amount of routing information that must be communicated among routers.

## Creating the Network Mask for IPv4 Addresses

As part of the subnetting process, you need to select a network-wide *netmask*. The netmask determines how many and which bits in the host address space represent the subnet number and how many and which bits represent the host number. Recall that the complete IPv4 address consists of 32 bits. Depending on the address class, as many as 24 bits and as few as 8 bits can be available for representing the host address space. The netmask is specified in the `netmasks` database.

If you plan to use subnets, you must determine your netmask before you configure TCP/IP. If you plan to install the operating system as part of network configuration, the Solaris installation program requests the netmask for your network.

As described in "Designing an IPv4 Addressing Scheme" on page 58, 32-bit IP addresses consist of a network part and a host part. The 32 bits are divided into 4 bytes. Each byte is assigned to either the network number or the host number, depending on the network class.

For example, in a class B IPv4 address, the 2 bytes on the left are assigned to the network number, and the 2 bytes on the right are assigned to the host number. In the class B IPv4 address `172.16.10`, you can assign the 2 bytes on the right to hosts.

If you are to implement subnetting, you need to use some of the bits in the bytes that are assigned to the host number to apply to subnet addresses. For example, a 16-bit host address space provides addressing for 65,534 hosts. If you apply the third byte to subnet addresses and the fourth byte to host addresses, you can address up to 254 networks, with up to 254 hosts on each network.

The bits in the host address bytes that are applied to subnet addresses and those applied to host addresses are determined by a *subnet mask*. Subnet masks are used to select bits from either byte for use as subnet addresses. Although netmask bits must be contiguous, they need not align on byte boundaries.

The netmask can be applied to an IPv4 address by using the bitwise logical AND operator. This operation selects out the network number and subnet number positions of the address.

Netmasks can be explained in terms of their binary representation. You can use a calculator for binary-to-decimal conversion. The following examples show both the decimal and binary forms of the netmask.

If a netmask `255.255.255.0` is applied to the IPv4 address `172.16.41.101`, the result is the IPv4 address of `172.16.41.0`.

`172.16.41.101` & `255.255.255.0` = `172.16.41.0`

In binary form, the operation is as follows:

10000001.10010000.00101001.01100101 (IPv4 address)

ANDed with

11111111.11111111.11111111.00000000 (netmask)

Now the system looks for a network number of `172.16.41` instead of a network number of `172.16`. If your network has the number `172.16.41`, that number is what the system checks for and finds. Because you can assign up to 254 values to the third byte of the IPv4 address space, subnetting lets you create address space for 254 networks, where previously space was available for only one.

If you are providing address space for only two additional networks, you can use the following subnet mask:

`255.255.192.0`

This netmask provides the following result:

11111111.11111111.1100000.00000000

This result still leaves 14 bits available for host addresses. Because all 0s and 1s are reserved, at least 2 bits must be reserved for the host number.

## `/etc/inet/netmasks` File

If your network runs NIS or LDAP, the servers for these name services maintain `netmasks` databases. For networks that use local files for the name service, this information is maintained in the `/etc/inet/netmasks` file.

---

**Note –** For compatibility with BSD-based operating systems, the `/etc/netmasks` file is a symbolic link to `/etc/inet/netmasks`.

---

The following example shows the `/etc/inet/netmasks` file for a class B network.

**EXAMPLE 10–4** /etc/inet/netmasks File for a Class B Network

```
# The netmasks file associates Internet Protocol (IPv4) address
# masks with IPv4 network numbers.
#
#     network-number    netmask
#
# Both the network-number and the netmasks are specified in
# “decimal dot” notation, e.g:
#
#        128.32.0.0   255.255.255.0
192.168.0.0  255.255.255.0
```

If the /etc/netmasks file does not exist, create it with a text editor. Use the following syntax:

*network-number  netmask-number*

Refer to the netmasks(4) man page for complete details.

When creating netmask numbers, type the network number that is assigned by the ISP or Internet Registry (not the subnet number) and the netmask number in /etc/inet/netmasks. Each subnet mask should be on a separate line.

For example:

```
128.78.0.0        255.255.248.0
```

You can also type symbolic names for network numbers in the /etc/inet/hosts file. You can then use these network names instead of the network numbers as parameters to commands.

# inetd Internet Services Daemon

The inetd daemon starts up Internet standard services when a system boots, and can restart a service while a system is running. Use the Service Management Facility (SMF) to modify the standard Internet services or to have additional services started by the inetd daemon.

Use the following SMF commands to manage services started by inetd:

| | |
|---|---|
| svcadm | For administrative actions on a service, such as enabling, disabling, or restarting. For details, refer to the svcadm(1M) man page. |
| svcs | For querying the status of a service. For details, refer to the svcs(1) man page. |
| inetadm | For displaying and modifying the properties of a service. For details, refer to the inetadm(1M) man page. |

The proto field value in the inetadm profile for a particular service indicates the transport layer protocol on which the service runs. If the service is IPv4-only, the proto field must be specified as tcp, udp, or sctp.

- For instructions on using the SMF commands, refer to "SMF Command-Line Administrative Utilities" in *System Administration Guide: Basic Administration*.
- For a task that uses the SMF commands to add a service that runs over SCTP, refer to "How to Add Services That Use the SCTP Protocol" on page 133.
- For information on adding services that handle both IPv4 requests and IPv6 requests, refer to "inetd Internet Services Daemon" on page 244

# Network Databases and the nsswitch.conf File

The network databases are files that provide information that is needed to configure the network. The network databases follow:

- hosts
- netmasks
- ethers database
- bootparams
- protocols
- services
- networks

As part of the configuration process, you edit the hosts database and the netmasks database, if your network is subnetted. Two network databases, bootparams and ethers, are used to configure systems as network clients. The remaining databases are used by the operating system and seldom require editing.

Although nsswitch.conf file is not a network database, you need to configure this file along with the relevant network databases. nsswitch.conf specifies which name service to use for a particular system: local files, NIS, DNS, or LDAP.

## How Name Services Affect Network Databases

The format of your network database depends on the type of name service you select for your network. For example, the hosts database contains, at least the host name and IPv4 address of the local system and any network interfaces that are directly connected to the local system. However, the hosts database could contain other IPv4 addresses and host names, depending on the type of name service on your network.

The network databases are used as follows:

- Networks that use local files for their name service rely on files in the /etc/inet and /etc directories.
- NIS uses databases that are called NIS maps.
- DNS uses records with host information.

**Note –** DNS boot and data files do not correspond directly to the network databases.

The following figure shows the forms of the hosts database that are used by these name services.

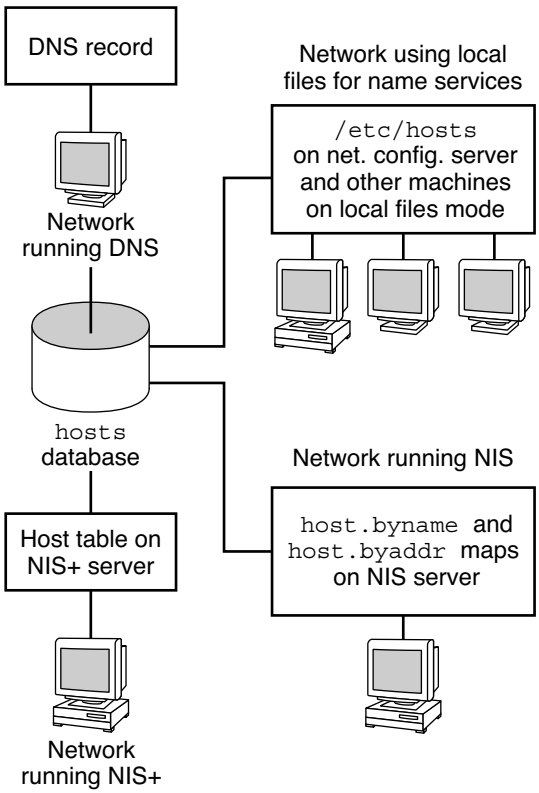



**FIGURE 10–2** Forms of the hosts Database Used by Name Services

The following table lists the network databases and their corresponding local files and NIS maps.

**Note –** The ipnodes database is removed from Solaris releases after Solaris 10 11/06.

**TABLE 10–1** Network Databases and Corresponding Name Service Files

| Network Database | Local Files | NIS Maps |
|---|---|---|
| hosts | /etc/inet/hosts | hosts.byaddr hosts.byname |
| ipnodes | /etc/inet/ipnodes | ipnodes.byaddr ipnodes.byname |
| netmasks | /etc/inet/netmasks | netmasks.byaddr |
| ethers | /etc/ethers | ethers.byname ethers.byaddr |
| bootparams | /etc/bootparams | bootparams |
| protocols | /etc/inet/protocols | protocols.byname protocols.bynumber |
| services | /etc/inet/services | services.byname |
| networks | /etc/inet/networks | networks.byaddr networks.byname |

This book discusses network databases as they are viewed by networks that use local files for name services.

- Information about the hosts database is in "hosts Database" on page 237.
- Information about the netmasks database is in "netmasks Database" on page 241.
- For Solaris 10 11/06 and earlier releases, information about the ipnodes database is in "ipnodes Database" on page 240.

Refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for information on network databases correspondences in NIS, DNS, and LDAP.

## nsswitch.conf File

The /etc/nsswitch.conf file defines the search order of the network databases. The Solaris installation program creates a default /etc/nsswitch.conf file for the local system, based on the name service you indicate during the installation process. If you selected the "None" option, indicating local files for name service, the resulting nsswitch.conf file resembles the following example.

**EXAMPLE 10–5** nsswitch.conf for Networks Using Files for Name Service

```
# /etc/nsswitch.files:
#
# An example file that could be copied over to /etc/nsswitch.conf;
```

**EXAMPLE 10–5** nsswitch.conf for Networks Using Files for Name Service *(Continued)*

```
# it does not use any naming service.
#
# "hosts:" and "services:" in this file are used only if the
# /etc/netconfig file contains "switch.so" as a
# nametoaddr library for "inet" transports.

passwd:          files
group:           files
hosts:           files
networks:        files
protocols:       files
rpc:             files
ethers:          files
netmasks:        files
bootparams:      files
publickey:       files
# At present there isn't a 'files' backend for netgroup; the
# system will figure it out pretty quickly,
# and won't use netgroups at all.
netgroup:        files
automount:       files
aliases:         files
services:        files
sendmailvars:    files
```

The nsswitch.conf(4) man page describes the file in detail. The basic syntax is shown here:

*database name-service-to-search*

The *database* field can list one of many types of databases that are searched by the operating system. For example, the field could indicate a database that affects users, such as passwd or aliases, or a network database. The parameter *name-service-to-search* can have the values files, nis, or nis+ for the network databases. The hosts database can also have dns as a name service to search. You can also list more than one name service, such as nis+ and files.

In Example 10–5, the only search option that is indicated is files. Therefore, the local system obtains security and automounting information, in addition to network database information, from files that are located in its /etc and /etc/inet directories.

## Changing nsswitch.conf

The /etc directory contains the nsswitch.conf file that is created by the Solaris installation program. This directory also contains template files for the following name services:

- nsswitch.files

- nsswitch.nis
- nsswitch.nis+

If you want to change from one name service to another name service, you can copy the appropriate template to nsswitch.conf. You can also selectively edit the nsswitch.conf file, and change the default name service to search for individual databases.

For example, on a network that runs NIS, you might have to change the nsswitch.conf file on network clients. The search path for the bootparams and ethers databases must list files as the first option, and then nis. The following example shows the correct search paths.

**EXAMPLE 10–6** nsswitch.conf for a Client on a Network Running NIS

```
# /etc/nsswitch.conf:#
.
.
passwd:         files nis
group:          file nis

# consult /etc "files" only if nis is down.
hosts:          nis    [NOTFOUND=return] files
networks:       nis    [NOTFOUND=return] files
protocols:      nis    [NOTFOUND=return] files
rpc:            nis    [NOTFOUND=return] files
ethers:         files  [NOTFOUND=return] nis
netmasks:       nis    [NOTFOUND=return] files
bootparams:     files  [NOTFOUND=return] nis
publickey:      nis
netgroup:       nis

automount:      files nis
aliases:        files nis

# for efficient getservbyname() avoid nis
services:       files nis
sendmailvars:   files
```

For complete details on the name service switch, refer to *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

## bootparams Database

The bootparams database contains information that is used by systems that are configured to boot in network client mode. You need to edit this database if your network has network clients. See "Configuring Network Clients" on page 104 for the procedures. The database is built from information that is entered into the /etc/bootparams file.

The bootparams(4) man page contains the complete syntax for this database. Basic syntax is shown here:

*system-name file-key-server-name:pathname*

For each network client system, the entry might contain the following information: the name of the client, a list of keys, the names of servers, and path names. The first item of each entry is the name of the client system. All items but the first item are optional. An example follows.

**EXAMPLE 10–7** bootparams Database

```
myclient   root=myserver : /nfsroot/myclient  \
swap=myserver : /nfsswap//myclient \
dump=myserver : /nfsdump/myclient
```

In this example, the term dump= tells client hosts not to look for a dump file.

### Wildcard Entry for bootparams

In most instances, use the wildcard entry when editing the bootparams database to support clients. This entry follows:

```
*  root=server:/path dump=:
```

The asterisk (*) wildcard indicates that this entry applies to all clients that are not specifically named within the bootparams database.

## ethers Database

The ethers database is built from information that is entered into the /etc/ethers file. This database associates host names to their *Media Access Control* (MAC) addresses. You need to create an ethers database only if you are running the RARP daemon. That is, you need to create this database if you are configuring network clients.

RARP uses the file to map MAC addresses to IP addresses. If you are running the RARP daemon in.rarpd, you need to set up the ethers file and maintain this file on all hosts that are running the daemon to reflect changes to the network.

The ethers(4) man page contains the complete syntax for this database. The basic syntax is shown here:

*MAC-address hostname #comment*

| | |
|---|---|
| *MAC-address* | MAC address of the host |
| *hostname* | Official name of the host |

*#comment* Any note that you want to append to an entry in the file

The equipment manufacturer provides the MAC address. If a system does not display the MAC address during the system booting process, see your hardware manuals for assistance.

When adding entries to the ethers database, ensure that host names correspond to the primary names in the hosts and, for Solaris 10 11/06 and earlier releases, the ipnodes database, not to the nicknames, as follows.

**EXAMPLE 10–8** Entries in the ethers Database

```
8:0:20:1:40:16  fayoum
8:0:20:1:40:15  nubian
8:0:20:1:40:7   sahara    # This is a comment
8:0:20:1:40:14  tenere
```

# Other Network Databases

The remaining network databases seldom need to be edited.

## networks database

The networks database associates network names with network numbers, enabling some applications to use and display names rather than numbers. The networks database is based on information in the /etc/inet/networks file. This file contains the names of all networks to which your network connects through routers.

The Solaris installation program configures the initial networks database. However, if you add a new network to your existing network topology, you must update this database.

The networks(4) man page contains the complete syntax for /etc/inet/networks. The basic format is shown here:

*network-name network-number nickname(s) #comment*

*network-name* Official name for the network

*network-number* Number assigned by the ISP or Internet Registry

*nickname* Any other name by which the network is known

*#comment* Any note that you want to append to an entry in the file

You must maintain the networks file. The netstat program uses the information in this database to produce status tables.

A sample /etc/networks file follows.

**EXAMPLE 10–9** /etc/networks File

```
#ident    "@(#)networks    1.4    92/07/14 SMI"    /* SVr4.0 1.1    */
#
# The networks file associates Internet Protocol (IP) network
# numbers with network names. The format of this file is:
#
#     network-name              network-number              nicnames . . .

# The loopback network is used only for intra-machine communication
loopback              127

#
# Internet networks
#
arpanet     10       arpa  # Historical
#
# local networks

eng   192.168.9 #engineering
acc   192.168.5 #accounting
prog  192.168.2 #programming
```

## protocols Database

The protocols database lists the TCP/IP protocols that are installed on your system and their protocol numbers. The Solaris installation program automatically creates the database. This file seldom requires any administration.

The protocols(4) man page describes the syntax of this database. An example of the /etc/inet/protocols file follows.

**EXAMPLE 10–10** /etc/inet/protocols File

```
#
# Internet (IP) protocols
#
ip    0   IP    # internet protocol, pseudo protocol number
icmp  1   ICMP  # internet control message protocol
tcp   6   TCP   # transmission control protocol
udp  17   UDP   # user datagram protocol
```

### `services` Database

The `services` database lists the names of TCP and UDP services and their well-known port numbers. This database is used by programs that call network services. The Solaris installation automatically creates the `services` database. Generally, this database does not require any administration.

The `services`(4) man page contains complete syntax information. An excerpt from a typical `/etc/inet/services` file follows.

**EXAMPLE 10–11** `/etc/inet/services` File

```
#
# Network services
#
echo      7/udp
echo      7/tcp
echo      7/sctp6
discard   9/udp     sink null
discard   11/tcp
daytime   13/udp
daytime   13/tcp
netstat   15/tcp
ftp-data  20/tcp
ftp       21/tcp
telnet    23/tcp
time      37/tcp    timeserver
time      37/udp    timeserver
name      42/udp    nameserver
whois     43/tcp    nickname
```

## Routing Protocols in the Solaris OS

This section describes two routing protocols supported in the Solaris 10 OS: Routing Information Protocol (RIP) and ICMP Router Discovery (RDISC). RIP and RDISC are both standard TCP/IP protocols. For complete lists of routing protocols available in the Solaris 10 OS, refer to Table 5–1 and Table 5–2.

## Routing Information Protocol (RIP)

RIP is implemented by `in.routed`, the routing daemon, which automatically starts when the system boots. When run on a router with the `s` option specified, `in.routed` fills the kernel routing table with a route to every reachable network and advertises "reachability" through all network interfaces.

When run on a host with the `q` option specified, `in.routed` extracts routing information but does not advertise reachability. On hosts, routing information can be extracted in two ways:

- Do *not* specify the `S` flag (capital "S": "Space-saving mode"). `in.routed` builds a full routing table exactly as it does on a router.
- Specify the `S` flag. `in.routed` creates a minimal kernel table, containing a single default route for each available router.

## ICMP Router Discovery (RDISC) Protocol

Hosts use RDISC to obtain routing information from routers. Thus, when hosts are running RDISC, routers must also run another protocol, such as RIP, in order to exchange router information.

RDISC is implemented by `in.routed`, which should run on both routers and hosts. On hosts, `in.routed` uses RDISC to discover default routes from routers that advertise themselves through RDISC. On routers, `in.routed` uses RDISC to advertise default routes to hosts on directly-connected networks. See the `in.routed`(1M) man page and the `gateways`(4) man page.

# Network Classes

**Note –** Class-based network numbers are no longer available from the IANA, though many older networks are still class-based.

This section provides details about IPv4 network classes. Each class uses the 32-bit IPv4 address space differently, providing more or fewer bits for the network part of the address. These classes are class A, class B, and class C.

## Class A Network Numbers

A class A network number uses the first 8 bits of the IPv4 address as its "network part." The remaining 24 bits contain the host part of the IPv4 address, as the following figure illustrates.

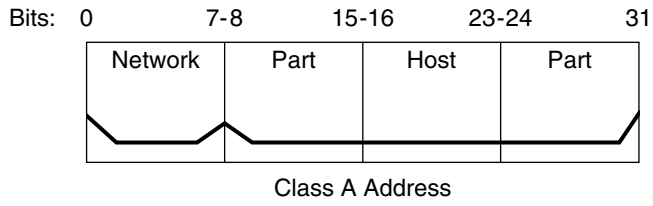



**FIGURE 10–3** Byte Assignment in a Class A Address

The values that are assigned to the first byte of class A network numbers fall within the range 0–127. Consider the IPv4 address `75.4.10.4`. The value 75 in the first byte indicates that the host is on a class A network. The remaining bytes, `4.10.4`, establish the host address. Only the first byte of a class A number is registered with the IANA. Use of the remaining three bytes is left to the discretion of the owner of the network number. Only 127 class A networks exist. Each one of these numbers can accommodate a maximum of 16,777,214 hosts.

## Class B Network Numbers

A class B network number uses 16 bits for the network number and 16 bits for host numbers. The first byte of a class B network number is in the range 128–191. In the number `172.16.50.56`, the first two bytes, `172.16`, are registered with the IANA, and compose the network address. The last two bytes, `50.56`, contain the host address, and are assigned at the discretion of the owner of the network number. The following figure graphically illustrates a class B address.

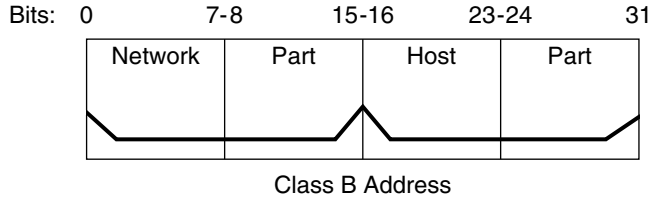



**FIGURE 10–4** Byte Assignment in a Class B Address

Class B is typically assigned to organizations with many hosts on their networks.

## Class C Network Numbers

Class C network numbers use 24 bits for the network number and 8 bits for host numbers. Class C network numbers are appropriate for networks with few hosts—the maximum being 254. A

class C network number occupies the first three bytes of an IPv4 address. Only the fourth byte is assigned at the discretion of the network owners. The following figure graphically represents the bytes in a class C address.

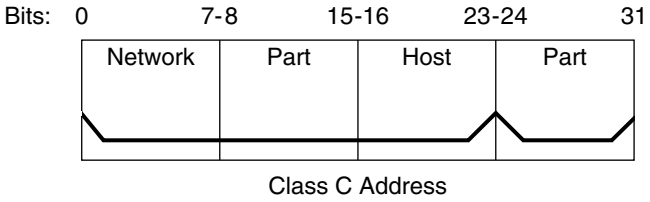



**FIGURE 10–5** Byte Assignment in a Class C Address

The first byte of a class C network number covers the range 192–223. The second and third bytes each cover the range 1– 255. A typical class C address might be `192.168.2.5`. The first three bytes, `192.168.2`, form the network number. The final byte in this example, `5`, is the host number.

◆ ◆ ◆ CHAPTER 11

# IPv6 in Depth (Reference)

This chapter contains the following reference information about the Solaris 10 IPv6 implementation.


- "IPv6 Addressing Formats Beyond the Basics" on page 258
- "IPv6 Packet Header Format" on page 261
- "Dual-Stack Protocols" on page 262
- "Solaris 10 IPv6 Implementation" on page 263
- "IPv6 Neighbor Discovery Protocol" on page 278
- "IPv6 Routing" on page 284
- "IPv6 Tunnels" on page 285
- "IPv6 Extensions to Solaris Name Services" on page 293
- "NFS and RPC IPv6 Support" on page 295
- "IPv6 Over ATM Support" on page 295


For an overview of IPv6, refer to Chapter 3, "Introducing IPv6 (Overview)." For tasks on configuring an IPv6-enabled network, refer to Chapter 7, "Configuring an IPv6 Network (Tasks)."

## What's New in IPv6 in Depth

In the Solaris 10 8/07, the `/etc/inet/ipnodes` file becomes obsolete. Use `/etc/inet/ipnodes` only for earlier Solaris 10 releases, as explained in the individual procedures.

# IPv6 Addressing Formats Beyond the Basics

Chapter 3, "Introducing IPv6 (Overview)," introduces the most common IPv6 addressing formats: unicast site address and link-local address. This section includes in-depth explanations of addressing formats that are not covered in detail in Chapter 3, "Introducing IPv6 (Overview),":


- "6to4-Derived Addresses" on page 258
- "IPv6 Multicast Addresses in Depth" on page 260


## 6to4-Derived Addresses

If you plan to configure a 6to4 tunnel from a router or host endpoint, you must advertise the 6to4 site prefix in the `/etc/inet/ndpd.conf` file on the endpoint system. For an introduction and tasks for configuring 6to4 tunnels, refer to "How to Configure a 6to4 Tunnel" on page 192.

The next figure shows the parts of a 6to4 site prefix.

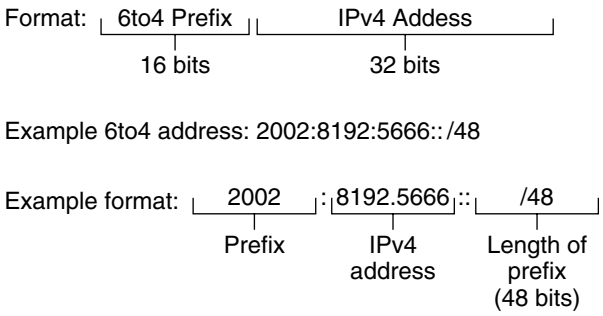



**FIGURE 11–1** Parts of a 6to4 Site Prefix

The next figure shows the parts of a subnet prefix for a 6to4 site, such as you would include in the `ndpd.conf` file.

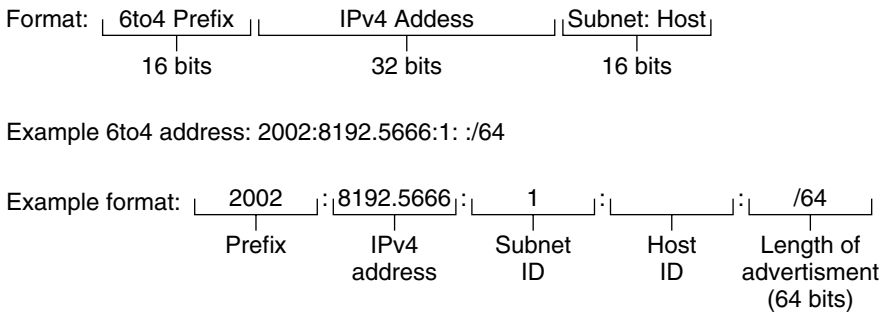



**FIGURE 11–2** Parts of a 6to4 Subnet Prefix

This table explains the parts of a 6to4 subnet prefix.

| Part | Length | Definition |
|---|---|---|
| Prefix | 16 bits | 6to4 prefix label 2002 (0x2002). |
| IPv4 address | 32 bits | Unique IPv4 address that is already configured on the 6to4 interface. For the advertisement, you specify the hexadecimal representation of the IPv4 address, rather than the IPv4 dotted-decimal representation. |
| Subnet ID | 16 bits | Subnet ID, which must be a value that is unique for the link at your 6to4 site. |

## 6to4-Derived Addressing on a Host

When an IPv6 host receives the 6to4-derived prefix by way of a router advertisement, the host automatically reconfigures a 6to4-derived address on an interface. The address has the following format:

*prefix:IPv4-address:subnet-ID:interface-ID*/64

The output from the `ifconfig -a` command on a host with a 6to4 interface might resemble the following:

```
qfe1:3: flags=2180841<UP,RUNNING,MULTICAST,ADDRCONF,ROUTER,IPv6>
 mtu 1500 index 7
        inet6 2002:8192:56bb:9258:a00:20ff:fea9:4521/64
```

In this output, the 6to4-derived address follows `inet6`.

This table explains the parts of the 6to4-derived address.

| Address Part | Length | Definition |
|---|---|---|
| *prefix* | 16 bits | `2002`, which is the 6to4 prefix |
| *IPv4-address* | 32 bits | `8192:56bb`, which is the IPv4 address, in hexadecimal notation, for the 6to4 pseudo-interface that is configured on the 6to4 router |
| *subnet-ID* | 16 bits | `9258`, which is the address of the subnet of which this host is a member |
| *interface-ID* | 64 bits | `a00:20ff:fea9:4521`, which is the interface ID of the host interface that is configured for 6to4 |

# IPv6 Multicast Addresses in Depth

The IPv6 multicast address provides a method for distributing identical information or services to a defined group of interfaces, called the *multicast group*. Typically, the interfaces of the multicast group are on different nodes. An interface can belong to any number of multicast groups. Packets sent to the multicast address go to all members of the multicast group. For example, one use of multicast addresses is for broadcasting information, similar to the capability of the IPv4 broadcast address.

The following table shows the format of the multicast address.

**TABLE 11–1** IPv6 Multicast Address Format

| 8 bits | 4 bits | 4 bits | 8 bits | 8 bits | 64 bits | 32 bits |
|---|---|---|---|---|---|---|
| `11111111` | *FLGS* | `SCOP` | *Reserved* | *Plen* | *Network prefix* | *Group ID* |

The following is a summary of the contents of each field.

- `11111111` – Identifies the address as a multicast address.
- *FLGS* – Set of the four flags 0,0,P,T. The first two flags must be zero. The P field has one of the following values:
  - 0 = Multicast address that is not assigned based on the network prefix
  - 1 = Multicast address that is assigned based on the network prefix

  If P is set to 1, then T must also be 1.
- *Reserved* - Reserved value of zero.
- *Plen* - Number of bits in the site prefix that identify the subnet, for a multicast address that is assigned based on a site prefix.
- *Group ID* - Identifier for the multicast group, either permanent or dynamic.

For complete details about the multicast format, refer to RFC 3306, "Unicast-Prefix-based IPv6 Multicast Addresses (`ftp://ftp.rfc-editor.org/in-notes/rfc3306.txt`).

Some IPv6 multicast addresses are permanently assigned by the Internet Assigned Numbers Authority (IANA). Some examples are the All Nodes Multicast Addresses and All Routers Multicast Addresses that are required by all IPv6 hosts and IPv6 routers. IPv6 multicast addresses can also be dynamically allocated. For more information about the proper use of multicast addresses and groups, see RFC 3307, "Allocation Guidelines for IPv6 Multicast Addresses".

# IPv6 Packet Header Format

The IPv6 protocol defines a set of headers, including the basic IPv6 header and the IPv6 extension headers. The following figure shows the fields that appear in the IPv6 header and the order in which the fields appear.

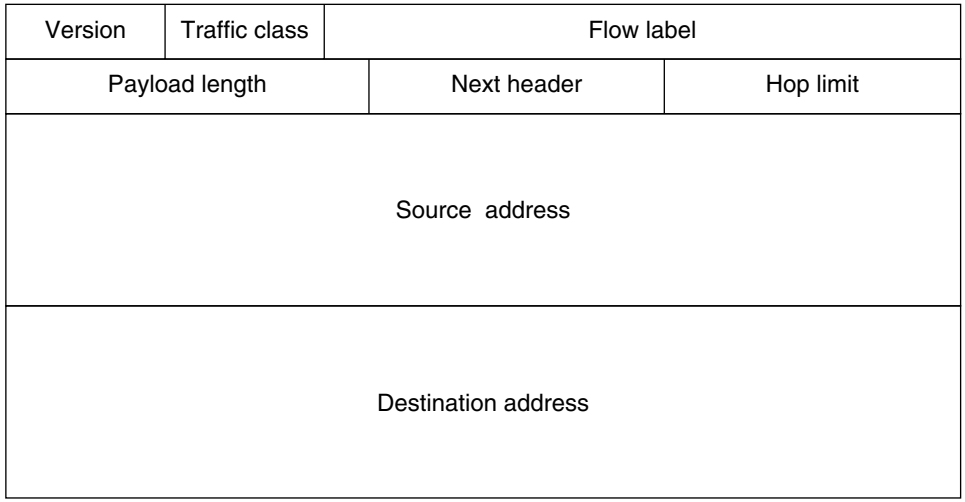



**FIGURE 11–3** IPv6 Basic Header Format

The following list describes the function of each header field.

- **Version** – 4-bit version number of Internet Protocol = 6.
- **Traffic class** – 8-bit traffic class field.
- **Flow label** – 20-bit field.
- **Payload length** – 16-bit unsigned integer, which is the rest of the packet that follows the IPv6 header, in octets.
- **Next header** – 8-bit selector. Identifies the type of header that immediately follows the IPv6 header. Uses the same values as the IPv4 protocol field.

- **Hop limit** – 8-bit unsigned integer. Decremented by one by each node that forwards the packet. The packet is discarded if the hop limit is decremented to zero.
- **Source address** – 128 bits. The address of the initial sender of the packet.
- **Destination address** – 128 bits. The address of the intended recipient of the packet. The intended recipient is not necessarily the recipient if an optional routing header is present.

## IPv6 Extension Headers

IPv6 options are placed in separate extension headers that are located between the IPv6 header and the transport-layer header in a packet. Most IPv6 extension headers are not examined or processed by any router along a packet's delivery path until the packet arrives at its final destination. This feature provides a major improvement in router performance for packets that contain options. In IPv4, the presence of any options requires the router to examine all options.

Unlike IPv4 options, IPv6 extension headers can be of arbitrary length. Also, the number of options that a packet carries is not limited to 40 bytes. This feature, in addition to the manner in which IPv6 options are processed, permits IPv6 options to be used for functions that are not practical in IPv4.

To improve performance when handling subsequent option headers, and the transport protocol that follows, IPv6 options are always an integer multiple of 8 octets long. The integer multiple of 8 octets retains the alignment of subsequent headers.

The following IPv6 extension headers are currently defined:

- **Routing** – Extended routing, such as IPv4 loose source route
- **Fragmentation** – Fragmentation and reassembly
- **Authentication** – Integrity and authentication, and security
- **Encapsulating Security Payload** – Confidentiality
- **Hop-by-Hop options** – Special options that require hop-by-hop processing
- **Destination options** – Optional information to be examined by the destination node

# Dual-Stack Protocols

The term *dual-stack* normally refers to a complete duplication of all levels in the protocol stack from applications to the network layer. One example of complete duplication is a system that runs both the OSI and TCP/IP protocols.

The Solaris OS is *dual-stack*, meaning that the Solaris OS implements both IPv4 and IPv6 protocols. When you install the operating system, you can choose to enable the IPv6 protocols in the IP layer or use only the default IPv4 protocols. The remainder of the TCP/IP stack is

identical. Consequently, the same transport protocols, TCP UDP and SCTP, can run over both IPv4 and IPv6. Also, the same applications can run over both IPv4 and IPv6. Figure 11–4 shows how the IPv4 and IPv6 protocols work as a dual-stack throughout the various layers of the Internet protocol suite.

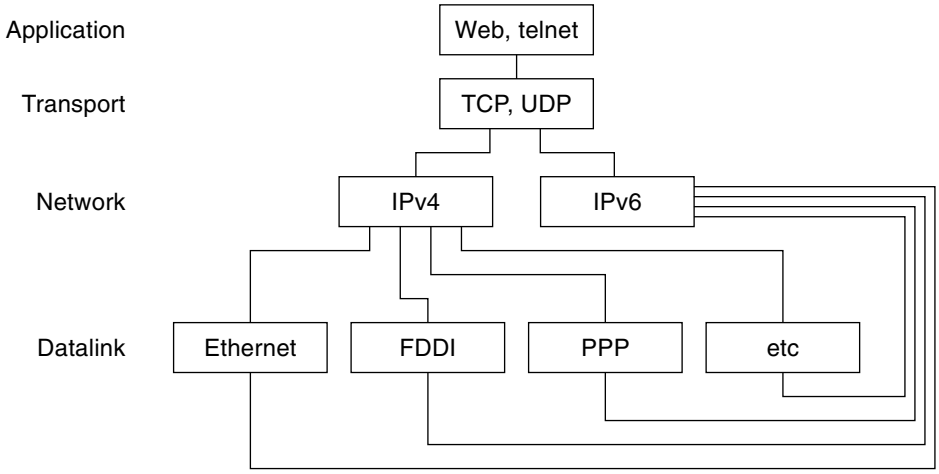



**FIGURE 11–4** Dual-Stack Protocol Architecture

In the dual-stack scenario, subsets of both hosts and routers are upgraded to support IPv6, in addition to IPv4. The dual-stack approach ensures that the upgraded nodes can always interoperate with IPv4-only nodes by using IPv4.

# Solaris 10 IPv6 Implementation

This section describes the files, commands, and daemons that enable IPv6 in the Solaris OS.

## IPv6 Configuration Files

This section describes the configuration files that are part of an IPv6 implementation:


- "`ndpd.conf` Configuration File" on page 263
- "IPv6 Interface Configuration File" on page 267
- "`/etc/inet/ipaddrsel.conf` Configuration File" on page 268


### `ndpd.conf` Configuration File

The `/etc/inet/ndpd.conf` file is used to configure options that are used by the `in.ndpd` Neighbor Discovery daemon. For a router, you primarily use `ndpd.conf` to configure the site prefix to be advertised to the link. For a host, you use `ndpd.conf` to turn off address autoconfiguration or to configure temporary addresses.

The next table shows the keywords that are used in the `ndpd.conf` file.

**TABLE 11–2** `/etc/inet/ndpd.conf` Keywords

| Variable | Description |
|---|---|
| `ifdefault` | Specifies the router behavior for all interfaces. Use the following syntax to set router parameters and corresponding values:<br>`ifdefault [`*variable-value*`]` |
| `prefixdefault` | Specifies the default behavior for prefix advertisements. Use the following syntax to set router parameters and corresponding values:<br>`prefixdefault [`*variable-value*`]` |
| `if` | Sets per-interface parameters. Use the following syntax:<br>`if` *interface* `[`*variable-value*`]` |
| `prefix` | Advertises per-interface prefix information. Use the following syntax:<br>`prefix` *prefix/length interface* `[`*variable-value*`]` |

In the `ndpd.conf` file, you use the keywords in this table with a set of router configuration variables. These variables are defined in detail in RFC 2461, Neighbor Discovery for IP Version 6 (IPv6) (`http://www.ietf.org/rfc/rfc2461.txt?number=2461`).

The next table shows the variables for configuring an interface, along with brief definitions.

**TABLE 11–3** `/etc/inet/ndpd.conf` Interface Configuration Variables

| Variable | Default | Definition |
|---|---|---|
| `AdvRetransTimer` | 0 | Specifies the value in the Retrans Timer field in the advertisement messages sent by the router. |
| `AdvCurHopLimit` | Current diameter of the Internet | Specifies the value to be placed in the current hop limit in the advertisement messages sent by the router. |
| `AdvDefaultLifetime` | 3 + `MaxRtrAdvInterval` | Specifies the default lifetime of the router advertisements. |
| `AdvLinkMTU` | 0 | Specifies a maximum transmission unit (MTU) value to be sent by the router. The zero indicates that the router does not specify MTU options. |
| `AdvManaged Flag` | False | Indicates the value to be placed in the Manage Address Configuration flag in the router advertisement. |
| `AdvOtherConfigFlag` | False | Indicates the value to be placed in the Other Stateful Configuration flag in the router advertisement. |

**TABLE 11–3** `/etc/inet/ndpd.conf` Interface Configuration Variables *(Continued)*

| Variable | Default | Definition |
|---|---|---|
| `AdvReachableTime` | 0 | Specifies the value in the Reachable Time field in the advertisement messages sent by the router. |
| `AdvSendAdvertisements` | False | Indicates whether the node should send out advertisements and respond to router solicitations. You need to explicitly set this variable to "TRUE" in the `ndpd.conf` file to turn on router advertisement functions. For more information, refer to "How to Configure an IPv6-Enabled Router" on page 177. |
| `DupAddrDetect Transmits` | 1 | Defines the number of consecutive neighbor solicitation messages that the Neighbor Discovery protocol should send during duplicate address detection of the local node's address. |
| `MaxRtrAdvInterval` | 600 seconds | Specifies the maximum time to wait between sending unsolicited multicast advertisements. |
| `MinRtrAdvInterval` | 200 seconds | Specifies the minimum time to wait between sending unsolicited multicast advertisements. |
| `StatelessAddrConf` | True | Controls whether the node configures its IPv6 address through stateless address autoconfiguration. If False is declared in `ndpd.conf`, then the address must be manually configured. For more information, refer to "How to Configure a User-Specified IPv6 Token" on page 185. |
| `TmpAddrsEnabled` | False | Indicates whether a temporary address should be created for all interfaces or for a particular interface of a node. For more information, refer to "How to Configure a Temporary Address" on page 182. |
| `TmpMaxDesyncFactor` | 600 seconds | Specifies a random value to be subtracted from the preferred lifetime variable `TmpPreferredLifetime` when `in.ndpd` starts. The purpose of the `TmpMaxDesyncFactor` variable is to prevent all the systems on your network from regenerating their temporary addresses at the same time. `TmpMaxDesyncFactor` allows you to change the upper bound on that random value. |
| `TmpPreferredLifetime` | False | Sets the preferred lifetime of a temporary address. For more information, refer to "How to Configure a Temporary Address" on page 182. |
| `TmpRegenAdvance` | False | Specifies the lead time in advance of address deprecation for a temporary address. For more information, refer to "How to Configure a Temporary Address" on page 182. |
| `TmpValidLifetime` | False | Sets the valid lifetime for a temporary address. For more information, refer to "How to Configure a Temporary Address" on page 182. |

The next table shows the variables that are used for configuring IPv6 prefixes.

**TABLE 11–4** `/etc/inet/ndpd.conf` Prefix Configuration Variables

| Variable | Default | Definition |
|---|---|---|
| `AdvAutonomousFlag` | True | Specifies the value to be placed in the Autonomous Flag field in the Prefix Information option. |
| `AdvOnLinkFlag` | True | Specifies the value to be placed in the on-link flag ("L-bit") in the Prefix Information option. |
| `AdvPreferredExpiration` | Not set | Specifies the preferred expiration date of the prefix. |
| `AdvPreferredLifetime` | 604800 seconds | Specifies the value to be placed in the preferred lifetime in the Prefix Information option. |
| `AdvValidExpiration` | Not set | Specifies the valid expiration date of the prefix. |
| `AdvValidLifetime` | 2592000 seconds | Specifies the valid lifetime of the prefix that is being configured. |

**EXAMPLE 11–1** `/etc/inet/ndpd.conf` File

The following example shows how the keywords and configuration variables are used in the `ndpd.conf` file. Remove the comment (#) to activate the variable.

```
# ifdefault       [variable-value ]*
# prefixdefault [variable-value ]*
# if ifname    [variable-value ]*
# prefix prefix/length ifname
#
#  Per interface configuration variables
#
#DupAddrDetectTransmits
#AdvSendAdvertisements
#MaxRtrAdvInterval
#MinRtrAdvInterval
#AdvManagedFlag
#AdvOtherConfigFlag
#AdvLinkMTU
#AdvReachableTime
#AdvRetransTimer
#AdvCurHopLimit
#AdvDefaultLifetime
#
# Per Prefix:  AdvPrefixList configuration variables
#
#
#AdvValidLifetime
```

**EXAMPLE 11–1** `/etc/inet/ndpd.conf` File *(Continued)*

```
#AdvOnLinkFlag
#AdvPreferredLifetime
#AdvAutonomousFlag
#AdvValidExpiration
#AdvPreferredExpiration

ifdefault AdvReachableTime 30000 AdvRetransTimer 2000
prefixdefault AdvValidLifetime 240m AdvPreferredLifetime 120m

if qe0 AdvSendAdvertisements 1
prefix 2:0:0:56::/64 qe0
prefix fec0:0:0:56::/64 qe0

if qe1 AdvSendAdvertisements 1
prefix 2:0:0:55::/64 qe1
prefix fec0:0:0:56::/64 qe1

if hme1 AdvSendAdvertisements 1
prefix  2002:8192:56bb:1::/64 qfe0

if hme1 AdvSendAdvertisements 1
prefix  2002:8192:56bb:2::/64 hme1
```

## IPv6 Interface Configuration File

IPv6 uses the `/etc/hostname6.`*interface* file at start up to automatically define IPv6 logical interfaces. When you select the IPv6 Enabled option during Solaris installation, the installation program creates an `/etc/hostname6.`*interface* file for the primary network interface, in addition to the `/etc/hostname.`*interface* file.

If more than one physical interface is detected during installation, you are prompted as to whether you want to configure these interfaces. The installation program creates IPv4 physical interface configuration files and IPv6 logical interface configuration files for each additional interface that you indicate.

As with IPv4 interfaces, you can also configure IPv6 interfaces manually, after Solaris installation. You create`/etc/hostname6.`*interface* files for the new interfaces. For instructions for manually configuring interfaces, refer to "Administering Interfaces in Solaris 10 3/05" on page 137 or Chapter 6, "Administering Network Interfaces (Tasks)."

The network interface configuration file names have the following syntax:

`hostname.`*interface*
`hostname6.`*interface*

The *interface* variable has the following syntax:

*dev*[.*module*[.*module* ...]]*PPA*

*dev* Indicates a network interface device. The device can be a physical network interface, such as `eri` or `qfe`, or a logical interface, such as a tunnel. See "IPv6 Interface Configuration File" on page 267 for more details.

*Module* Lists one or more `STREAMS` modules to be pushed onto the device when the device is plumbed.

*PPA* Indicates the physical point of attachment.

The syntax [.[.]] is also accepted.

**EXAMPLE 11–2** IPv6 Interface Configuration Files

The following are examples of valid IPv6 configuration file names:

```
hostname6.qfe0
hostname.ip.tun0
hostname.ip6.tun0
hostname6.ip6to4tun0
hostname6.ip.tun0
hostname6.ip6.tun0
```

### `/etc/inet/ipaddrsel.conf` Configuration File

The `/etc/inet/ipaddrsel.conf` file contains the IPv6 default address selection policy table. When you install the Solaris OS with IPv6 enabled, this file contains the contents that are shown in Table 11–5.

You can edit the contents of `/etc/inet/ipaddrsel.conf`. However, in most cases, you should refrain from modifying this file. If modification is necessary, refer to the procedure "How to Administer the IPv6 Address Selection Policy Table" on page 226. For more information on `ippaddrsel.conf`, refer to "Reasons for Modifying the IPv6 Address Selection Policy Table" on page 269 and the `ipaddrsel.conf`(4) man page.

## IPv6-Related Commands

This section describes commands that are added with the Solaris IPv6 implementation. The text also describes modifications to existing commands to support IPv6.

### `ipaddrsel` Command

The `ipaddrsel` command enables you to modify the IPv6 default address selection policy table.

The Solaris kernel uses the IPv6 default address selection policy table to perform destination address ordering and source address selection for an IPv6 packet header. The `/etc/inet/ipaddrsel.conf` file contains the policy table.

The following table lists the default address formats and their priorities for the policy table. You can find technical details for IPv6 address selection in the `inet6`(7P) man page.

**TABLE 11–5** IPv6 Address Selection Policy Table

| Prefix | Precedence | Definition |
|---|---|---|
| `::1/128` | 50 | Loopback |
| `::/0` | 40 | Default |
| `2002::/16` | 30 | 6to4 |
| `::/96` | 20 | IPv4 Compatible |
| `::ffff:0:0/96` | 10 | IPv4 |

In this table, IPv6 prefixes (`::1/128` and `::/0`) take precedence over 6to4 addresses (`2002::/16`) and IPv4 addresses (`::/96` and `::ffff:0:0/96`). Therefore, by default, the kernel selects the global IPv6 address of the interface for packets going to another IPv6 destination. The IPv4 address of the interface has a lower priority, particularly for packets going to an IPv6 destination. Given the selected IPv6 source address, the kernel also uses the IPv6 format for the destination address.

## Reasons for Modifying the IPv6 Address Selection Policy Table

Under most instances, you do not need to change the IPv6 default address selection policy table. If you do need to administer the policy table, you use the `ipaddrsel` command.

You might want to modify the policy table under the following circumstances:

- If the system has an interface that is used for a 6to4 tunnel, you can give higher priority to 6to4 addresses.
- If you want a particular source address to be used only in communications with a particular destination address, you can add these addresses to the policy table. Then, you can use `ifconfig` to flag these addresses as preferred.
- If you want IPv4 addresses to take precedence over IPv6 addresses, you can change the priority of `::ffff:0:0/96` to a higher number.
- If you need to assign a higher priority to deprecated addresses, you can add the deprecated address to the policy table. For example, site-local addresses are now deprecated in IPv6. These addresses have the prefix `fec0::/10`. You can change the policy table to give higher priority to site-local addresses.

For details about the `ipaddrsel` command, refer to the `ipaddrsel`(1M) man page.

## `6to4relay` Command

*6to4 tunneling* enables communication between isolated 6to4 sites. However, to transfer packets with a native, non-6to4 IPv6 site, the 6to4 router must establish a tunnel with a 6to4 relay router. The *6to4 relay router* then forwards the 6to4 packets to the IPv6 network and ultimately, to the native IPv6 site. If your 6to4-enabled site must exchange data with a native IPv6 site, you use the `6to4relay` command to enable the appropriate tunnel.

Because the use of relay routers is insecure, tunneling to a relay router is disabled by default in the Solaris OS. Carefully consider the issues that are involved in creating a tunnel to a 6to4 relay router before deploying this scenario. For detailed information on 6to4 relay routers, refer to "Considerations for Tunnels to a 6to4 Relay Router" on page 291. If you decide to enable 6to4 relay router support, you can find the related procedures in "How to Configure a 6to4 Tunnel" on page 192.

### Syntax of `6to4relay`

The `6to4relay` command has the following syntax:

```
6to4relay -e [-a IPv4-address] -d -h
```

| | |
|---|---|
| `-e` | Enables support for tunnels between the 6to4 router and an anycast 6to4 relay router. The tunnel endpoint address is then set to `192.88.99.1`, the default address for the anycast group of 6to4 relay routers. |
| `-a` *IPv4-address* | Enables support for tunnels between the 6to4 router and a 6to4 relay router with the specified *IPv4-address*. |
| `-d` | Disables support for tunneling to the 6to4 relay router, the default for the Solaris OS. |
| `-h` | Displays help for `6to4relay`. |

For more information, refer to the `6to4relay`(1M) man page.

**EXAMPLE 11–3** Default Status Display of 6to4 Relay Router Support

The `6to4relay` command, without arguments, shows the current status of 6to4 relay router support. This example shows the default for the Solaris OS implementation of IPv6.

```
# /usr/sbin/6to4relay
6to4relay:6to4 Relay Router communication support is disabled
```

**EXAMPLE 11–4** Status Display With 6to4 Relay Router Support Enabled

If relay router support is enabled, `6to4relay` displays the following output:

**EXAMPLE 11–4** Status Display With 6to4 Relay Router Support Enabled *(Continued)*

```
# /usr/sbin/6to4relay
6to4relay:6to4 Relay Router communication support is enabled
IPv4 destination address of Relay Router=192.88.99.1
```

**EXAMPLE 11–5** Status Display With a 6to4 Relay Router Specified

If you specify the `-a` option and an IPv4 address to the `6to4relay` command, the IPv4 address that you give with `-a` is displayed instead of `192.88.99.1`.

`6to4relay` does not report successful execution of the `-d`, `-e`, and `-a` *IPv4 address* options. However, `6to4relay` does display any error messages that might be generated when you run these options.

## `ifconfig` Command Extensions for IPv6 Support

The `ifconfig` command enables IPv6 interfaces and the tunneling module to be plumbed. `ifconfig` uses an extended set of ioctls to configure both IPv4 and IPv6 network interfaces. The following describes `ifconfig` options that support IPv6 operations. See "Monitoring the Interface Configuration With the `ifconfig` Command" on page 204 for a range of both IPv4 and IPv6 tasks that involve `ifconfig`.

| | |
|---|---|
| `index` | Sets the interface index. |
| `tsrc/tdst` | Sets the tunnel source or destination. |
| `addif` | Creates the next available logical interface. |
| `removeif` | Deletes a logical interface with a specific IP address. |
| `destination` | Sets the point-to-point destination address for an interface. |
| `set` | Sets an address, netmask, or both for an interface. |
| `subnet` | Sets the subnet address of an interface. |
| `xmit/-xmit` | Enables or disables packet transmission on an interface. |

Chapter 7, "Configuring an IPv6 Network (Tasks)," provides IPv6 configuration procedures.

**EXAMPLE 11–6** Adding a Logical IPv6 Interface With the `-addif` Option of the `ifconfig` Command

The following form of the `ifconfig` command creates the `hme0:3` logical interface:

```
# ifconfig hme0 inet6 addif up
Created new logical interface hme0:3
```

This form of `ifconfig` verifies the creation of the new interface:

**EXAMPLE 11–6** Adding a Logical IPv6 Interface With the `-addif` Option of the `ifconfig` Command *(Continued)*

```
# ifconfig hme0:3 inet6
hme0:3: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        inet6  inet6 fe80::203:baff:fe11:b321/10
```

**EXAMPLE 11–7** Removing a Logical IPv6 Interface With the `-removeif` Option of the `ifconfig` Command

The following form of the `ifconfig` command removes the `hme0:3` logical interface.

```
# ifconfig hme0:3 inet6 down

# ifconfig hme0 inet6 removeif 1234::5678
```

**EXAMPLE 11–8** Using `ifconfig` to Configure an IPv6 Tunnel Source

```
# ifconfig ip.tun0 inet6 plumb index 13
```

Opens the tunnel to be associated with the physical interface name.

```
# ifconfig ip.tun0 inet6
ip.tun0: flags=2200850<POINTOPOINT,RUNNING,MULTICAST,NONUD,
#IPv6> mtu 1480 index 13
        inet tunnel src 0.0.0.0
        inet6 fe80::/10 --> ::
```

Configures the streams that are needed for TCP/IP to use the tunnel device and report the status of the device.

```
# ifconfig ip.tun0 inet6 tsrc 120.46.86.158 tdst 120.46.86.122
```

Configures the source and the destination address for the tunnel.

```
# ifconfig ip.tun0 inet6
ip.tun0: flags=2200850<POINTOPOINT,RUNNING,MULTICAST,NONUD,
IPv6> mtu 1480 index 13
        inet tunnel src 120.46.86.158  tunnel dst 120.46.86.122
        inet6 fe80::8192:569e/10 --> fe80::8192:567a
```

Reports the new status of the device after the configuration.

**EXAMPLE 11–9** Configuring a 6to4 Tunnel Through `ifconfig` (Long Form)

This example of a 6to4 pseudo-interface configuration uses the subnet ID of 1 and specifies the host ID, in hexadecimal form.

```
# ifconfig ip.6to4tun0 inet6 plumb
# ifconfig ip.6to4tun0 inet tsrc 129.146.86.187 \
2002:8192:56bb:1::8192:56bb/64 up

# ifconfig ip.6to4tun0 inet6
ip.6to4tun0: flags=2200041<UP,RUNNING,NONUD,IPv6>mtu 1480 index 11
        inet tunnel src 129.146.86.187
        tunnel hop limit 60
        inet6 2002:8192:56bb:1::8192:56bb/64
```

**EXAMPLE 11–10** Configuring a 6to4 Tunnel Through `ifconfig` (Short Form)

This example shows the short form for configuring a 6to4 tunnel.

```
# ifconfig ip.6to4tun0 inet6 plumb
# ifconfig ip.6to4tun0 inet tsrc 129.146.86.187 up

# ifconfig ip.6to4tun0 inet6
ip.6to4tun0: flags=2200041<UP,RUNNING,NONUD,IPv6>mtu 1480 index 11
        inet tunnel src 129.146.86.187
        tunnel hop limit 60
        inet6 2002:8192:56bb::1/64
```

## `netstat` Command Modifications for IPv6 Support

The `netstat` command displays both IPv4 and IPv6 network status. You can choose which protocol information to display by setting the `DEFAULT_IP` value in the `/etc/default/inet_type` file or by using the `-f` command-line option. With a permanent setting of `DEFAULT_IP`, you can ensure that `netstat` displays only IPv4 information. You can override this setting by using the `-f` option. For more information on the `inet_type` file, see the `inet_type`(4) man page.

The `-p` option of the `netstat` command displays the net-to-media table, which is the ARP table for IPv4 and the neighbor cache for IPv6. See the `netstat`(1M) man page for details. See "How to Display the Status of Sockets" on page 212 for descriptions of procedures that use this command.

### `snoop` Command Modifications for IPv6 Support

The `snoop` command can capture both IPv4 and IPv6 packets. This command can display IPv6 headers, IPv6 extension headers, ICMPv6 headers, and Neighbor Discovery protocol data. By default, the `snoop` command displays both IPv4 and IPv6 packets. If you specify the `ip` or `ip6` protocol keyword, the `snoop` command displays only IPv4 or IPv6 packets. The IPv6 filter option enables you to filter through all packets, both IPv4 and IPv6, displaying only the IPv6 packets. See the `snoop`(1M) man page for details. See "How to Monitor IPv6 Network Traffic" on page 225 for procedures that use the `snoop` command.

### `route` Command Modifications for IPv6 Support

The `route` command operates on both IPv4 and IPv6 routes, with IPv4 routes as the default. If you use the `-inet6` option on the command line immediately after the `route` command, operations are performed on IPv6 routes. See the `route`(1M) man page for details.

### `ping` Command Modifications for IPv6 Support

The `ping` command can use both IPv4 and IPv6 protocols to probe target hosts. Protocol selection depends on the addresses that are returned by the name server for the specific target host. By default, if the name server returns an IPv6 address for the target host, the `ping` command uses the IPv6 protocol. If the server returns only an IPv4 address, the `ping` command uses the IPv4 protocol. You can override this action by using the `-A` command-line option to specify which protocol to use.

For detailed information, see the `ping`(1M) man page. For procedures that use `ping`, refer to "Probing Remote Hosts With the `ping` Command" on page 216.

### `traceroute` Command Modifications for IPv6 Support

You can use the `traceroute` command to trace both the IPv4 and IPv6 routes to a specific host. From a protocol perspective, `traceroute` uses the same algorithm as `ping`. Use the `-A` command-line option to override this selection. You can trace each individual route to every address of a multihomed host by using the `-a` command-line option.

For detailed information, see the `traceroute`(1M) man page. For procedures that use `traceroute`, refer to "Displaying Routing Information With the `traceroute` Command" on page 220.

## IPv6-Related Daemons

This section discusses the IPv6-related daemons.

## `in.ndpd` Daemon, for Neighbor Discovery

The`in.ndpd` daemon implements the IPv6 Neighbor Discovery protocol and router discovery. The daemon also implements address autoconfiguration for IPv6. The following shows the supported options of `in.ndpd`.

`-d` Turns on debugging.

`-D` Turns on debugging for specific events.

`-f` Specifies a file to read configuration data from, instead of the default `/etc/inet/ndpd.conf` file.

`-I` Prints related information for each interface.

`-n` Does not loop back router advertisements.

`-r` Ignores received packets.

`-v` Specifies verbose mode, reporting various types of diagnostic messages.

`-t` Turns on packet tracing.

The `in.ndpd` daemon is controlled by parameters that are set in the `/etc/inet/ndpd.conf` configuration file and any applicable parameters in the `/var/inet/ndpd_state.`*interface* startup file.

When the `/etc/inet/ndpd.conf` file exists, the file is parsed and used to configure a node as a router. Table 11–2 lists the valid keywords that might appear in this file. When a host is booted, routers might not be immediately available. Advertised packets by the router might be dropped. Also, advertised packets might not reach the host.

The `/var/inet/ndpd_state.`*interface* file is a state file. This file is updated periodically by each node. When the node fails and is restarted, the node can configure its interfaces in the absence of routers. This file contains the interface address, the last time that the file was updated, and how long the file is valid. This file also contains other parameters that are "learned" from previous router advertisements.

---

**Note –** You do not need to alter the contents of state files. The `in.ndpd` daemon automatically maintains state files.

---

See the `in.ndpd`(1M) man page and the `ndpd.conf`(4) man page for lists of configuration variables and allowable values.

## `in.ripngd` Daemon, for IPv6 Routing

The `in.ripngd` daemon implements the Routing Information Protocol next-generation for IPv6 routers (RIPng). RIPng defines the IPv6 equivalent of RIP. When you configure an IPv6 router with the `routeadm` command and turn on IPv6 routing, the `in.ripngd` daemon implements RIPng on the router.

The following shows the supported options of RIPng.

`-p` *n* — *n* specifies the alternate port number that is used to send or receive RIPnG packets.

`-q` — Suppresses routing information.

`-s` — Forces routing information even if the daemon is acting as a router.

`-P` — Suppresses use of poison reverse.

`-S` — If `in.ripngd` does not act as a router, the daemon enters only a default route for each router.

## `inetd` Daemon and IPv6 Services

An IPv6-enabled server application can handle both IPv4 requests and IPv6 requests, or IPv6 requests only. The server always handles requests through an IPv6 socket. Additionally, the server uses the same protocol that the corresponding client uses. To add or modify a service for IPv6, use the commands available from the Service Management Facility (SMF).

- For information about the SMF commands, refer to "SMF Command-Line Administrative Utilities" in *System Administration Guide: Basic Administration*.
- For an example task that uses SMF to configure an IPv4 service manifest that runs over SCTP, refer to "How to Add Services That Use the SCTP Protocol" on page 133.

To configure an IPv6 service, you must ensure that the `proto` field value in the `inetadm` profile for that service lists the appropriate value:

- For a service that handles both IPv4 and IPv6 requests, choose `tcp6`, `udp6`, or `sctp`. A `proto` value of `tcp6`, `udp6`, or `sctp6` causes `inetd` to pass on an IPv6 socket to the server. The server contains an IPv4-mapped address in case a IPv4 client has a request.
- For a service that handles only IPv6 requests, choose `tcp6only` or `udp6only`. With either of these values for `proto`, `inetd` passes the server an IPv6 socket.

If you replace a Solaris command with another implementation, you must verify that the implementation of that service supports IPv6. If the implementation does not support IPv6, then you must specify the `proto` value as either `tcp`, `udp`, or `sctp`.

Here is a profile that results from running `inetadm` for an `echo` service manifest that supports both IPv4 and IPv6 and runs over SCTP:

```
# inetadm -l svc:/network/echo:sctp_stream
    SCOPE    NAME=VALUE      name="echo"
             endpoint_type="stream"
             proto="sctp6"
             isrpc=FALSE
             wait=FALSE
             exec="/usr/lib/inet/in.echod -s"
             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
    default  tcp_trace=FALSE
    default  tcp_wrappers=FALSE
```

To change the value of the `proto` field, use the following syntax:

```
# inetadm -m FMRI proto="transport-protocols"
```

All servers that are provided with Solaris software require only one profile entry that specifies `proto` as `tcp6`, `udp6`, or `sctp6`. However, the remote shell server (`shell`) and the remote execution server (`exec`) now are composed of a single service instance, which requires a `proto` value containing both the `tcp` and `tcp6only` values. For example, to set the `proto` value for `shell`, you would issue the following command:

```
# inetadm -m network/shell:default proto="tcp,tcp6only"
```

See IPv6 extensions to the Socket API in *Programming Interfaces Guide* for more details on writing IPv6-enabled servers that use sockets.

## Considerations When Configuring a Service for IPv6

When you add or modify a service for IPv6, keep in mind the following caveats:

- You need to specify the `proto` value as `tcp6`, `sctp6`, or `udp6` to enable both IPv4 or IPv6 connections. If you specify the value for `proto` as `tcp`, `sctp`, or `udp`, the service uses only IPv4.
- Though you can add a service instance that uses one-to-many style SCTP sockets for `inetd`, this is not recommended. `inetd` does not work with one-to-many style SCTP sockets.
- If a service requires two entries because its `wait-status` or `exec` properties differ, then you must create two instances/services from the original service.

# IPv6 Neighbor Discovery Protocol

IPv6 introduces the Neighbor Discovery protocol, as described in RFC 2461, Neighbor Discovery for IP Version 6 (IPv6) (`http://www.ietf.org/rfc/rfc2461.txt?number=2461`). For an overview of major Neighbor Discovery features, refer to "IPv6 Neighbor Discovery Protocol Overview" on page 79.

This section discusses the following features of the Neighbor Discovery protocol:


- "ICMP Messages From Neighbor Discovery" on page 278
- "Autoconfiguration Process" on page 278
- "Neighbor Solicitation and Unreachability" on page 280
- "Duplicate Address Detection Algorithm" on page 281
- "Comparison of Neighbor Discovery to ARP and Related IPv4 Protocols" on page 282


## ICMP Messages From Neighbor Discovery

Neighbor Discovery defines five new Internet Control Message Protocol (ICMP) messages. The messages serve the following purposes:

- **Router solicitation** – When an interface becomes enabled, hosts can send router solicitation messages. The solicitations request routers to generate router advertisements immediately, rather than at their next scheduled time.
- **Router advertisement** – Routers advertise their presence, various link parameters, and various Internet parameters. Routers advertise either periodically, or in response to a router solicitation message. Router advertisements contain prefixes that are used for on-link determination or address configuration, a suggested hop-limit value, and so on.
- **Neighbor solicitation** – Nodes send neighbor solicitation messages to determine the link-layer address of a neighbor. Neighbor solicitation messages are also sent to verify that a neighbor is still reachable by a cached link-layer address. Neighbor solicitations are also used for duplicate address detection.
- **Neighbor advertisement** – A node sends neighbor advertisement messages in response to a neighbor solicitation message. The node can also send unsolicited neighbor advertisements to announce a link-layer address change.
- **Redirect** – Routers use redirect messages to inform hosts of a better first hop for a destination, or that the destination is on the same link.

## Autoconfiguration Process

This section provides an overview of the typical steps that are performed by an interface during autoconfiguration. Autoconfiguration is performed only on multicast-capable links.

1. A multicast-capable interface is enabled, for example, during system startup of a node.
2. The node begins the autoconfiguration process by generating a link-local address for the interface.

   The link-local address is formed from the Media Access Control (MAC) address of the interface.
3. The node sends a neighbor solicitation message that contains the tentative link-local address as the target.

   The purpose of the message is to verify that the prospective address is not already in use by another node on the link. After verification, the link-local address can be assigned to an interface.

   a. If another node already uses the proposed address, that node returns a neighbor advertisement stating that the address is already in use.

   b. If another node is also attempting to use the same address, the node also sends a neighbor solicitation for the target.

      The number of neighbor solicitation transmissions or retransmissions, and the delay between consecutive solicitations, are link specific. You can set these parameters, if necessary.
4. If a node determines that its prospective link-local address is not unique, autoconfiguration stops. At that point, you must manually configure the link-local address of the interface.

   To simplify recovery, you can supply an alternate interface ID that overrides the default identifier. Then, the autoconfiguration mechanism can resume by using the new, presumably unique, interface ID.
5. When a node determines that its prospective link-local address is unique, the node assigns the address to the interface.

   At this point, the node has IP-level connectivity with neighboring nodes. The remaining autoconfiguration steps are performed only by hosts.

## Obtaining a Router Advertisement

The next phase of autoconfiguration involves obtaining a router advertisement or determining that no routers are present. If routers are present, the routers send router advertisements that specify what type of autoconfiguration a host should perform.

Routers send router advertisements periodically. However, the delay between successive advertisements is generally longer than a host that performs autoconfiguration can wait. To quickly obtain an advertisement, a host sends one or more router solicitations to the all-routers multicast group.

### Prefix Configuration Variables

Router advertisements also contain prefix variables with information that stateless address autoconfiguration uses to generate prefixes. The Stateless Address Autoconfiguration field in router advertisements are processed independently. One option field that contains prefix information, the Address Autoconfiguration flag, indicates whether the option even applies to stateless autoconfiguration. If the option field does apply, additional option fields contain a subnet prefix with lifetime values. These values indicate the length of time that addresses created from the prefix remain preferred and valid.

Because routers periodically generate router advertisements, hosts continually receive new advertisements. IPv6-enabled hosts process the information that is contained in each advertisement. Hosts add to the information. They also refresh the information that is received in previous advertisements.

### Address Uniqueness

For security reasons, all addresses must be tested for uniqueness prior to their assignment to an interface. The situation is different for addresses that are created through stateless autoconfiguration. The uniqueness of an address is determined primarily by the portion of the address that is formed from an interface ID. Thus, if a node has already verified the uniqueness of a link-local address, additional addresses need not be tested individually. The addresses must be created from the same interface ID. In contrast, all addresses that are obtained manually should be tested individually for uniqueness. System administrators at some sites believe that the overhead of performing duplicate address detection outweighs its benefits. For these sites, the use of duplicate address detection can be disabled by setting a per-interface configuration flag.

To accelerate the autoconfiguration process, a host can generate its link-local address, and verify its uniqueness, while the host waits for a router advertisement. A router might delay a response to a router solicitation for a few seconds. Consequently, the total time necessary to complete autoconfiguration can be significantly longer if the two steps are done serially.

## Neighbor Solicitation and Unreachability

Neighbor Discovery uses *neighbor solicitation* messages to determine if more than one node is assigned the same unicast address. *Neighbor unreachability detection* detects the failure of a neighbor or the failure of the forward path to the neighbor. This detection requires positive confirmation that packets that are sent to a neighbor are actually reaching that neighbor. Neighbor unreachability detection also determines that packets are being processed properly by the node's IP layer.

Neighbor unreachability detection uses confirmation from two sources: upper-layer protocols and neighbor solicitation messages. When possible, upper-layer protocols provide a positive

confirmation that a connection is making *forward progress*. For example, when new TCP acknowledgments are received, it is confirmed that previously sent data has been delivered correctly.

When a node does not get positive confirmation from upper-layer protocols, the node sends unicast neighbor solicitation messages. These messages solicit neighbor advertisements as reachability confirmation from the next hop. To reduce unnecessary network traffic, probe messages are sent only to neighbors to which the node is actively sending packets.

## Duplicate Address Detection Algorithm

To ensure that all configured addresses are likely to be unique on a particular link, nodes run a *duplicate address detection* algorithm on addresses. The nodes must run the algorithm before assigning the addresses to an interface. The duplicate address detection algorithm is performed on all addresses.

The autoconfiguration process that is described in this section applies only to hosts, and not routers. Because host autoconfiguration uses information that is advertised by routers, routers need to be configured by some other means. However, routers generate link-local addresses by using the mechanism that is described in this chapter. In addition, routers are expected to successfully pass the duplicate address detection algorithm on all addresses prior to assigning the address to an interface.

## Proxy Advertisements

A router that accepts packets on behalf of a target address can issue non-override neighbor advertisements. The router can accept packets for a target address that is unable to respond to neighbor solicitations. Currently, the use of proxy is not specified. However, proxy advertising can potentially be used to handle cases such as mobile nodes that have moved off-link. Note that the use of proxy is not intended as a general mechanism to handle nodes that do not implement this protocol.

## Inbound Load Balancing

Nodes with replicated interfaces might need to load balance the reception of incoming packets across multiple network interfaces on the same link. Such nodes have multiple link-local addresses assigned to the same interface. For example, a single network driver can represent multiple network interface cards as a single logical interface that has multiple link-local addresses.

Load balancing is handled by allowing routers to omit the source link-local address from router advertisement packets. Consequently, neighbors must use neighbor solicitation messages to learn link-local addresses of routers. Returned neighbor advertisement messages can then contain link-local addresses that differ, depending on which issued the solicitation.

## Link-Local Address Change

A node that knows its link-local address has been changed can send out multicast unsolicited, neighbor advertisement packets. The node can send multicast packets to all nodes to update cached link-local addresses that have become invalid. The sending of unsolicited advertisements is a performance enhancement only. The detection algorithm for neighbor unreachability ensures that all nodes reliably discover the new address, though the delay might be somewhat longer.

## Comparison of Neighbor Discovery to ARP and Related IPv4 Protocols

The functionality of the IPv6 Neighbor Discovery protocol corresponds to a combination of the IPv4 protocols: Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP) Router Discovery, and ICMP Redirect. IPv4 does not have a generally agreed on protocol or mechanism for neighbor unreachability detection. However, host requirements do specify some possible algorithms for dead gateway detection. Dead gateway detection is a subset of the problems that neighbor unreachability detection solves.

The following list compares the Neighbor Discovery protocol to the related set of IPv4 protocols.

- Router discovery is part of the base IPv6 protocol set. IPv6 hosts do not need to `snoop` the routing protocols to find a router. IPv4 uses ARP, ICMP router discovery, and ICMP redirect for router discovery.
- IPv6 router advertisements carry link-local addresses. No additional packet exchange is needed to resolve the router's link-local address.
- Router advertisements carry site prefixes for a link. A separate mechanism is not needed to configure the netmask, as is the case with IPv4.
- Router advertisements enable address autoconfiguration. Autoconfiguration is not implemented in IPv4.
- Neighbor Discovery enables IPv6 routers to advertise an MTU for hosts to use on the link. Consequently, all nodes use the same MTU value on links that lack a well-defined MTU. IPv4 hosts on the same network might have different MTUs.

- Unlike IPv4 broadcast addresses, IPv6 address resolution multicasts are spread over 4 billion (2^32) multicast addresses, greatly reducing address resolution-related interrupts on nodes other than the target. Moreover, non-IPv6 machines should not be interrupted at all.
- IPv6 redirects contain the link-local address of the new first hop. Separate address resolution is not needed on receiving a redirect.
- Multiple site prefixes can be associated with the same IPv6 network. By default, hosts learn all local site prefixes from router advertisements. However, routers can be configured to omit some or all prefixes from router advertisements. In such instances, hosts assume that destinations are on remote networks. Consequently, hosts send the traffic to routers. A router can then issue redirects, as appropriate.
- Unlike IPv4, the recipient of an IPv6 redirect message assumes that the new next-hop is on the local network. In IPv4, a host ignores redirect messages that specify a next-hop that is not on the local network, according to the network mask. The IPv6 redirect mechanism is analogous to the XRedirect facility in IPv4. The redirect mechanism is useful on non-broadcast and shared media links. On these networks, nodes should not check for all prefixes for local link destinations.
- IPv6 neighbor unreachability detection improves packet delivery in the presence of failing routers. This capability improves packet delivery over partially failing or partitioned links. This capability also improves packet delivery over nodes that change their link-local addresses. For example, mobile nodes can move off the local network without losing any connectivity because of stale ARP caches. IPv4 has no corresponding method for neighbor unreachability detection.
- Unlike ARP, Neighbor Discovery detects half-link failures by using neighbor unreachability detection. Neighbor Discovery avoids sending traffic to neighbors when two-way connectivity is absent.
- By using link-local addresses to uniquely identify routers, IPv6 hosts can maintain the router associations. The ability to identify routers is required for router advertisements and for redirect messages. Hosts need to maintain router associations if the site uses new global prefixes. IPv4 does not have a comparable method for identifying routers.
- Because Neighbor Discovery messages have a hop limit of 255 upon receipt, the protocol is immune to spoofing attacks originating from off-link nodes. In contrast, IPv4 off-link nodes can send ICMP redirect messages. IPv4 off-link nodes can also send router advertisement messages.
- By placing address resolution at the ICMP layer, Neighbor Discovery becomes more media independent than ARP. Consequently, standard IP authentication and security mechanisms can be used.

# IPv6 Routing

Routing in IPv6 is almost identical to IPv4 routing under Classless Inter-Domain Routing (CIDR). The only difference is that the addresses are 128-bit IPv6 addresses instead of 32-bit IPv4 addresses. With very straightforward extensions, all of IPv4's routing algorithms, such as OSPF, RIP, IDRP, and IS-IS, can be used to route IPv6.

IPv6 also includes simple routing extensions that support powerful new routing capabilities. The following list describes the new routing capabilities:

- Provider selection that is based on policy, performance, cost, and so on
- Host mobility, route to current location
- Auto-readdressing, route to new address

You obtain the new routing capabilities by creating sequences of IPv6 addresses that use the IPv6 routing option. An IPv6 source uses the routing option to list one or more intermediate nodes, or topological group, to be visited on the way to a packet's destination. This function is very similar in function to IPv4's loose source and record route option.

To make address sequences a general function, IPv6 hosts are required, in most instances, to reverse routes in a packet that a host receives. The packet must be successfully authenticated by using the IPv6 authentication header. The packet must contain address sequences in order to return the packet to its originator. This technique forces IPv6 host implementations to support the handling and reversal of source routes. The handling and reversal of source routes is the key that enables providers to work with hosts that implement the new IPv6 capabilities such as provider selection and extended addresses.

## Router Advertisement

On multicast-capable links and point-to-point links, each router periodically sends to the multicast group a router advertisement packet that announces its availability. A host receives router advertisements from all routers, building a list of default routers. Routers generate router advertisements frequently enough so that hosts learn of their presence within a few minutes. However, routers do not advertise frequently enough to rely on an absence of advertisements to detect router failure. A separate detection algorithm that determines neighbor unreachability provides failure detection.

### Router Advertisement Prefixes

Router advertisements contain a list of subnet prefixes that is used to determine if a host is on the same link (on-link) as the router. The list of prefixes is also used for autonomous address configuration. Flags that are associated with the prefixes specify the intended uses of a particular prefix. Hosts use the advertised on-link prefixes to build and maintain a list that is used to decide when a packet's destination is on-link or beyond a router. A destination can be on-link

even though the destination is not covered by any advertised on-link prefix. In such instances, a router can send a redirect. The redirect informs the sender that the destination is a neighbor.

Router advertisements, and per-prefix flags, enable routers to inform hosts how to perform stateless address autoconfiguration.

### Router Advertisement Messages

Router advertisement messages also contain Internet parameters, such as the hop limit, that hosts should use in outgoing packets. Optionally, router advertisement messages also contain link parameters, such as the link MTU. This feature enables the centralized administration of critical parameters. The parameters can be set on routers and automatically propagated to all hosts that are attached.

Nodes accomplish address resolution by sending to the multicast group a neighbor solicitation that asks the target node to return its link-layer address. Multicast neighbor solicitation messages are sent to the solicited-node multicast address of the target address. The target returns its link-layer address in a unicast neighbor advertisement message. A single request-response pair of packets is sufficient for both the initiator and the target to resolve each other's link-layer addresses. The initiator includes its link-layer address in the neighbor solicitation.

## IPv6 Tunnels

To minimize any dependencies at a dual-stack, IPv4/IPv6 site, all the routers in the path between two IPv6 nodes do not need to support IPv6. The mechanism that supports such a network configuration is called *tunneling*. Basically, IPv6 packets are placed inside IPv4 packets, which are then routed through the IPv4 routers. The following figure illustrates the tunneling mechanism through IPv4 routers, which are indicated in the figure by "R."

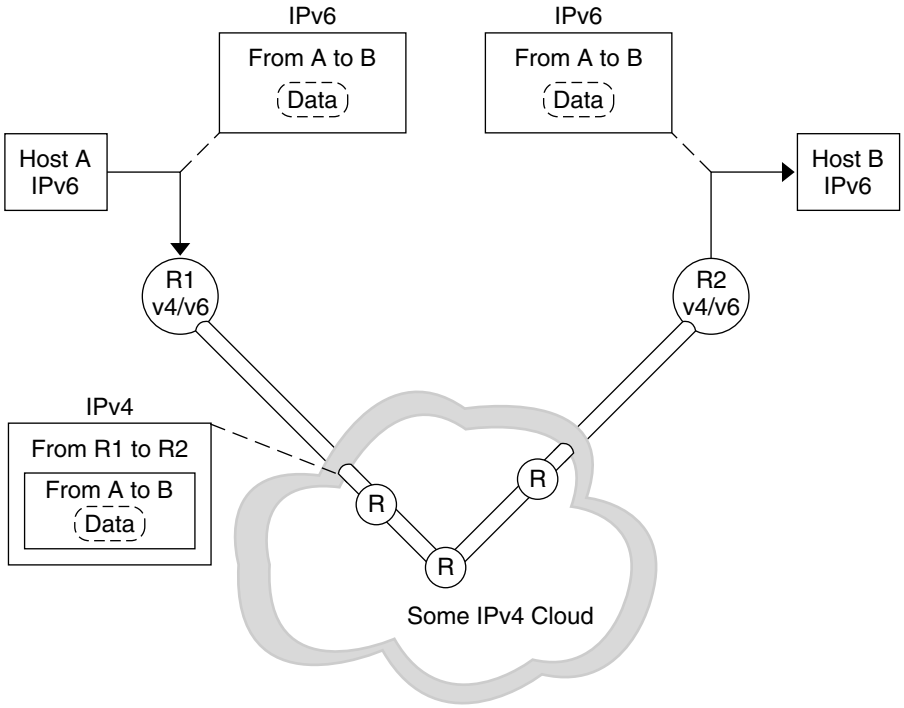



**FIGURE 11–5** IPv6 Tunneling Mechanism

The Solaris IPv6 implementation includes two types of tunneling mechanisms:

- Configured tunnels between two routers, as in Figure 11–5
- Automatic tunnels that terminate at the endpoint hosts

A configured tunnel is currently used on the Internet for other purposes, for example, on the MBONE, the IPv4 multicast backbone. Operationally, the tunnel consists of two routers that are configured to have a virtual point-to-point link between the two routers over the IPv4 network. This kind of tunnel is likely to be used on some parts of the Internet for the foreseeable future.

Automatic tunnels require IPv4-compatible addresses. Automatic tunnels can be used to connect IPv6 nodes when IPv6 routers are not available. These tunnels can originate either on a dual-stack host or on a dual-stack router by configuring an automatic tunneling network interface. The tunnels always terminate on the dual-stack host. These tunnels work by dynamically determining the destination IPv4 address, which is the endpoint of the tunnel, by extracting the address from the IPv4-compatible destination address.

# Configured Tunnels

Tunneling interfaces have the following format:

`ip.tun` *ppa*

*ppa* is the physical point of attachment.

At system startup, the tunneling module (`tun`) is pushed, by the `ifconfig` command, on top of IP to create a virtual interface. The push is accomplished by creating the appropriate `hostname6.*` file.

For example, to create a tunnel to encapsulate IPv6 packets over an IPv4 network, IPv6 over IPv4, you would create the following file name:

`/etc/hostname6.ip.tun0`

The content of this file is passed to `ifconfig` after the interfaces have been plumbed. The content becomes the parameters that are necessary to configure a point-to-point tunnel.

**EXAMPLE 11–11** `hostname6.ip.tun0` File for an IPv6 Over IPv4 Tunnel

The following is an example of entries in the `hostname6.ip.tun0` file:

```
tsrc 10.10.10.23 tdst 172.16.7.19 up
addif 2001:db8:3b4c:1:5678:5678::2 up
```

In this example, the IPv4 source and destination addresses are used as tokens to autoconfigure IPv6 link-local addresses. These addresses are the source and destination for the `ip.tun0` interface. Two interfaces are configured. The `ip.tun0` interface is configured. A logical interface, `ip.tun0:1`, is also configured. The logical interface has the source and destination IPv6 addresses specified by the `addif` command.

The contents of these configuration files are passed to `ifconfig` without change when the system is started in multiuser mode. The entries in Example 11–11 are equivalent to the following:

```
# ifconfig ip.tun0 inet6 plumb
# ifconfig ip.tun0 inet6 tsrc 10.0.0.23 tdst 172.16.7.19 up
# ifconfig ip.tun0 inet6 addif 2001:db8:3b4c:1:5678:5678::2 up
```

The following shows the output of `ifconfig -a` for this tunnel.

```
ip.tun0: flags=2200850<UP,POINTOPOINT,RUNNING,MULTICAST,
  NONUD,IPv6> mtu 1480 index 6
        inet tunnel src 10.0.0.23  tunnel dst 172.16.7.19
```

```
        inet6 fe80::c0a8:6417/10 --> fe80::c0a8:713
ip.tun0:1: flags=2200850<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480
  index 5
        inet6 2001:db8:3b4c:1:5678:5678::2
```

You can configure more logical interfaces by adding lines to the configuration file by using the following syntax:

```
addif IPv6-source IPv6-destination up
```

---

**Note –** When either end of the tunnel is an IPv6 router that advertises one or more prefixes over the tunnel, you do not need `addif` commands in the tunnel configuration files. Only `tsrc` and `tdst` might be required because all other addresses are autoconfigured.

---

In some situations, specific source and destination link-local addresses need to be manually configured for a particular tunnel. Change the first line of the configuration file to include these link-local addresses. The following line is an example:

```
tsrc 10.0.0.23 tdst 172.16.7.19 fe80::1/10 fe80::2 up
```

Notice that the source link-local address has a prefix length of 10. In this example, the `ip.tun0` interface resembles the following:

```
ip.tun0: flags=2200850<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,IPv6> mtu 1480
index 6
        inet tunnel src 10.0.0.23  tunnel dst 172.16.7.19
        inet6 fe80::1/10 --> fe80::2
```

To create a tunnel to encapsulate IPv6 packets over an IPv6 network, IPv6 over IPv6, you create the following file name:

```
/etc/hostname6.ip6.tun0
```

**EXAMPLE 11–12** `hostname6.ip6.tun0` File for an IPv6 over IPv6 Tunnel

The following is an example of entries in the `hostname6.ip6.tun0` file for IPv6 encapsulation over an IPv6 network:

```
tsrc 2001:db8:3b4c:114:a00:20ff:fe72:668c
        tdst 2001:db8:15fa:25:a00:20ff:fe9b:a1c3
fe80::4 fe80::61 up
```

To create a tunnel to encapsulate IPv4 packets over an IPv6 network, IPv4 over IPv6, you would create the following file name:

```
/etc/hostname.ip6.tun0
```

**EXAMPLE 11–13** `hostname.ip6.tun0` File for an IPv4 Over IPv6 Tunnel

The following is an example of entries in the `hostname.ip6.tun0` file for IPv4 encapsulation over an IPv6 network:

```
tsrc 2001:db8:3b4c:114:a00:20ff:fe72:668c
         tdst 2001:db8:15fa:25:a00:20ff:fe9b:a1c3
10.0.0.4 10.0.0.61 up
```

To create a tunnel to encapsulate IPv4 packets over an IPv4 network, IPv4 over IPv4, you would create the following file name:

```
/etc/hostname.ip.tun0
```

**EXAMPLE 11–14** `hostname.ip.tun0` for an IPv4 Over IPv4 Tunnel

The following is an example of entries in the `hostname.ip.tun0` file for IPv4 encapsulation over an IPv4 network:

```
tsrc 172.16.86.158 tdst 192.168.86.122
10.0.0.4 10.0.0.61 up
```

For specific information about `tun`, see the `tun`(7M) man page. .For a general description of tunneling concepts during the transition to IPv6, see "Overview of IPv6 Tunnels" on page 82. For a description of procedures for configuring tunnels, see "Tasks for Configuring Tunnels for IPv6 Support (Task Map)" on page 188.

## 6to4 Automatic Tunnels

The Solaris OS includes 6to4 tunnels as a preferred interim method for making the transition from IPv4 to IPv6 addressing. 6to4 tunnels enable isolated IPv6 sites to communicate across an automatic tunnel over an IPv4 network that does not support IPv6. To use 6to4 tunnels, you must configure a boundary router on your IPv6 network as one endpoint of the 6to4 automatic tunnel. Thereafter, the 6to4 router can participate in a tunnel to another 6to4 site, or, if required, to a native IPv6, non-6to4 site.

This section provides reference materials on the following 6to4 topics:

- Topology of a 6to4 tunnel
- 6to4 addressing, including the format of the advertisement
- Description of the packet flow across a 6to4 tunnel
- Topology of a tunnel between a 6to4 router and a 6to4 relay router

- Points to consider before you configure 6to4 relay router support

More information about 6to4 routing is available from the following sources.

| Task or Detail | For Information |
|---|---|
| Tasks for configuring a 6to4 tunnel | "How to Configure a 6to4 Tunnel" on page 192 |
| 6to4-related RFC | RFC 3056, "Connection of IPv6 Domains via IPv4 Clouds" (`http://www.ietf.org/rfc/rfc3056.txt`) |
| Detailed information about the `6to4relay` command, which enables support for tunnels to a 6to4 relay router | `6to4relay`(1M) |
| 6to4 security issues | Security Considerations for 6to4 (`http://www.ietf.org/rfc/rfc3964.txt`) |

## Topology of a 6to4 Tunnel

A 6to4 tunnel provides IPv6 connectivity to all 6to4 sites everywhere. Likewise, the tunnel also functions a link to all IPv6 sites, including the native IPv6 internet, provided that the tunnel is configured to forward to a relay router. The following figure shows how a 6to4 tunnel provides this connectivity between 6to4 sites.

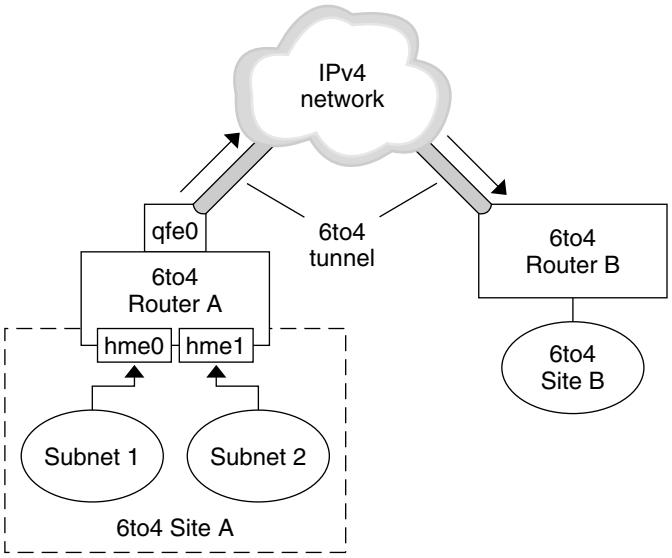



**FIGURE 11–6** Tunnel Between Two 6to4 Sites

The figure depicts two isolated 6to4 networks, Site A and Site B. Each site has configured a router with an external connection to an IPv4 network. A 6to4 tunnel across the IPv4 network provides a connection to link 6to4 sites.

Before an IPv6 site can become a 6to4 site, you must configure at least one router interface for 6to4 support. This interface must provide the external connection to the IPv4 network. The address that you configure on `qfe0` must be globally unique. In this figure, boundary Router A's interface `qfe0` connects Site A to the IPv4 network. Interface `qfe0` must already be configured with an IPv4 address before you can configure `qfe0` as a 6to4 pseudo-interface.

In the figure, 6to4 Site A is composed of two subnets, which are connected to interfaces `hme0` and `hme1` on Router A. All IPv6 hosts on either subnet of Site A automatically reconfigure with 6to4-derived addresses upon receipt of the advertisement from Router A.

Site B is another isolated 6to4 site. To correctly receive traffic from Site A, a boundary router on Site B must be configured for 6to4 support. Otherwise, packets that the router receives from Site A are not recognized and are then dropped.

## Packet Flow Through the 6to4 Tunnel

This section describes the flow of packets from a host at one 6to4 site to a host at a remote 6to4 site. This scenario uses the topology that is shown in Figure 11–6. Moreover, the scenario assumes that the 6to4 routers and the 6to4 hosts are already configured.

1. A host on Subnet 1 of 6to4 Site A sends a transmission, with a host at 6to4 Site B as the destination. Each packet header has a 6to4-derived source address and 6to4-derived destination address.
2. Site A's router encapsulates each 6to4 packet within an IPv4 header. In this process, the router sets the IPv4 destination address of the encapsulating header to Site B's router address. For each IPv6 packet that flows through the tunnel interface, the packet's IPv6 destination address also contains the IPv4 destination address. Thus, the router is able to determine the IPv4 destination address that is set on the encapsulating header. Then, the router uses standard IPv4 routing procedures to forward the packet over the IPv4 network.
3. Any IPv4 routers that the packets encounter use the packets' IPv4 destination address for forwarding. This address is the globally unique IPv4 address of the interface on Router B, which also serves as the 6to4 pseudo-interface.
4. Packets from Site A arrive at Router B, which decapsulates the IPv6 packets from the IPv4 header.
5. Router B then uses the destination address in the IPv6 packet to forward the packets to the recipient host at Site B.

## Considerations for Tunnels to a 6to4 Relay Router

6to4 relay routers function as endpoints for tunnels from 6to4 routers that need to communicate with native IPv6, non-6to4 networks. Relay routers are essentially bridges

between the 6to4 site and native IPv6 sites. Because this solution might be insecure, by default, the Solaris OS does not enable 6to4 relay router support. However, if your site requires such a tunnel, you can use the `6to4relay` command to enable the following tunneling scenario.

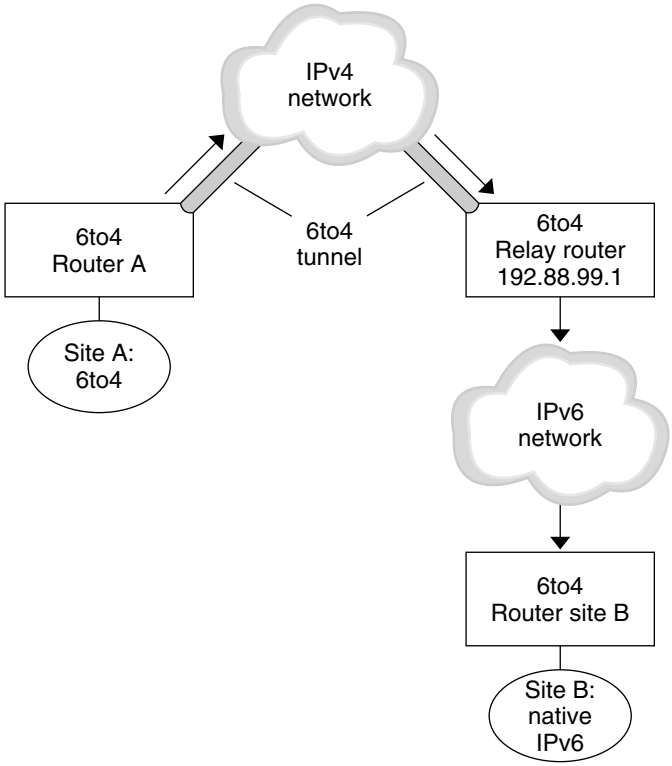



**FIGURE 11–7** Tunnel From a 6to4 Site to a 6to4 Relay Router

In Figure 11–7, 6to4 Site A needs to communicate with a node at the native IPv6 Site B. The figure shows the path of traffic from Site A onto a 6to4 tunnel over an IPv4 network. The tunnel has 6to4 Router A and a 6to4 relay router as its endpoints. Beyond the 6to4 relay router is the IPv6 network, to which IPv6 Site B is connected.

## Packet Flow Between a 6to4 Site and a Native IPv6 Site

This section describes the flow of packets from a 6to4 site to a native IPv6 site. This scenario uses the topology that is shown in Figure 11–7.

1. A host on 6to4 Site A sends a transmission that specifies as the destination a host at native IPv6 Site B. Each packet header has a 6to4-derived address as its source address. The destination address is a standard IPv6 address.

2. Site A's 6to4 router encapsulates each packet within an IPv4 header, which has the IPv4 address of the 6to4 relay router as its destination. The 6to4 router uses standard IPv4 routing procedures to forward the packet over the IPv4 network. Any IPv4 routers that the packets encounter forward the packets to the 6to4 relay router.
3. The physically closest anycast 6to4 relay router to Site A retrieves the packets that are destined for the `192.88.99.1` anycast group.

---

**Note –** 6to4 relay routers that are part of the 6to4 relay router anycast group have the IP address `192.88.99.1`. This anycast address is the default address for 6to4 relay routers. If you need to use a specific 6to4 relay router, you can override the default and specify that router's IPv4 address.

---

4. The relay router decapsulates the IPv4 header from the 6to4 packets, revealing the native IPv6 destination address.
5. The relay router then sends the now IPv6-only packets onto the IPv6 network, where the packets are ultimately retrieved by a router at Site B. The router then forwards the packets to the destination IPv6 node.

# IPv6 Extensions to Solaris Name Services

This section describes naming changes that were introduced by the implementation of IPv6. You can store IPv6 addresses in any of the Solaris naming services, NIS, LDAP, DNS, and files. You can also use NIS over IPv6 RPC transports to retrieve any NIS data.

## DNS Extensions for IPv6

An IPv6-specific resource record, the AAAA resource record, has been specified by in RFC 1886 *DNS Extensions to Support IP Version 6*. This AAAA record maps a host name into a 128 bit IPv6 address. The PTR record is still used with IPv6 to map IP addresses into host names. The 32 four bit nibbles of the 128 bit address are reversed for an IPv6 address. Each nibble is converted to its corresponding hexadecimal ASCII value. Then, `ip6.int` is appended.

## Changes to the `nsswitch.conf` File

For Solaris 10 11/06 and previous releases, in addition to the capability of looking up IPv6 addresses through `/etc/inet/ipnodes`, IPv6 support has been added to the NIS, LDAP, and DNS name services. Consequently, the `nsswitch.conf` file has been modified to support IPv6 lookups.

```
hosts:  files dns nisplus [NOTFOUND=return]
ipnodes: files dns nisplus [NOTFOUND=return]
```

**Note –** Before changing the `/etc/nsswitch.conf` file to search `ipnodes` in multiple name services, populate these `ipnodes` databases with IPv4 and IPv6 addresses. Otherwise, unnecessary delays can result in the resolution of host addresses, including possible boot-timing delays.

The following diagram shows the new relationship between the `nsswitch.conf` file and the new name services databases for applications that use the `gethostbyname` and `getipnodebyname` commands. Items in italics are new. The `gethostbyname` command checks only for IPv4 addresses that are stored in `/etc/inet/hosts`. In Solaris 10 11/06 and previous releases, the `getipnodebyname` command consults the database that is specified in the `ipnodes` entry in the `nsswitch.conf` file. If the lookup fails, then the command checks the database that is specified in the `hosts` entry in the `nsswitch.conf` file.

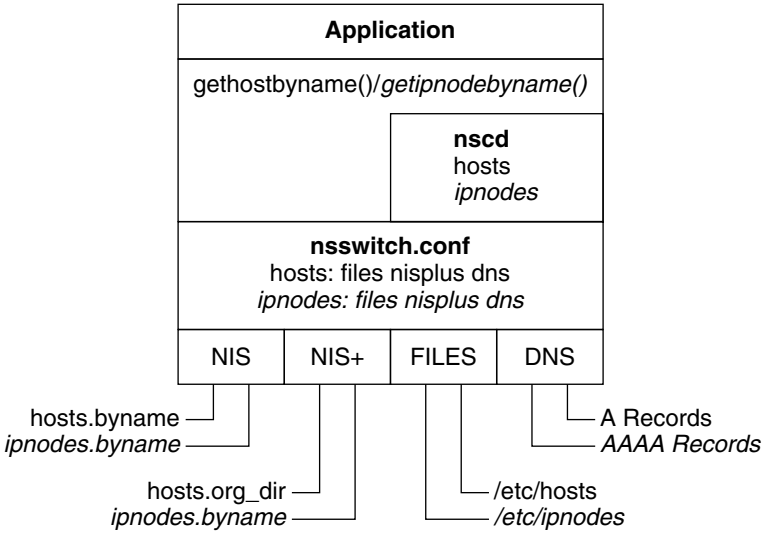



**FIGURE 11–8** Relationship Between `nsswitch.conf` and Name Services

For more information on name services, see *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

# Changes to Name Service Commands

To support IPv6, you can look up IPv6 addresses with the existing name service commands. For example, the `ypmatch` command works with the new NIS maps. The `nslookup` command can look up the new AAAA records in DNS.

# NFS and RPC IPv6 Support

NFS software and Remote Procedure Call (RPC) software support IPv6 in a seamless manner. Existing commands that are related to NFS services have not changed. Most RPC applications also run on IPv6 without any change. Some advanced RPC applications with transport knowledge might require updates.

# IPv6 Over ATM Support

The Solaris OS supports IPv6 over ATM, permanent virtual circuits (PVC), and static switched virtual circuits (SVC).

PART III

# DHCP

This part contains conceptual information about the Dynamic Host Configuration Protocol (DHCP), and tasks for planning, configuring, administering, and troubleshooting the Solaris DHCP service.

◆ ◆ ◆ CHAPTER 12

# About Solaris DHCP (Overview)

This chapter introduces the Dynamic Host Configuration Protocol (DHCP), and explains the concepts that underlie the protocol. This chapter also describes the advantages of using DHCP in your network.

This chapter contains the following information:


- "About the DHCP Protocol" on page 299
- "Advantages of Using Solaris DHCP" on page 300
- "How DHCP Works" on page 301
- "Solaris DHCP Server" on page 304
- "Solaris DHCP Client" on page 312


## About the DHCP Protocol

The DHCP protocol enables host systems in a TCP/IP network to be configured automatically for the network as the systems boot. DHCP uses a client-server mechanism. Servers store and manage configuration information for clients and provide that information upon a client's request. The information includes the client's IP address and information about network services that are available to the client.

DHCP evolved from an earlier protocol, BOOTP, which was designed for booting over a TCP/IP network. DHCP uses the same format as BOOTP for messages between the client and server. However, unlike BOOTP messages, DHCP messages can include network configuration data for the client.

A primary benefit of DHCP is its ability to manage IP address assignments through leases. *Leases* allow IP addresses to be reclaimed when they are not in use. The reclaimed IP addresses can be reassigned to other clients. A site that uses DHCP can use a smaller pool of IP addresses than would be needed if all clients were assigned a permanent IP address.

# Advantages of Using Solaris DHCP

DHCP relieves you of some of the time-consuming tasks involved in setting up a TCP/IP network and in the daily management of that network. Note that Solaris DHCP works only with IPv4.

Solaris DHCP offers the following advantages:

- **IP address management** – A primary advantage of DHCP is easier management of IP addresses. In a network without DHCP, you must manually assign IP addresses. You must be careful to assign unique IP addresses to each client and to configure each client individually. If a client moves to a different network, you must make manual modifications for that client. When DHCP is enabled, the DHCP server manages and assigns IP addresses without administrator intervention. Clients can move to other subnets without manual reconfiguration because they obtain, from a DHCP server, new client information appropriate for the new network.
- **Centralized network client configuration** – You can create a tailored configuration for certain clients, or for certain types of clients. The configuration information is stored in one place, in the DHCP data store. You do not need to log in to a client to change its configuration. You can make changes for multiple clients just by changing the information in the data store.
- **Support of BOOTP clients** – Both BOOTP servers and DHCP servers listen and respond to broadcasts from clients. The DHCP server can respond to requests from BOOTP clients as well as DHCP clients. BOOTP clients receive an IP address and the information needed to boot from a server.
- **Support of local clients and remote clients** – BOOTP provides for the relaying of messages from one network to another network. DHCP takes advantage of the BOOTP relay feature in several ways. Most network routers can be configured to act as BOOTP relay agents to pass BOOTP requests to servers that are not on the client's network. DHCP requests can be relayed in the same manner because, to the router, DHCP requests are indistinguishable from BOOTP requests. The Solaris DHCP server can also be configured to behave as a BOOTP relay agent, if a router that supports BOOTP relay is not available.
- **Network booting** – Clients can use DHCP to obtain the information that is needed to boot from a server on the network, instead of using RARP (Reverse Address Resolution Protocol) and the `bootparams` file. The DHCP server can give a client all the information that the client needs to function, including IP address, boot server, and network configuration information. Because DHCP requests can be relayed across subnets, you can deploy fewer boot servers in your network when you use DHCP network booting. RARP booting requires that each subnet have a boot server.
- **Large network support** – Networks with millions of DHCP clients can use Solaris DHCP. The DHCP server uses multithreading to process many client requests simultaneously. The server also supports data stores that are optimized to handle large amounts of data. Data store access is handled by separate processing modules. This data store approach enables you to add support for any database that you require.

# How DHCP Works

You must first install and configure the DHCP server. During configuration, you specify information about the network that clients need to operate on the network. After this information is in place, clients are able to request and receive network information.

The sequence of events for DHCP service is shown in the following diagram. The numbers in circles correlate to the numbered items in the description following the diagram.

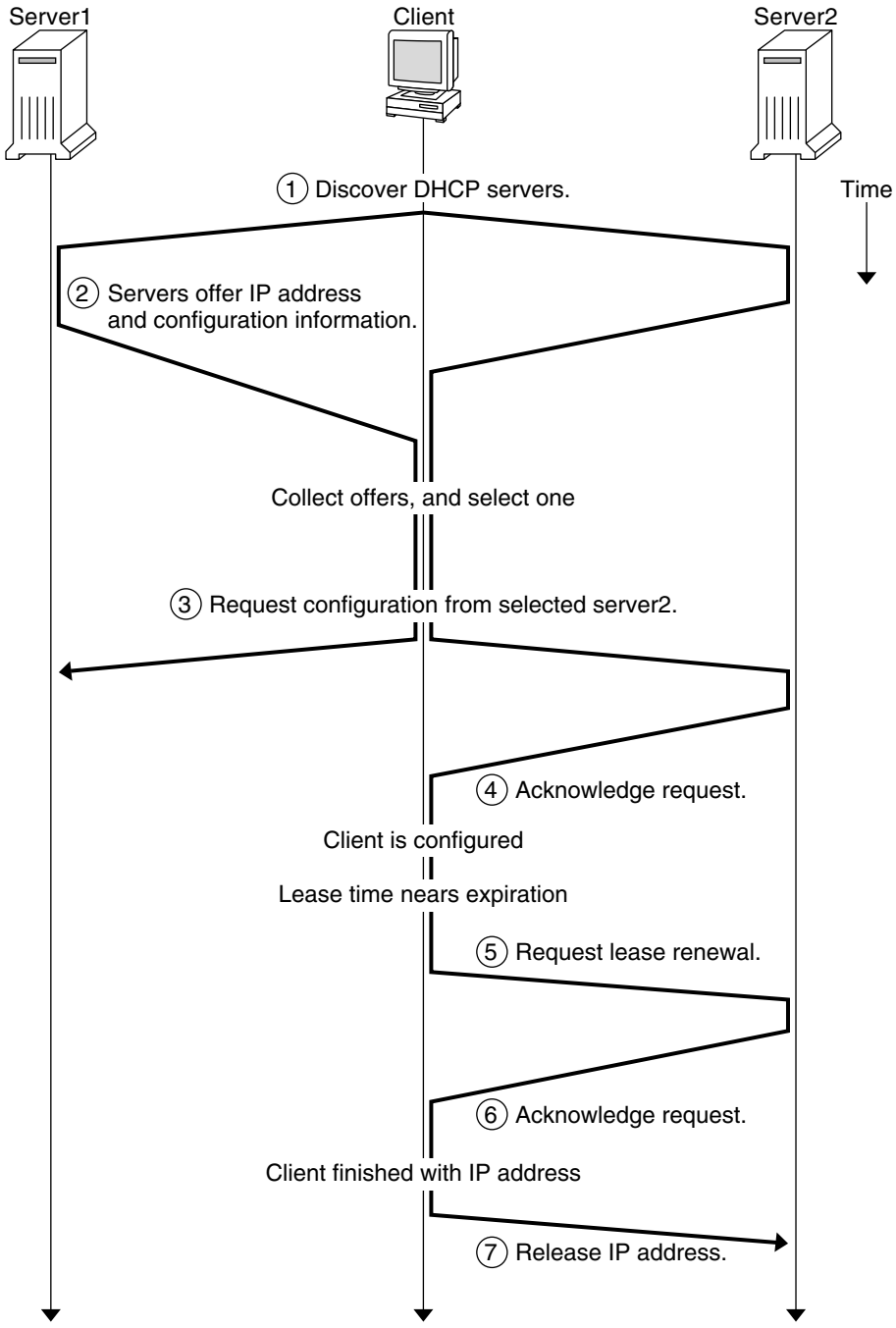



**FIGURE 12–1** Sequence of Events for DHCP Service

The preceding diagram shows the following steps:

1. The client discovers a DHCP server by broadcasting a *discover message* to the limited broadcast address (`255.255.255.255`) on the local subnet. If a router is present and configured to behave as a BOOTP relay agent, the request is passed to other DHCP servers on different subnets. The client's *broadcast* includes its unique ID, which, in the Solaris DHCP implementation, is derived from the client's Media Access Control (MAC) address. On an Ethernet network, the MAC address is the same as the Ethernet address.

   DHCP servers that receive the discover message can determine the client's network by looking at the following information:

   - Which network interface did the request come in on? The server determines either that the client is on the network to which the interface is connected, or that the client is using a BOOTP relay agent connected to that network.
   - Does the request include the IP address of a BOOTP relay agent? When a request passes through a relay agent, the relay agent inserts its address in the request header. When the server detects a *relay agent address*, the server knows that the network portion of the address indicates the client's network address because the relay agent must be connected to the client's network.
   - Is the client's network subnetted? The server consults the `netmasks` table to find the subnet mask used on the network indicated by the relay agent's address or by the address of the network interface that received the request. Once the server knows the subnet mask used, it can determine which portion of the network address is the host portion, and then it can select an IP address appropriate for the client. See the `netmasks`(4) man page for information on `netmasks`.
2. After the DHCP servers determine the client's network, the servers select an appropriate IP address and verify that the address is not already in use. The DHCP servers then respond to the client by broadcasting an *offer message*. The offer message includes the selected IP address and information about services that can be configured for the client. Each server temporarily reserves the offered IP address until the client determines whether to use the IP address.
3. The client selects the best offer, based on the number and type of services offered. The client broadcasts a request that specifies the IP address of the server that made the best offer. The broadcast ensures that all the responding DHCP servers know that the client has chosen a server. The servers that are not chosen can cancel the reservations for the IP addresses that they had offered.
4. The selected server allocates the IP address for the client and stores the information in the DHCP data store. The server also sends an acknowledgement message (ACK) to the client. The *acknowledgement message* contains the network configuration parameters for the client. The client uses the `ping` utility to test the IP address to make sure no other system is using it. The client then continues booting to join the network.
5. The client monitors the lease time. When a set period of time has elapsed, the client sends a new message to the chosen server to increase the lease time.

6. The DHCP server that receives the request extends the lease time if the lease still adheres to the local lease policy set by the administrator. If the server does not respond within 20 seconds, the client broadcasts a request so that one of the other DHCP servers can extend the lease.
7. When the client no longer needs the IP address, the client notifies the server that the IP address is released. This notification can happen during an orderly shutdown and can also be done manually.

# Solaris DHCP Server

The Solaris DHCP server runs as a daemon in the Solaris Operating System (Solaris OS) on a host system. The server has two basic functions:

- **Managing IP addresses** – The DHCP server controls a range of IP addresses and allocates them to clients, either permanently or for a defined period of time. The server uses a lease mechanism to determine how long a client can use a nonpermanent address. When the address is no longer in use, it is returned to the pool and can be reassigned. The server maintains information about the binding of IP addresses to clients in its DHCP network tables, ensuring that no address is used by more than one client.
- **Providing network configuration for clients** – The server assigns an IP address and provides other information for network configuration, such as a host name, broadcast address, network subnet mask, default gateway, name service, and potentially much more information. The network configuration information is obtained from the server's `dhcptab` database.

The Solaris DHCP server can also be configured to perform the following additional functions:

- **Responding to BOOTP client requests** – The server listens for broadcasts from BOOTP clients discovering a BOOTP server and provides them with an IP address and boot parameters. The information must have been configured statically by an administrator. The DHCP server can simultaneously perform as a BOOTP server and as a DHCP server.
- **Relaying requests** – The server relays BOOTP and DHCP requests to appropriate servers on other subnets. The server cannot provide DHCP or BOOTP service when configured as a BOOTP relay agent.
- **Providing network booting support for DHCP clients** – The server can provide DHCP clients with information needed to boot over the network: an IP address, boot parameters, and network configuration information. The server can also provide information that DHCP clients need to boot and install over a wide area network (WAN).
- **Updating DNS tables for clients that supply a host name** – For clients that provide a `Hostname` option and value in their requests for DHCP service, the server can attempt DNS updates on their behalf.

# DHCP Server Management

As superuser, you can start, stop, and configure the DHCP server with DHCP Manager or with command-line utilities described in "DHCP Command-Line Utilities" on page 307. Generally, the DHCP server is configured to start automatically when the system boots, and to stop when the system is shut down. You should not need to start and stop the server manually under normal conditions.

# DHCP Data Store

All the data used by the Solaris DHCP server is maintained in a data store. The data store might consist of plain text files, NIS+ tables, or binary-format files. While configuring the DHCP service, you choose the type of data store to be used. The section "Choosing the DHCP Data Store" on page 320 describes the differences between the types of data stores. You can convert a data store from one format to another by using DHCP Manager or the `dhcpconfig` command.

You can also move data from one DHCP server's data store to another server's data store. You can use export and import utilities that work with the data stores, even if the servers are using different data store formats. You can export and import the entire content of a data store, or just some of the data within it, using DHCP Manager or the `dhcpconfig` command.

---

**Note –** Any database or file format can be used for DHCP data storage if you develop your own code module to provide an interface between Solaris DHCP (server and management tools) and the database. For more information, see the *Solaris DHCP Service Developer's Guide*.

---

Within the Solaris DHCP data store are two types of tables. You can view and manage the contents if these tables by using either DHCP Manager or the command-line utilities. The data tables are as follows:

- `dhcptab` **table** – Table of configuration information that can be passed to clients.
- **DHCP network tables** – Tables containing information about the DHCP and BOOTP clients that reside on the network specified in the table name. For example, the network `192.168.32.0` would have a table whose name includes `192_168_32_0`.

## The `dhcptab` Table

The `dhcptab` table contains all the information that clients can obtain from the DHCP server. The DHCP server scans the `dhcptab` table each time it starts. The file name of the `dhcptab` table varies according to the data store used. For example, the `dhcptab` table created by the NIS+ data store `SUNWnisplus` is `SUNWnisplus1_dhcptab`.

The DHCP protocol defines a number of standard items of information that can be passed to clients. These items are referred to as parameters, symbols, or options. Options are defined in

the DHCP protocol by numeric codes and text labels, but without values. Some commonly used standard options are shown in the following table.

**TABLE 12–1** Sample DHCP Standard Options

| Code | Label | Description |
|---|---|---|
| 1 | `Subnet` | Subnet mask IP address |
| 3 | `Router` | IP address for the router |
| 6 | `DNSserv` | IP address for the DNS server |
| 12 | `Hostname` | Text string for the client host name |
| 15 | `DNSdmain` | DNS domain name |

Some options are automatically assigned values when you provide information during server configuration. You can also explicitly assign values to other options at a later time. Options and their values are passed to the client to provide configuration information. For example, the option/value pair, `DNSdmain=Georgia.Peach.COM`, sets the client's DNS domain name to `Georgia.Peach.COM`.

Options can be grouped with other options in containers known as *macros*, which makes it easier to pass information to a client. Some macros are created automatically during server configuration and contain options that were assigned values during configuration. Macros can also contain other macros.

The format of the `dhcptab` table is described in the `dhcptab`(4) man page. In DHCP Manager, all the information shown in the Options and Macros tabs comes from the `dhcptab` table. See "About DHCP Options" on page 310 for more information about options. See "About DHCP Macros" on page 311 for more information about macros.

Note that the `dhcptab` table should not be edited manually. You should use either the `dhtadm` command or DHCP Manager to create, delete, or modify options and macros.

### DHCP Network Tables

A DHCP network table maps client identifiers to IP addresses and the configuration parameters associated with each address. The format of the network tables is described in the `dhcp_network`(4) man page. In DHCP Manager, all the information shown in the Addresses tab comes from the network tables.

## DHCP Manager

DHCP Manager is a graphical user interface (GUI) tool you can use to perform all management duties associated with the DHCP service. You can use it to manage the server as well as the data the server uses. You must be superuser when you run DHCP Manager.

You can use DHCP Manager with the server in the following ways:

- Configuring and unconfiguring the DHCP server
- Starting, stopping, and restarting the DHCP server
- Disabling and enabling DHCP service
- Customizing DHCP server settings

DHCP Manager enables you to manage the IP addresses, network configuration macros, and network configuration options in the following ways:

- Adding and deleting networks under DHCP management
- Viewing, adding, modifying, deleting, and releasing IP addresses under DHCP management
- Viewing, adding, modifying, and deleting network configuration macros
- Viewing, adding, modifying, and deleting nonstandard network configuration options

DHCP Manager allows you to manage the DHCP data stores in the following ways:

- Convert data to a new data store format
- Move DHCP data from one DHCP server to another by exporting it from the first server and importing it on the second server

DHCP Manager includes extensive online help for procedures you can perform with the tool. For more information, see "About DHCP Manager" on page 340.

# DHCP Command-Line Utilities

All DHCP management functions can be performed by using command-line utilities. You can run the utilities if you are logged in as superuser or as a user assigned to the DHCP Management profile. See "Setting Up User Access to DHCP Commands" on page 343.

The following table lists the utilities and describes the purpose of each utility.

**TABLE 12–2** DHCP Command-Line Utilities

| Command | Description and Purpose | Man Page Links |
|---|---|---|
| `in.dhcpd` | The DHCP service daemon. Command-line arguments enable you to set several runtime options. | `in.dhcpd`(1M) |

**TABLE 12–2** DHCP Command-Line Utilities *(Continued)*

| Command | Description and Purpose | Man Page Links |
|---|---|---|
| `dhcpconfig` | Used to configure and unconfigure a DHCP server. This utility enables you to perform many of the functions of DHCP Manager from the command line. This utility is primarily intended for use in scripts for sites that want to automate some configuration functions. `dhcpconfig` collects information from the server system's network topology files to create useful information for the initial configuration. | `dhcpconfig`(1M) |
| `dhtadm` | Used to add, delete, and modify configuration options and macros for DHCP clients. This utility lets you edit the `dhcptab` table indirectly, which ensures the correct format of the `dhcptab` table. You should not directly edit the `dhcptab` table. | `dhtadm`(1M) |
| `pntadm` | Used to manage the DHCP network tables. You can use this utility to perform the following tasks:<br>■ Add and remove IP addresses and networks under DHCP management.<br>■ Modify the network configuration for specified IP addresses.<br>■ Display information about IP addresses and networks under DHCP management. | `pntadm`(1M) |

# Role-Based Access Control for DHCP Commands

Security for the `dhcpconfig`, `dhtadm`, and `pntadm` commands is determined by role-based access control (RBAC) settings. By default, the commands can be run only by superuser. If you want to use the commands under another user name, you must assign the user name to the DHCP Management profile as described in "Setting Up User Access to DHCP Commands" on page 343.

# DHCP Server Configuration

You configure the Solaris DHCP server the first time you run DHCP Manager on the system where you want to run the DHCP server.

DHCP Manager server configuration dialog boxes prompt you for essential information needed to enable and run the DHCP server on one network. Some default values are obtained from existing system files. If you have not configured the system for the network, there are no default values. DHCP Manager prompts for the following information:

- Role of the server, either as the DHCP server or as the BOOTP relay agent
- Data store type (files, binary files, NIS+, or something specific to your site)
- Data store configuration parameters for the data store type you selected
- Name service to use to update host records, if any (`/etc/hosts`, NIS+, or DNS)
- Length of lease time and whether clients should be able to renew leases
- DNS domain name and IP addresses of DNS servers
- Network address and subnet mask for the first network you want to configure for DHCP service
- Network type, either local area network (LAN) or point-to-point network
- Router discovery or the IP address of a particular router
- NIS domain name and IP address of NIS servers
- NIS+ domain name and IP address of NIS+ servers

You can also configure the DHCP server using the `dhcpconfig` command. This utility automatically gathers information from existing system files to provide a useful initial configuration. Therefore, you must ensure that the files are correct before running `dhcpconfig`. See the `dhcpconfig`(1M) man page for information about the files that `dhcpconfig` uses to obtain information.

## IP Address Allocation

The Solaris DHCP server supports the following types of IP address allocation:

- **Manual allocation** – The server provides a specific IP address that you choose for a specific DHCP client. The address cannot be reclaimed or assigned to another client.
- **Automatic, or permanent, allocation** – The server provides an IP address that has no expiration time, making it permanently associated with the client until you change the assignment or the client releases the address.
- **Dynamic allocation** – The server provides an IP address to a requesting client, with a lease for a specific period of time. When the lease expires, the address is taken back by the server and can be assigned to another client. The period of time is determined by the lease time configured for the server.

# Network Configuration Information

You determine what information to provide to DHCP clients. When you configure the DHCP server, you provide essential information about the network. Later, you can add more information that you want to provide to clients.

The DHCP server stores network configuration information in the `dhcptab` table, in the form of option/value pairs and macros. Options are keywords for network data that you want to supply to clients. Values are assigned to options and passed to clients in DHCP messages. For example, the NIS server address is passed by way of an option called `NISservs`. The `NISservs` option has a value that is equal to a list of IP addresses, which is assigned by the DHCP server. Macros provide a convenient way to group together any number of options that you want to supply to clients. You can use DHCP Manager to create macros to group options and to assign values to the options. If you prefer a command-line tool, you can use `dhtadm`, the DHCP configuration table management utility, to work with options and macros.

# About DHCP Options

In Solaris DHCP, an *option* is a piece of network information to be passed to a client. The DHCP literature also refers to options as *symbols* or *tags*. An option is defined by a numeric code and a text label. An option receives a value when it is used in the DHCP service.

The DHCP protocol defines a large number of standard options for commonly specified network data: `Subnet`, `Router`, `Broadcst`, `NIS+dom`, `Hostname`, and `LeaseTim` are a few examples. A complete list of standard options is shown in the `dhcp_inittab`(4) man page. You cannot modify the standard option keywords in any way. However, you can assign values to the options that are relevant to your network when you include the options in macros.

You can create new options for data that is not represented by the standard options. Options you create must be classified in one of three categories:

- **Extended** – Reserved for options that have become standard DHCP options but are not yet included in the DHCP server implementation. You might use an extended option if you know of a standard option that you want to use, but you do not want to upgrade your DHCP server.
- **Site** – Reserved for options that are unique to your site. You create these options.
- **Vendor** – Reserved for options that should apply only to clients of a particular class, such as a hardware or vendor platform. The Solaris DHCP implementation includes a number of vendor options for Solaris clients. For example, the option `SrootIP4` is used to specify the IP address of a server that a client that boots from the network should use for its root (/) file system.

Chapter 15, "Administering DHCP (Tasks)," includes procedures for creating, modifying, and deleting DHCP options.

# About DHCP Macros

In the Solaris DHCP service, a *macro* is a collection of network configuration options and the values that you assign to them. Macros are created to group options together to be passed to specific clients or types of clients. For example, a macro intended for all clients of a particular subnet might contain option/value pairs for subnet mask, router IP address, broadcast address, NIS+ domain, and lease time.

## Macro Processing by the DHCP Server

When the DHCP server processes a macro, it places the network options and values defined in the macro in a DHCP message to a client. The server processes some macros automatically for clients of a particular type.

For the server to process a macro automatically, the name of the macro must comply with one of the categories shown in the following table.

**TABLE 12–3** DHCP Macro Categories for Automatic Processing

| Macro Category | Description |
|---|---|
| Client class | The macro name matches a class of client, indicated by the client machine type, operating system, or both. For example, if a server has a macro named `SUNW.Sun-Blade-100`, any client whose hardware implementation is `SUNW,Sun-Blade-100` automatically receives the values in the `SUNW.Sun-Blade-100` macro. |
| Network address | The macro name matches a DHCP-managed network IP address. For example, if a server has a macro named `10.53.224.0`, any client connected to the `10.53.224.0` network automatically receives the values in the `10.53.224.0` macro. |
| Client ID | The macro name matches some unique identifier for the client, usually derived from an Ethernet or MAC address. For example, if a server has a macro named `08002011DF32`, the client with the client ID `08002011DF32` (derived from the Ethernet address `8:0:20:11:DF:32`) automatically receives the values in the macro named `08002011DF32`. |

A macro with a name that does not use one of the categories listed in Table 12–3 can be processed only if one of the following is true:

- The macro is mapped to an IP address.
- The macro is included in another macro that is processed automatically.
- The macro is included in another macro that is mapped to an IP address.

**Note –** When you configure a server, a macro that is named to match the server's name is created by default. This server macro is *not* processed automatically for any client because it is not named with one of the name types that cause automatic processing. When you later create IP addresses on the server, the IP addresses are mapped to use the server macro by default.

### Order of Macro Processing

When a DHCP client requests DHCP services, the DHCP server determines which macros match the client. The server processes the macros, using the macro categories to determine the order of processing. The most general category is processed first, and the most specific category is processed last. The macros are processed in the following order:

1. Client class macros – The most general category
2. Network address macros – More specific than Client class
3. Macros mapped to IP addresses – More specific than Network address
4. Client ID macros – The most specific category, pertaining to one client

A macro that is included in another macro is processed as part of the container macro.

If the same option is included in more than one macro, the value for that option in the macro with the most specific category is used because it is processed last. For example, if a Network address macro contains the lease time option with a value of 24 hours, and a Client ID macro contains the lease time option with a value of 8 hours, the client receives a lease time of 8 hours.

### Size Limit for DHCP Macros

The sum total of the values assigned to all the options in a macro must not exceed 255 bytes, including the option codes and length information. This limit is dictated by the DHCP protocol.

The macros that are most likely to be impacted by this limit are macros that are used to pass paths to files on Solaris installation servers. Generally, you should pass the minimum amount of vendor information needed. You should use short path names in options that require path names. If you create symbolic links to long paths, you can pass the shorter link names.

## Solaris DHCP Client

The term “client” is sometimes used to refer to a physical machine that is performing a client role on the network. However, the DHCP client described in this document is a software entity. The Solaris DHCP client is a daemon (`dhcpagent`) that runs in the Solaris OS on a system that is configured to receive its network configuration from a DHCP server. DHCP clients from other vendors can also use the services of the Solaris DHCP server. However, this document describes only the Solaris DHCP client.

See Chapter 16, "Configuring and Administering the DHCP Client," for detailed information about the Solaris DHCP client.

◆ ◆ ◆ CHAPTER 13

# Planning for DHCP Service (Tasks)

You can use the DHCP service in a network that you are creating or in a network that exists. If you are setting up a network, see Chapter 2, "Planning Your TCP/IP Network (Tasks)," before you attempt to set up the DHCP service. If the network already exists, continue in this chapter.

This chapter describes what you need to do before you set up the DHCP service on your network. The information is intended for use with DHCP Manager, although you can also use the command-line utility `dhcpconfig` to set up the DHCP service.

This chapter contains the following information:


- "Preparing Your Network for the DHCP Service (Task Map)" on page 315
- "Making Decisions for Your DHCP Server Configuration (Task Map)" on page 319
- "Making Decisions for IP Address Management (Task Map)" on page 322
- "Planning for Multiple DHCP Servers" on page 326
- "Planning DHCP Configuration of Your Remote Networks" on page 326
- "Selecting the Tool for Configuring DHCP" on page 327


## Preparing Your Network for the DHCP Service (Task Map)

Before you set up your network to use DHCP, you must collect information to help you make decisions for configuring one or more servers. Use the following task map to identify the tasks for preparing your network for DHCP.

| Task | Description | For Instructions |
|---|---|---|
| Map your network topology. | Determine and locate the services that are available on the network. | "Mapping Your Network Topology" on page 316 |

| Task | Description | For Instructions |
|---|---|---|
| Determine the number of DHCP servers you need. | Use the expected number of DHCP clients as a basis for determining the number of DHCP servers you need. | "Determining the Number of DHCP Servers" on page 317 |
| Update system files and `netmasks` table. | Reflect the network topology accurately. | "Updating System Files and Netmask Tables" on page 318 |

# Mapping Your Network Topology

If you have not already done so, you should map the physical structure of your network. Indicate the location of routers and clients, and the location of servers that provide network services. This map of your network topology can help you determine which server to use for the DHCP service. The map can also help you determine the configuration information that the DHCP server can provide to clients.

See Chapter 2, "Planning Your TCP/IP Network (Tasks)," for more information about planning your network.

The DHCP configuration process can gather some network information from the server's system and network files. "Updating System Files and Netmask Tables" on page 318 discusses these files. However, you might want to give clients other service information, which you must enter into the server's macros. As you examine your network topology, record the IP addresses of any servers you want your clients to know about. The following servers, for example, might provide services on your network. The DHCP configuration does not discover these servers.

- Time server
- Log server
- Print server
- Install server
- Boot server
- Web proxy server
- Swap server
- X Window font server
- Trivial File Transfer Protocol (TFTP) server

## Network Topology to Avoid

In some IP network environments, several local area networks (LANs) share the same network hardware media. The networks may use multiple network hardware interfaces or multiple logical interfaces. DHCP does not work well in this kind of shared media network. When multiple LANs run across the same physical network, a DHCP client's request arrives on all network hardware interfaces. This effect makes the client appear to be attached to all of the IP networks simultaneously.

DHCP must be able to determine the address of a client's network in order to assign an appropriate IP address to the client. If more than one network is present on the hardware media, the server cannot determine the client's network. The server cannot assign an IP address without knowing the network number.

You can use DHCP on only one of the networks. If one network does not suit your DHCP needs, you must reconfigure the networks. You should consider the following suggestions:

- Use a variable length subnet mask (VLSM) on your subnets to make better use of the IP address space you have. You may not need to run multiple networks on the same physical network. See the `netmasks`(4) man page for information about implementing variable length subnetting. For more detailed information about Classless Inter-Domain Routing (CIDR) and VLSM, see `http://www.ietf.org/rfc/rfc1519.txt`.
- Configure the ports on your switches to assign devices to different physical LANs. This technique preserves the mapping of one LAN to one IP network, required for Solaris DHCP. See the documentation for the switch for information about port configuration.

## Determining the Number of DHCP Servers

The data store option that you choose has a direct effect on the number of servers you must have to support your DHCP clients. The following table shows the maximum number of DHCP and BOOTP clients that can be supported by one DHCP server for each data store.

**TABLE 13–1** Estimated Maximum Number of Clients Supported by One DHCP Server

| Data Store Type | Maximum Number of Clients Supported |
|---|---|
| Text files | 10,000 |
| NIS+ | 40,000 |
| Binary files | 100,000 |

This maximum number is a general guideline, not an absolute number. A DHCP server's client capacity depends greatly on the number of transactions per second that the server must process. Lease times and usage patterns have a significant impact on the transaction rate. For example, suppose leases are set to 12 hours and users turn their systems off at night. If many users turn on their systems at the same time in the morning, the server must handle transaction peaks as many clients request leases simultaneously. The DHCP server can support fewer clients in such an environment. The DHCP server can support more clients in an environment with longer leases, or an environment that consists of constantly connected devices such as cable modems.

The section "Choosing the DHCP Data Store" on page 320 compares the types of data stores.

# Updating System Files and Netmask Tables

During DHCP configuration, the DHCP tools scan various system files on your server for information that can be used to configure the server.

You must be sure the information in the system files is current before you run DHCP Manager or `dhcpconfig` to configure your server. If you notice errors after you configure the server, use DHCP Manager or `dhtadm` to modify the macros on the server.

The following table lists some of the information gathered during DHCP server configuration, and the sources for the information. Be sure this information is set correctly on the server before you configure DHCP on the server. If you make changes to the system files after you configure the server, you should reconfigure the service to reflect these changes.

**TABLE 13–2** Information Used for DHCP Configuration

| Information | Source | Comments |
|---|---|---|
| Time zone | System date, time zone settings | The date and time zone are initially set during Solaris installation. You can change the date by using the `date` command. You can change the time zone by editing the `/etc/default/init` file to set the `TZ` environment variable. See the `TIMEZONE`(4) man page for more information. |
| DNS parameters | `/etc/resolv.conf` | The DHCP server uses the `/etc/resolv.conf` file to obtain DNS parameters such as the DNS domain name and DNS server addresses. See *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* or the `resolv.conf`(4) man page for more information about `resolv.conf`. |
| NIS or NIS+ parameters | System domain name, `nsswitch.conf`, NIS or NIS+ | The DHCP server uses the `domainname` command to obtain the domain name of the server system. The `nsswitch.conf` file tells the server where to look for domain-based information. If the server system is an NIS or NIS+ client, the DHCP server performs a query to get NIS or NIS+ server IP addresses. See the `nsswitch.conf`(4) man page for more information. |

**TABLE 13–2** Information Used for DHCP Configuration *(Continued)*

| Information | Source | Comments |
|---|---|---|
| Default router | System routing tables, user prompt | The DHCP server searches the network routing tables to find the default router for clients that are attached to the local network. For clients not on the same network, the DHCP server must prompt you for the information. |
| Subnet mask | Network interface, `netmasks` table | The DHCP server looks to its own network interfaces to determine the netmask and broadcast address for local clients. If the request was forwarded by a relay agent, the server obtains the subnet mask in the `netmasks` table on the relay agent's network. |
| Broadcast address | Network interface, `netmasks` table | For the local network, the DHCP server obtains the broadcast address by querying the network interface. For remote networks, the server uses the BOOTP relay agent's IP address and the remote network's netmask to calculate the broadcast address for the network. |

# Making Decisions for Your DHCP Server Configuration (Task Map)

This section discusses some of the decisions to make before you configure the first DHCP server on your network. Use this task map to identify the decisions that you must make.

| Task | Description | For Instructions |
|---|---|---|
| Select a server for DHCP. | Determine if a server meets the system requirements to run the DHCP service. | "Selecting a Host to Run the DHCP Service" on page 320 |
| Choose a data store. | Compare the data store types to determine the best data store for your site. | "Choosing the DHCP Data Store" on page 320 |
| Set a lease policy. | Learn about IP address leases to help you determine appropriate lease policy for your site. | "Setting a Lease Policy" on page 321 |
| Select a router address or router discovery. | Determine whether DHCP clients use router discovery or a specific router. | "Determining Routers for DHCP Clients" on page 322 |

## Selecting a Host to Run the DHCP Service

With your network topology in mind, you can use the following system requirements to select a host on which to set up a DHCP server.

The host must meet the following requirements:

- The host must run the Solaris 2.6 release or later. If you need to support a large number of clients, you must install the Solaris 8 7/01 release or a later version.
- The host must be accessible to all the networks that have clients that plan to use DHCP, either directly on the network or through a BOOTP relay agent.
- The host must be configured to use routing.
- The host must have a correctly configured `netmasks` table that reflects your network topology.

## Choosing the DHCP Data Store

You can choose to store the DHCP data in text files, binary files, or the NIS+ directory service. The following table summarizes the features of each type of data store, and indicates the environment in which to use each data store type.

**TABLE 13–3** Comparison of DHCP Data Stores

| Data Store Type | Performance | Maintenance | Sharing | Environment |
|---|---|---|---|---|
| Binary files | High performance, high capacity | Low maintenance, no database servers required. Contents must be viewed with DHCP Manager or `dhtadm` and `pntadm`. Regular file backups suggested. | Data stores cannot be shared among DHCP servers. | Midsize to large environments with many networks with thousands of clients per network. Useful for small to medium ISPs. |
| NIS+ | Moderate performance and capacity, dependent upon NIS+ service's performance and capacity | DHCP server system must be configured as an NIS+ client. Requires NIS+ service maintenance. Contents must be viewed with DHCP Manager or `dhtadm` and `pntadm`. Regular backup with `nisbackup` is suggested. | DHCP data is distributed in NIS+, and multiple servers can access the same containers. | Small to midsize environments with up to 5000 clients per network. |

**TABLE 13–3** Comparison of DHCP Data Stores *(Continued)*

| Data Store Type | Performance | Maintenance | Sharing | Environment |
|---|---|---|---|---|
| Text files | Moderate performance, low capacity | Low maintenance, no database servers required. ASCII format is readable without DHCP Manager, `dhtadm`, or `pntadm`. Regular file backups suggested. | Data store can be shared among DHCP servers if DHCP data is stored on one file system that is exported through an NFS mount point. | Small environments with less than 10,000 clients, with a few hundred to a thousand clients per network. |

Traditional NIS is not offered as a data store option because NIS does not support fast incremental updates. If your network uses NIS, you should use text files or binary files for your data store.

## 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. The *lease policy* indicates the lease time and specifies whether clients can renew their leases. The server uses the information that you supply to set option values in the default macros that the server creates during configuration. You can set different lease policies for specific clients or type of clients, by setting options in configuration macros you create.

The *lease time* is specified as a number of hours, days, or weeks for which the lease is valid. When a client is assigned an IP address, or renegotiates a lease on an IP address, the lease expiration date and time is calculated. The number of hours in the lease time is added to the timestamp on the client's DHCP acknowledgement. For example, suppose the timestamp of the DHCP acknowledgment is September 16, 2005 9:15 A.M., and the lease time is 24 hours. The lease expiration time in this example is September 17, 2005 9:15 A.M. The lease expiration time is stored in the client's DHCP network record, viewable in DHCP Manager or with the `pntadm`utility.

The lease time value should be relatively small so that expired addresses are reclaimed quickly. The lease time value also should be large enough to outlast DHCP service disruptions. Clients should be able to function while the system that runs the DHCP service is repaired. A general guideline is to specify a time that is two times the predicted downtime of a system. For example, if you need four hours to obtain and replace a defective part and reboot the system, specify a lease time of eight hours.

The lease negotiation option determines whether a client can renegotiate its lease with the server before the lease expires. If lease negotiation is allowed, the client tracks the time that remains in its lease. When half of the lease time has passed, the client requests the DHCP server to extend its lease to the original lease time. You should disable lease negotiation in

environments where there are more systems than IP addresses. The time limit is then enforced on the use of IP addresses. If there are enough IP addresses, you should enable lease negotiation to avoid forcing clients to take down their network interfaces when leases expire. If you make clients obtain new leases, the clients' TCP connections such as NFS and telnet sessions might be interrupted. You can enable lease negotiation for all clients during the server configuration. You can enable lease negotiation for particular clients or particular types of clients through the use of the `LeaseNeg` option in configuration macros.

---

**Note –** Systems that provide services on the network should retain their IP addresses. Such systems should not be subject to short-term leases. You can use DHCP with such systems if you assign reserved manual IP addresses to those systems, rather than IP addresses with permanent leases. You can then detect when the system's IP address is no longer in use.

---

## Determining Routers for DHCP Clients

Host systems use routers for any network communication beyond their local network. The hosts must know the IP addresses of these routers.

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.

If clients on your network can perform router discovery, you should use the router discovery protocol, even if there is only one router. Router discovery enables a client to adapt easily to router changes in the network. For example, suppose that a router fails and is replaced by a router with a new address. Clients can discover the new address automatically without having to obtain a new network configuration to get the new router address.

# Making Decisions for IP Address Management (Task Map)

As part of the DHCP service setup, you determine several aspects of the IP addresses that the server is to manage. If your network needs more than one DHCP server, you can assign responsibility for some IP addresses to each server. You must decide how to divide responsibility for the addresses. The following task map can help you make IP address management decisions.

| Task | Description | For Information |
|---|---|---|
| Specify which addresses that the server should manage. | Determine how many addresses you want the DHCP server to manage, and what those addresses are. | "Number and Ranges of IP Addresses" on page 323 |
| Decide if the server should automatically generate host names for clients. | Learn how client host names are generated so that you can decide whether to generate host names. | "Client Host Name Generation" on page 323 |
| Determine what configuration macro to assign to clients. | Learn about client configuration macros so that you can select an appropriate macro for clients. | "Default Client Configuration Macros" on page 324 |
| Determine lease types to use. | Learn about lease types to help you determine what type is best for your DHCP clients. | "Dynamic and Permanent Lease Types" on page 325 |

## Number and Ranges of IP Addresses

During the initial server configuration, DHCP Manager allows you to add one block, or range, of IP addresses under DHCP management by specifying the total number of addresses and the first address in the block. DHCP Manager adds a list of contiguous addresses from this information. If you have several blocks of noncontiguous addresses, you can add the others by running DHCP Manager's Address Wizard again, after the initial configuration.

Before you configure your IP addresses, know how many addresses are in the initial block of addresses you want to add and the IP address of the first address in the range.

## Client Host Name Generation

The dynamic nature of DHCP means that an IP address is not permanently associated with the host name of the system that is using it. The DHCP management tools can generate a client name to associate with each IP address if you select this option. The client names consist of a prefix, or root name, plus a dash and a number assigned by the server. For example, if the root name is `charlie`, the client names are `charlie-1`, `charlie-2`, `charlie-3`, and so on.

By default, generated client names begin with the name of the DHCP server that manages them. This strategy is useful in environments that have more than one DHCP server because you can quickly see in the DHCP network tables which clients any given DHCP server manages. However, you can change the root name to any name you choose.

Before you configure your IP addresses, decide if you want the DHCP management tools to generate client names, and if so, what root name to use for the names.

The generated client names can be mapped to IP addresses in `/etc/inet/hosts`, DNS, or NIS+ if you specify to register host names during DHCP configuration. See "Client Host Name Registration" on page 355 for more information.

## Default Client Configuration Macros

In Solaris DHCP, a *macro* is a collection of network configuration options and their assigned values. The DHCP server uses macros to determine what network configuration information to send to a DHCP client.

When you configure the DHCP server, the management tools gather information from system files and directly from you through prompts or command-line options you specify. With this information, the management tools create the following macros:

- **Network address macro** — The network address macro is named to match the IP address of the client network. For example, if the network is `192.68.0.0`, the network address macro is also named `192.68.0.0`. The macro contains information needed by any client that is part of the network, such as subnet mask, network broadcast address, default router or router discovery token, and NIS/NIS+ domain and server if the server uses NIS/NIS+. Other options that are applicable to your network might be included. The network address macro is automatically processed for all clients located on that network, as described in "Order of Macro Processing" on page 312.
- **Locale macro** — The locale macro is named `Locale`. The macro contains the offset (in seconds) from Coordinated Universal Time (UTC) to specify the time zone. The locale macro is not automatically processed, but is included in the server macro.
- **Server macro** — The server macro is named to match the server's host name. For example, if the server is named `pineola`, the server macro is also named `pineola`. The server macro contains information about the lease policy, time server, DNS domain, and DNS server, and possibly other information that the configuration program was able to obtain from system files. The server macro includes the locale macro, so the DHCP server processes the locale macro as part of the server macro.

  When you configure IP addresses for the first network, you must select a client configuration macro to be used for all DHCP clients that use the addresses you are configuring. The macro that you select is mapped to the IP addresses. By default, the server macro is selected because the macro contains information needed by all clients that use this server.

Clients receive the options contained in the network address macro before the options in the macro that is mapped to IP addresses. This processing order causes the options in the server macro to take precedence over any conflicting options in the network address macro. See "Order of Macro Processing" on page 312 for more information about the order in which macros are processed.

## Dynamic and Permanent Lease Types

The *lease type* determines whether the lease policy applies to the IP addresses you are configuring. During initial server configuration, DHCP Manager allows you to select either dynamic or permanent leases for the addresses you are adding. If you configure the DHCP server with the `dhcpconfig` command, leases are dynamic.

When an IP address has a *dynamic lease*, the DHCP server can manage the address. The DHCP server can allocate the IP address to a client, extend the lease time, detect when the address is no longer in use, and reclaim the address. When an IP address has a *permanent lease*, the DHCP server can only allocate the address. The client then owns the address until explicitly releasing the address. When the address is released, the server can assign the address to another client. The address is not subject to the lease policy as long as the address is configured with a permanent lease type.

When you configure a range of IP addresses, the lease type you select applies to all the addresses in the range. To get the most benefit from DHCP, you should use dynamic leases for most of the addresses. You can later modify individual addresses to make them permanent, if necessary. However, the total number of permanent leases should be kept to a minimum.

## Reserved IP Addresses and Lease Type

IP addresses can be reserved by manually assigning them to particular clients. A reserved address can be associated with a permanent lease or a dynamic lease. When a reserved address is assigned a permanent lease, the following statements are true:

- The address can be allocated only to the client that is bound to the address.
- The DHCP server cannot allocate the address to another client.
- The address cannot be reclaimed by the DHCP server.

If a reserved address is assigned a dynamic lease, the address can be allocated only to the client that is bound to the address. However, the client must track lease time and negotiate for a lease extension as if the address were not reserved. This strategy enables you to track when the client is using the address by looking at the network table.

You cannot create reserved addresses for all the IP addresses during the initial configuration. Reserved addresses are intended to be used sparingly for individual addresses.

# Planning for Multiple DHCP Servers

If you want to configure more than one DHCP server to manage your IP addresses, consider the following guidelines:

- Divide the pool of IP addresses so that each server is responsible for a range of addresses, and there is no overlap of responsibility.
- Choose NIS+ as your data store, if available. If not, choose text files and specify a shared directory for the absolute path to the data store. The binary files data store cannot be shared.
- Configure each server separately so that address ownership is allocated correctly and so that server-based macros can be automatically created.
- Set up the servers to scan the options and macros in the `dhcptab` table at specified intervals so that the servers are using the latest information. You can use DHCP Manager to schedule automatic reading of `dhcptab` as described in "Customizing Performance Options for the DHCP Server" on page 356.
- Be sure all clients can access all DHCP servers so that the servers can support one another. A client that has a valid IP address lease might try to verify its configuration or extend the lease when the server that owns the client's address is not reachable. Another server can respond to the client after the client has attempted to contact the primary server for 20 seconds. If a client requests a specific IP address, and the server that owns the address is not available, one of the other servers handles the request. In this case, the client does not receive the requested address. The client receives an IP address that is owned by the responding DHCP server.

# Planning DHCP Configuration of Your Remote Networks

After the initial DHCP configuration, you can place IP addresses in remote networks under DHCP management. However, because the system files are not local to the server, DHCP Manager and `dhcpconfig` cannot look up information to provide default values, so you must provide the information. Before you try to configure a remote network, be sure you know the following information:

- The remote network's IP address.
- The subnet mask of the remote network. This information can be obtained from the `netmasks` table in the name service. If the network uses local files, look in `/etc/netmasks` on a system in the network. If the network uses NIS+, use the command `niscat netmasks.org_dir`. If the network uses NIS, use the command `ypcat -k netmasks.byaddr`. Make sure the `netmasks` table contains all the topology information for all the subnets you want to manage.
- The network type. The clients connect to the network through either a local area network (LAN) connection or a Point-to-Point Protocol (PPP).
- Routing information. Can the clients use router discovery? If not, you must determine the IP address of a router they can use.

- NIS domain and NIS servers, if applicable.
- NIS+ domain and NIS+ servers, if applicable.

See "Adding DHCP Networks" on page 360 for the procedure for adding DHCP networks.

# Selecting the Tool for Configuring DHCP

After you gather information and plan for DHCP service, you are ready to configure a DHCP server. You can use the DHCP Manager or the command-line utility `dhcpconfig` to configure a server. DHCP Manager lets you select options and specify data that is then used to create the `dhcptab` and network tables used by the DHCP server. The `dhcpconfig` utility requires you to use command-line options to specify data.

## DHCP Manager Features

DHCP Manager, a Java™ technology-based GUI tool, provides a DHCP Configuration Wizard. The configuration wizard starts automatically the first time you run DHCP Manager on a system that is not configured as a DHCP server. The DHCP Configuration Wizard provides a series of dialog boxes that prompt you for the essential information required to configure a server: data store format, lease policy, DNS/NIS/NIS+ servers and domains, and router addresses. Some of the information is obtained by the wizard from system files, and you only need to confirm that the information is correct, or to correct information, if necessary.

When you progress through the dialog boxes and approve the information, the DHCP server daemon starts on the server system. You are then prompted to start the Add Addresses Wizard to configure IP addresses for the network. Only the server's network is configured for DHCP initially, and other server options are given default values. You can run DHCP Manager again after the initial configuration is complete to add networks and modify other server options.

See "Configuring and Unconfiguring a DHCP Server Using DHCP Manager" on page 329 for more information about the DHCP Configuration Wizard. See "About DHCP Manager" on page 340 for more detailed information about DHCP Manager.

## `dhcpconfig` Features

The `dhcpconfig` utility supports options that enable you to configure and unconfigure a DHCP server, as well as convert to a new data store and import/export data to and from other DHCP servers. When you use the `dhcpconfig` utility to configure a DHCP server, the utility obtains information from the system files discussed in "Updating System Files and Netmask Tables" on page 318. You cannot view and confirm the information obtained from system files as you can with DHCP Manager. So, it is important that the system files be updated before you run

`dhcpconfig`. You can also use command-line options to override the values `dhcpconfig` would obtain by default from system files. The `dhcpconfig` command can be used in scripts. See the `dhcpconfig`(1M) man page for more information.

## Comparison of DHCP Manager and `dhcpconfig`

The following table summarizes the differences between the two server configuration tools.

**TABLE 13–4** Comparison of DHCP Manager and the `dhcpconfig` Command

| Feature | DHCP Manager | `dhcpconfig` With Options |
|---|---|---|
| Network information that is gathered from system. | Enables you to view the information gathered from system files, and to change it if needed. | You can specify the network information with command-line options. |
| Speed of configuration. | Speeds the configuration process by omitting prompts for nonessential server options, using default values instead. You can change nonessential options after initial configuration. | Fastest configuration process, but you might need to specify values for many options. |

Chapter 14, "Configuring the DHCP Service (Tasks)," includes procedures you can use to configure your server with either DHCP Manager or the `dhcpconfig` utility.

◆ ◆ ◆ CHAPTER 14

# Configuring the DHCP Service (Tasks)

When you configure the DHCP service on your network, you configure and start the first DHCP server. Other DHCP servers can be added later and can access the same data from a shared location if the data store supports shared data. This chapter describes tasks that enable you to configure the DHCP server and place networks and their associated IP addresses under DHCP management. This chapter also explains how to unconfigure a DHCP server.

Each task includes a procedure to help you perform the task in DHCP Manager and a procedure for the equivalent task with the `dhcpconfig` utility. This chapter contains the following information:

- "Configuring and Unconfiguring a DHCP Server Using DHCP Manager" on page 329
- "Configuring and Unconfiguring a DHCP Server Using `dhcpconfig` Commands" on page 336

If you experience problems configuring the DHCP service, see Chapter 17, "Troubleshooting DHCP (Reference)."

After you configure the DHCP service, see Chapter 15, "Administering DHCP (Tasks)," for information about managing the DHCP service.

## Configuring and Unconfiguring a DHCP Server Using DHCP Manager

This section includes procedures to help you configure and unconfigure a DHCP server with DHCP Manager. Note that you must be running an X Window system such as CDE or GNOME to use DHCP Manager.

DHCP Manager can be run as superuser with the `/usr/sadm/admin/bin/dhcpmgr` command. See "About DHCP Manager" on page 340 for general information about the utility. See "How to Start and Stop the DHCP Service (DHCP Manager)" on page 344 for more detailed information about running DHCP Manager.

When you run DHCP Manager on a server that is not configured for DHCP, the following screen is displayed. You can specify whether you want to configure a DHCP server or a BOOTP relay agent.

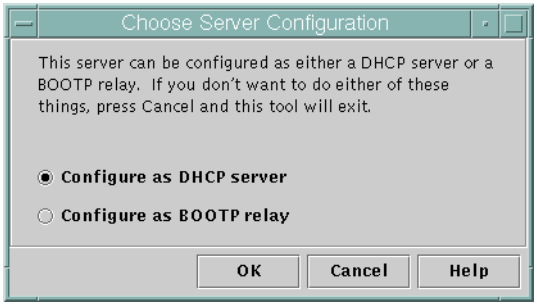



**FIGURE 14–1** Choose Server Configuration Dialog Box in DHCP Manager

# Configuring DHCP Servers

When you configure a DHCP server, DHCP Manager starts the DHCP Configuration Wizard, which prompts you for information that is needed to configure the server. The initial screen of the wizard is shown in the following figure.

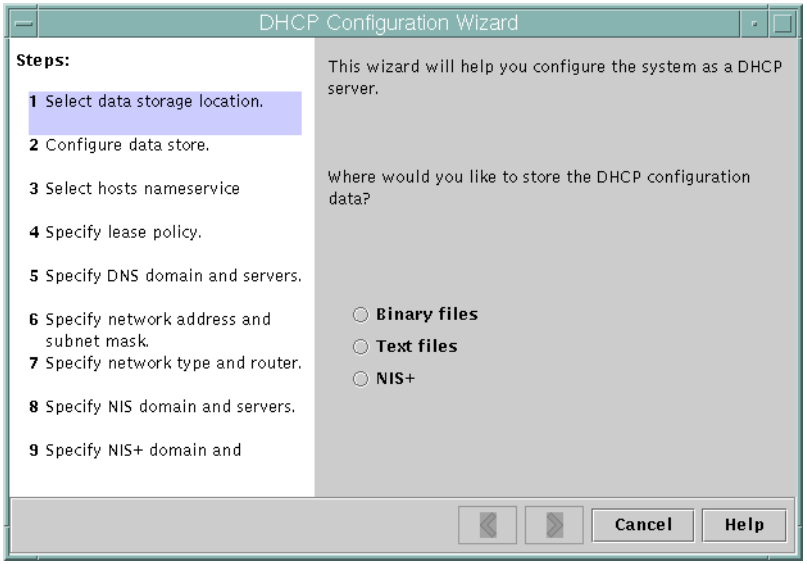



**FIGURE 14–2** DHCP Configuration Wizard's Initial Screen

When you finish answering the wizard prompts, DHCP Manager creates the items that are listed in the following table.

**TABLE 14–1** Items Created During DHCP Server Configuration

| Item | Description | Contents |
|---|---|---|
| Service configuration file, `/etc/inet/dhcpsvc.conf` | Records keywords and values for server configuration options. | Data store type and location, and options that are used with `in.dhcpd` to start the DHCP daemon when the system boots. Do not edit this file manually. You must use `dhcpmgr` or `dhcpconfig` to modify DHCP configuration information. |
| `dhcptab` table | DHCP Manager creates a `dhcptab` table if the table does not already exist. | Macros and options with assigned values. |
| Locale macro (optional), which is named `Locale` | Contains the local time zone's offset in seconds from Universal time (UTC). | `UTCoffst` option with assigned number of seconds. |
| Server macro, which is named to match the server's node name | Contains options whose values are determined by input from the administrator who configured the DHCP server. Options apply to all clients that use addresses owned by the server. | The `Locale` macro, plus the following options:<br>■ `Timeserv`, set to point to the server's primary IP address.<br>■ `LeaseTim`, set to the number of seconds for the leases.<br>■ `LeaseNeg`, if you selected negotiable leases.<br>■ `DNSdmain` and `DNSserv`, if DNS is configured.<br>■ `Hostname`, which *must not* be assigned a value. The presence of this option indicates that the host name must be obtained from the name service. |
| Network address macro, whose name is the same as the network address of client's network | Contains options whose values are determined by input from the administrator who configured the DHCP server. Options apply to all clients that reside on the network specified by the macro name. | The following options:<br>■ `Subnet`, set to the subnet mask for the local subnet<br>■ `Router`, set to the IP address of a router, or `RDiscvyF`, to cause the client to use router discovery<br>■ `Broadcst`, set to the broadcast IP address. This option is present only if the network is not a Point-to-Point network.<br>■ `MTU`, for the maximum transmission unit<br>■ `NISdmain` and `NISservs`, if NIS is configured<br>■ `NIS+dom` and `NIS+serv`, if NIS+ is configured |
| Network table for the network | An empty table is created until you create IP addresses for the network. | No content until you add IP addresses. |

## ▼ How to Configure a DHCP Server (DHCP Manager)

**Before You Begin**

Make sure that you have read Chapter 13, "Planning for DHCP Service (Tasks)," before you configure your DHCP server. In particular, you should use the guidelines in "Making Decisions for Your DHCP Server Configuration (Task Map)" on page 319 to help you perform the following tasks:

- Select the system that you want to use as a DHCP server.
- Make decisions about your data store, lease policy, and router information.

1. **Become superuser on the server system.**

2. **Start DHCP Manager.**

   ```
   #/usr/sadm/admin/bin/dhcpmgr &
   ```

3. **Choose the option Configure as DHCP Server.**

   The DHCP Configuration Wizard starts, to help you configure your server.

4. **Select options, or type requested information, based on the decisions you made in the planning phase.**

   If you have difficulty, click Help in the wizard window to open your web browser and display help for the DHCP Configuration Wizard.

5. **Click Finish to complete the server configuration when you have finished specifying the requested information.**

6. **At the Start Address Wizard prompt, click Yes to configure IP addresses for the server.**

   The Add Addresses to Network wizard enables you to specify which addresses to place under the control of DHCP.

7. **Answer the prompts according to decisions you made in the planning phase.**

   See "Making Decisions for IP Address Management (Task Map)" on page 322 for more information. If you have difficulty, click Help in the wizard window to open your web browser and display help for the Add Addresses to Network wizard.

8. **Review your selections, and then click Finish to add the IP addresses to the network table.**

   The network table is updated with records for each address in the range you specified.

**See Also**

You can add more networks to the DHCP server with the Network Wizard, as explained in "Adding DHCP Networks" on page 360.

## Configuring BOOTP Relay Agents

When you configure a BOOTP relay agent, DHCP Manager takes the following actions:

- Prompts you for the IP address for one or more DHCP servers to which requests should be relayed
- Stores settings needed for BOOTP relay service

The following figure shows the screen displayed when you choose to configure a BOOTP relay agent.

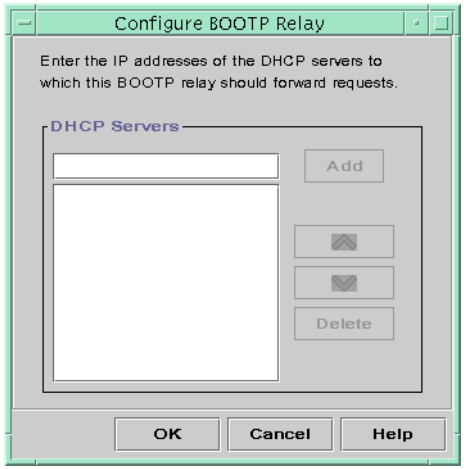



**FIGURE 14–3** Configure BOOTP Relay Dialog Box in DHCP Manager

## ▼ How to Configure a BOOTP Relay Agent (DHCP Manager)

**Before You Begin** Make sure that you have read Chapter 13, "Planning for DHCP Service (Tasks)," before you configure your BOOTP relay agent. In particular, you should see "Selecting a Host to Run the DHCP Service" on page 320 for help in selecting the system to use.

1. **Become superuser on the server system.**

2. **Start the DHCP Manager.**

   ```
   #/usr/sadm/admin/bin/dhcpmgr &
   ```

   If the system has not been configured as a DHCP server or BOOTP relay agent, the DHCP Configuration Wizard starts. If the system has already been configured as a DHCP server, you must first unconfigure the server. See "Unconfiguring DHCP Servers and BOOTP Relay Agents" on page 334.

3 **Select Configure as BOOTP Relay.**

The Configure BOOTP Relay dialog box opens.

4 **Type the IP address or host name of one or more DHCP servers, and click Add.**

The specified DHCP servers must be configured to handle BOOTP or DHCP requests received by this BOOTP relay agent.

5 **Click OK to exit the dialog box.**

Notice that DHCP Manager offers only the File menu to exit the application and the Service menu to manage the server. The disabled menu options are useful only on a DHCP server.

# Unconfiguring DHCP Servers and BOOTP Relay Agents

When you unconfigure a DHCP server or a BOOTP relay agent, DHCP Manager takes the following actions:

- Stops the DHCP daemon (`in.dhpcd`) process
- Removes the `/etc/inet/dhcpsvc.conf` file, which records information about daemon startup and the data store location

The following figure shows the screen that is displayed when you choose to unconfigure a DHCP server.

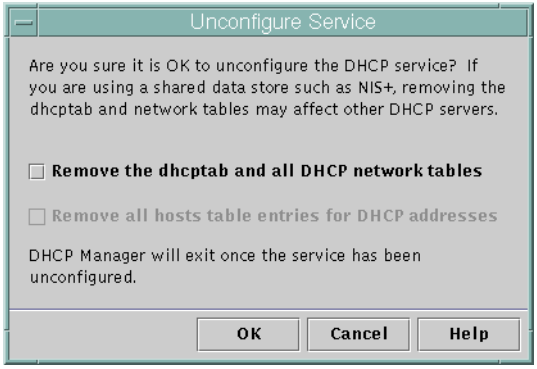



**FIGURE 14–4** Unconfigure Service Dialog Box in DHCP Manager

# DHCP Data on an Unconfigured Server

When you unconfigure a DHCP server, you must decide what to do with the `dhcptab` table and the DHCP network tables. If the data is shared among servers, you should not remove the `dhcptab` and DHCP network tables. If the tables are removed, DHCP would become unusable across your network. Data can be shared through NIS+ or on exported local file systems. The file `/etc/inet/dhcpsvc.conf` records the data store used and its location.

You can unconfigure a DHCP server but leave the data intact by not selecting any of the options to remove data. If you unconfigure the server and leave the data intact, you disable the DHCP server.

If you want another DHCP server to take ownership of the IP addresses, you must move the DHCP data to the other DHCP server. You must move the data before you unconfigure the current server. See "Moving Configuration Data Between DHCP Servers (Task Map)" on page 410 for more information.

If you are certain you want to remove the data, you can select an option to remove the `dhcptab` and network tables. If you had generated client names for the DHCP addresses, you can also elect to remove those entries from the hosts table. Client name entries can be removed from DNS, `/etc/inet/hosts`, or NIS+.

Before you unconfigure a BOOTP relay agent, be sure that no clients rely on this agent to forward requests to a DHCP server.

## How to Unconfigure a DHCP Server or a BOOTP Relay Agent (DHCP Manager)

1. **Become superuser.**

2. **Start DHCP Manager.**

   ```
   #/usr/sadm/admin/bin/dhcpmgr &
   ```

3. **From the Service menu, choose Unconfigure.**

   The Unconfigure Service dialog box is displayed. If the server is a BOOTP relay agent, the dialog box enables you to confirm your intention to unconfigure the relay agent. If the server is a DHCP server, you must decide what to do with the DHCP data and make selections in the dialog box. See Figure 14–4.

4. **(Optional) Select options to remove data.**

   If the server uses shared data through NIS+ or in files shared through NFS, do not select any options to remove the data. If the server does not use shared data, select one option or both options to remove the data.

   See "DHCP Data on an Unconfigured Server" on page 334 for more information about removing data.

5. **Click OK to unconfigure the server.**

   The Unconfigure Service dialog box and DHCP Manager are closed.

# Configuring and Unconfiguring a DHCP Server Using dhcpconfig Commands

This section includes procedures to help you configure and unconfigure a DHCP server or a BOOTP relay agent by using dhcpconfig with command-line options.

## ▼ How to Configure a DHCP Server (dhcpconfig -D)

**Before You Begin** Make sure that you have read Chapter 13, "Planning for DHCP Service (Tasks)," before you configure your DHCP server. In particular, you should use the guidelines in "Making Decisions for Your DHCP Server Configuration (Task Map)" on page 319 to help you perform the following tasks:

- Select the system that you want to use as a DHCP server.
- Make decisions about your data store, lease policy, and router information.

**1 Log in to the system on which you want to configure the DHCP server.**

**2 Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**3 Configure the DHCP server by typing a command of the following format:**

```
#/usr/sbin/dhcpconfig -D -r datastore -p location
```

*datastore* is one of the following: SUNWfiles, SUNWbinfiles, or SUNWnisplus.

*location* is the data-store-dependent location where you want to store the DHCP data. For SUNWfiles and SUNWbinfiles, the location must be an absolute path name. For SUNWnisplus, the location must be a fully specified NIS+ directory.

For example, you might type a command similar to the following:

```
dhcpconfig -D -r SUNWbinfiles -p /var/dhcp
```

The dhcpconfig utility uses the host's system files and network files to determine values used to configure the DHCP server. See the dhcpconfig(1M) man page for information about additional options to the dhcpconfig command that enable you to override the default values.

4 **Add one or more networks to the DHCP service.**

See "How to Add a DHCP Network (dhcpconfig)" on page 362 for the procedure to add a network.

## How to Configure a BOOTP Relay Agent (dhcpconfig -R)

**Before You Begin** Select the system that you want to use as a BOOTP relay agent, using the requirements listed in "Selecting a Host to Run the DHCP Service" on page 320.

1 **Log in to the server that you want to configure as a BOOTP relay agent.**

2 **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

3 **Configure the BOOTP relay agent by typing a command of the following format:**

```
# /usr/sbin/dhcpconfig -R server-addresses
```

Specify one or more IP addresses of DHCP servers to which you want requests to be forwarded. If you specify more than one address, separate the addresses with commas.

For example, you might type a command similar to the following:

```
/usr/sbin/dhcpconfig -R 192.168.1.18,192.168.42.132
```

## How to Unconfigure a DHCP Server or a BOOTP Relay Agent (dhcpconfig -U)

1 **Log in to the DHCP server or the BOOTP relay agent system that you want to unconfigure.**

2 **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**3 Unconfigure the DHCP server or the BOOTP relay agent:**

```
# /usr/sbin/dhcpconfig -U
```

If the server does not use shared data, you can also use the `-x` option to remove the `dhcptab` and network tables. If the server uses shared data, do not use the `-x` option. The `-h` option can be used to remove host names from the host table. See the `dhcpconfig`(1M) man page for more information about `dhcpconfig` options.

See "DHCP Data on an Unconfigured Server" on page 334 for more information about removing data.

◆ ◆ ◆ CHAPTER 15

# Administering DHCP (Tasks)

This chapter describes tasks that you might find useful when you administer the Solaris DHCP service. The chapter includes tasks for the server, BOOTP relay agent, and client. Each task includes a procedure to help you perform the task in DHCP Manager and a procedure for the equivalent task with DHCP command-line utilities. DHCP command-line utilities are more fully documented in man pages.

You should have already completed the initial configuration of your DHCP service and initial network before you use this chapter. Chapter 14, "Configuring the DHCP Service (Tasks)," discusses DHCP configuration.

This chapter contains the following information:


- "About DHCP Manager" on page 340
- "Setting Up User Access to DHCP Commands" on page 343
- "Starting and Stopping the DHCP Service" on page 343
- "DHCP Service and the Service Management Facility" on page 346
- "Modifying DHCP Service Options (Task Map)" on page 346
- "Adding, Modifying, and Removing DHCP Networks (Task Map)" on page 358
- "Supporting BOOTP Clients With the DHCP Service (Task Map)" on page 367
- "Working With IP Addresses in the DHCP Service (Task Map)" on page 370
- "Working With DHCP Macros (Task Map)" on page 385
- "Working With DHCP Options (Task Map)" on page 396
- "Supporting Solaris Network Installation With the DHCP Service" on page 405
- "Supporting Remote Boot and Diskless Boot Clients (Task Map)" on page 406
- "Setting Up DHCP Clients to Receive Information Only (Task Map)" on page 407
- "Converting to a New DHCP Data Store" on page 408
- "Moving Configuration Data Between DHCP Servers (Task Map)" on page 410

# About DHCP Manager

DHCP Manager is a graphical user interface (GUI) tool that you can use to perform administration tasks on the DHCP service.

## DHCP Manager Window

The DHCP Manager window's appearance depends on how the DHCP server is configured on the system on which DHCP Manager is running.

DHCP Manager uses a tab-based window when the system is configured as a DHCP server. You select a tab for the type of information you want to work with. DHCP Manager features the following tabs:

- **Addresses**tab – Lists all networks and IP addresses placed under DHCP management. From the Addresses tab, you can work with networks and IP addresses. You can add or delete items individually or in blocks. You can also modify the properties of individual networks or IP addresses or simultaneously make the same property modifications for a block of addresses. When you start DHCP Manager, the Addresses tab opens first.
- **Macros** tab – Lists all available macros in the DHCP configuration table (`dhcptab`) and the options contained within the macros. From the Macros tab, you can create or delete macros. You can also modify macros by adding options and providing values for the options.
- **Options** tab – Lists all options that have been defined for this DHCP server. Options that are listed on this tab are not the standard options defined in the DHCP protocol. The options are extensions to the standard options, and have a class of Extended, Vendor, or Site. Standard options cannot be changed in any way so those options are not listed here.

The following figure shows how the DHCP Manager window might look when you start DHCP Manager on a DHCP server.

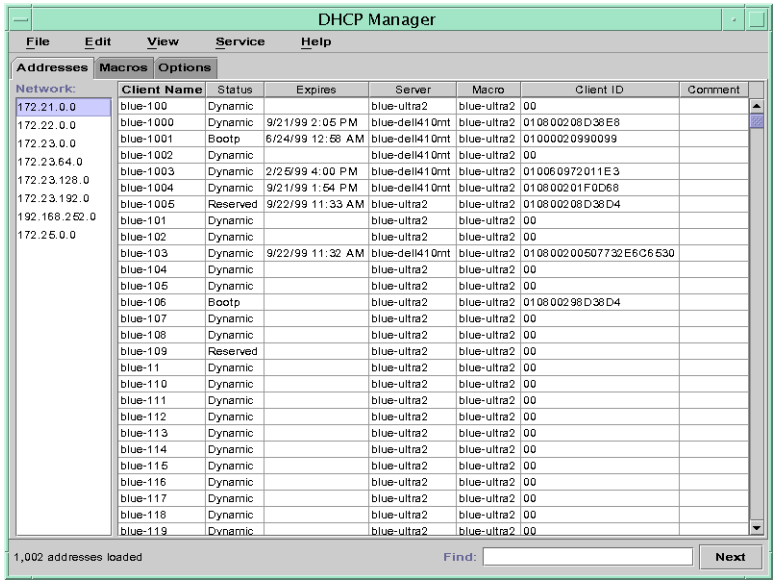



**FIGURE 15–1** DHCP Manager on a DHCP Server System

When the server is configured as a BOOTP relay agent, the DHCP Manager window does not show these tabs. The BOOTP relay agent does not need the same information. You can only modify the BOOTP relay agent's properties and stop or start the DHCP daemon with DHCP Manager. The following figure shows how DHCP Manager might look on a system that is configured as a BOOTP relay agent.

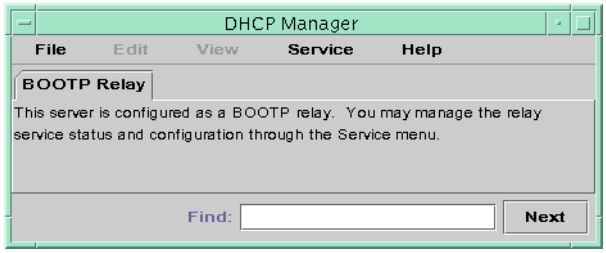



**FIGURE 15–2** DHCP Manager on a BOOTP Relay Agent

# DHCP Manager Menus

DHCP Manager menus include the following items:

- **File** – Exit DHCP Manager.
- **Edit** – Perform management tasks for networks, addresses, macros, and options.
- **View** – Change the look of the tab currently selected.
- **Service** – Manage the DHCP daemon and data store.

- **Help** – Open your web browser and display help for DHCP Manager.

When DHCP Manager runs on a BOOTP relay agent, the Edit and View menus are disabled.

All DHCP management tasks are accomplished through the Edit and Service menus.

You use the commands in the Edit menu to create, delete, and modify items in the selected tab. Items can include networks, addresses, macros, and options. When the Addresses tab is selected, the Edit menu also lists wizards. Wizards are sets of dialogs that help you create networks and multiple IP addresses.

The Service menu lists commands that enable you to manage the DHCP daemon. From the Service menu, you can perform the following tasks:

- Start and stop the DHCP daemon.
- Enable and disable the DHCP daemon.
- Modify the server configuration.
- Unconfigure the server.
- Convert the data store.
- Export and import data on the server.

## Starting and Stopping DHCP Manager

You must run DHCP Manager on a DHCP server system as superuser. If you must run DHCP Manager remotely, you can send the display to your system by using the X Window remote display feature.

### ▼ How to Start and Stop DHCP Manager

1. **Become superuser on the DHCP server system.**
2. **(Optional) If you are logged in to the DHCP server system remotely, display DHCP Manager on your local system as follows.**

   a. **Type the following on the local system:**

      ```
      # xhost +server-name
      ```

   b. **Type the following on the remote DHCP server system:**

      ```
      # DISPLAY=local-hostname;export DISPLAY
      ```

3. **Start DHCP Manager.**

   ```
   # /usr/sadm/admin/bin/dhcpmgr &
   ```

The DHCP Manager window opens. If the server is configured as a DHCP server, the window displays the Addresses tab. If the server is configured as a BOOTP relay agent, the window displays with no tabs.

4 **To stop DHCP Manager, choose Exit from the File menu.**

The DHCP Manager window closes.

# Setting Up User Access to DHCP Commands

By default, only root or superuser can execute `dhcpconfig`, `dhtadm`, and `pntadm` commands. If you want non root users to use the commands, you can set up role-based access control (RBAC) for those commands.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

You might also find the following man pages helpful: `rbac`(5), `exec_attr`(4), and `user_attr`(4).

The following procedure explains how to assign the DHCP Management profile, which enables the user to execute the DHCP commands.

## ▼ How to Grant Users Access to DHCP Commands

1 **Become superuser on the DHCP server system.**

2 **Edit the file `/etc/user_attr` to add an entry of the following form. Add one entry for each user or role that should manage the DHCP service.**

*username*`::::type=normal;profiles=DHCP Management`

For example, for user `ram`, you would add the following entry:

```
ram::::type=normal;profiles=DHCP Management
```

# Starting and Stopping the DHCP Service

This section describes starting and stopping the DHCP service by using DHCP Manager and the `dhcpconfig` command. The DHCP service can also be started and stopped by using the Service Management Facility (SMF) commands. See "DHCP Service and the Service Management Facility" on page 346 for more information about using SMF commands with the DHCP service.

Starting and stopping the DHCP service encompasses several degrees of action you can take to affect the operation of the DHCP daemon. You must understand what each action means in order to select the correct procedure to obtain the result that you want. The terms for the actions are as follows:

- **Start, stop, and restart commands** affect the daemon only for the current session. For example, if you stop the DHCP service, the daemon terminates but restarts when you reboot the system. DHCP data tables are not affected when you stop the service. You can use DHCP Manager or SMF commands to temporarily start and stop the DHCP service without enabling and disabling the service.
- **Enable and disable commands** affect the daemon for current and future sessions. If you disable the DHCP service, the currently running daemon terminates and does not start when you reboot the server. You must enable the DHCP daemon for automatic startup at system boot to occur. DHCP data tables are not affected. You can use DHCP Manager, the `dhcpconfig` command, or SMF commands to enable and disable the DHCP service.
- The **unconfigure command** shuts down the daemon, prevents the daemon from starting on system reboot, and enables you to remove the DHCP data tables. You can use DHCP Manager or the `dhcpconfig` command to unconfigure the DHCP service. Unconfiguration is described in Chapter 14, "Configuring the DHCP Service (Tasks)."

---

**Note –** If a server has multiple network interfaces but you do not want to provide DHCP services on all the networks, see "Specifying Network Interfaces for DHCP Monitoring" on page 358.

---

The following procedures help you start, stop, enable, and disable the DHCP service.

## ▼ How to Start and Stop the DHCP Service (DHCP Manager)

1. **Become superuser on the DHCP server system.**
2. **Start DHCP Manager.**

   ```
   # /usr/sadm/admin/bin/dhcpmgr &
   ```

3. **Select one of the following:**

   - **Choose Start from the Service menu to start the DHCP service.**
   - **Choose Stop from the Service menu to stop the DHCP service.**

     The DHCP daemon stops until it is restarted, or the system reboots.
   - **Choose Restart from the Service menu to stop and immediately restart the DHCP service.**

## ▼ How to Enable and Disable the DHCP Service (DHCP Manager)

- **In DHCP Manager, choose one of the following:**
  - **Choose Enable from the Service menu to configure the DHCP daemon for automatic startup when the system boots.**

    The DHCP service starts immediately when it is enabled.
  - **Choose Disable from the Service menu to prevent the DHCP daemon from automatically starting when the system boots.**

    The DHCP service immediately stops when it is disabled.

## ▼ How to Enable and Disable the DHCP Service (`dhcpconfig -S`)

1. **Log in to the DHCP server system.**
2. **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

   For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.
3. **Choose one of the following:**
   - **To enable the DHCP service, type the following command:**

     ```
     # /usr/sbin/dhcpconfig -S -e
     ```
   - **To disable the DHCP service, type the following command:**

     ```
     # /usr/sbin/dhcpconfig -S -d
     ```

# DHCP Service and the Service Management Facility

The Service Management Facility (SMF) is described in Chapter 15, "Managing Services (Overview)," in *System Administration Guide: Basic Administration*. The SMF `svcadm` command can be used to enable and start the DHCP server, and disable and stop the DHCP server. However, you cannot use SMF commands to modify the DHCP service options that the DHCP tools allow you to set. In particular, service options that are stored in the `/etc/dhcp/dhcpsvc.conf` file cannot be set by using the SMF tools.

The following table maps DHCP commands to the equivalent SMF commands.

**TABLE 15–1** SMF Commands For DHCP Server Tasks

| Task | DHCP Command | SMF Command |
|---|---|---|
| Enable DHCP service | `dhcpconfig -S -e` | `svcadm enable svc:/network/dhcp-server` |
| Disable DHCP service | `dhcpconfig -S -d` | `svcadm disable svc:/network/dhcp-server` |
| Start DHCP service for current session only | None | `svcadm enable -t svc:/network/dhcp-server` |
| Stop DHCP service for current session | None | `svcadm disable -t svc:/network/dhcp-server` |
| Restart DHCP service | `dhcpconfig -S -r` | `svcadm restart svc:/network/dhcp-server` |

# Modifying DHCP Service Options (Task Map)

You can change values for some additional features of the DHCP service, which might not have been offered during the initial configuration with DHCP Manager. To change service options, you can use the Modify Service Options dialog box in DHCP Manager. Or you can specify options with the `dhcpconfig` command.

The following task map shows the tasks related to service options and the procedures to use.

| Task | Description | For Instructions |
|---|---|---|
| Change logging options. | Enable or disable logging, and select a `syslog` facility to use for logging DHCP transactions. | "How to Generate Verbose DHCP Log Messages (DHCP Manager)" on page 349<br>"How to Generate Verbose DHCP Log Messages (Command Line)" on page 350<br>"How to Enable and Disable DHCP Transaction Logging (DHCP Manager)" on page 350<br>"How to Enable and Disable DHCP Transaction Logging (Command Line)" on page 351<br>"How to Log DHCP Transactions to a Separate `syslog` File" on page 352 |
| Change DNS update options. | Enable or disable server's capability to dynamically add DNS entries for clients that supply a host name. Determine the maximum time the server should spend attempting to update DNS. | "How to Enable Dynamic DNS Updating for DHCP Clients" on page 353 |
| Enable or disable duplicate IP address detection. | Enable or disable the DHCP server's capability to determine that an IP address is not already in use before offering the address to a client. | "How to Customize DHCP Performance Options (DHCP Manager)" on page 356<br>"How to Customize DHCP Performance Options (Command Line)" on page 357 |
| Change options for the DHCP server's reading of configuration information. | Enable or disable the automatic reading of `dhcptab` at specified intervals, or change the interval between reads. | "How to Customize DHCP Performance Options (DHCP Manager)" on page 356<br>"How to Customize DHCP Performance Options (Command Line)" on page 357 |
| Change the number of relay agent hops. | Increase or decrease the number of networks a request can travel through before being dropped by the DHCP daemon. | "How to Customize DHCP Performance Options (DHCP Manager)" on page 356<br>"How to Customize DHCP Performance Options (Command Line)" on page 357 |
| Change the length of time an IP address offer is cached. | Increase or decrease the number of seconds that the DHCP service reserves an offered IP address before offering the address to a new client. | "How to Customize DHCP Performance Options (DHCP Manager)" on page 356<br>"How to Customize DHCP Performance Options (Command Line)" on page 357 |

The following figure shows DHCP Manager's Modify Service Options dialog box.

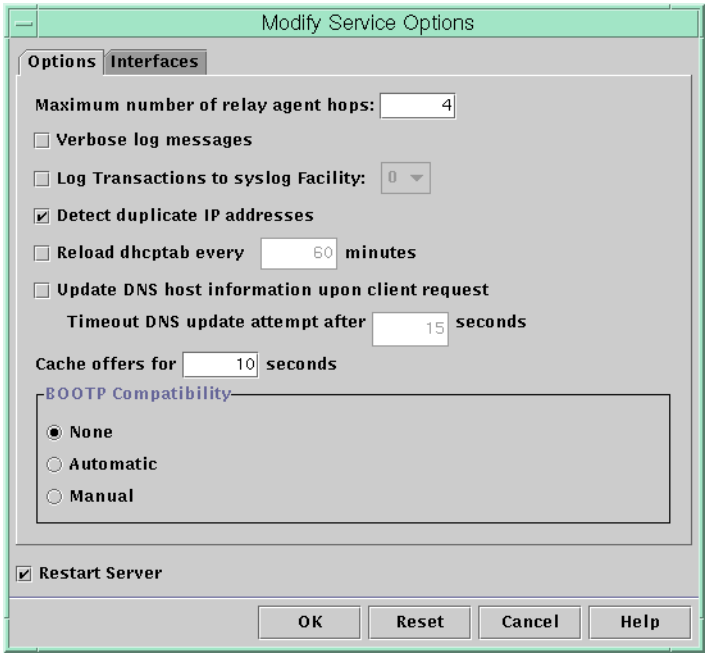



**FIGURE 15–3** Modify Service Options Dialog Box in DHCP Manager

# Changing DHCP Logging Options

The DHCP service can log DHCP service messages and DHCP transactions to `syslog`. See the `syslogd`(1M) and `syslog.conf`(4) man pages for more information about `syslog`.

DHCP service messages logged to `syslog` include the following:

- Error messages, which notify you of conditions that prevent the DHCP service from fulfilling a request by a client or by you.
- Warnings and notices, which notify you of conditions that are abnormal, but do not prevent the DHCP service from fulfilling a request.

You can increase the amount of information that is reported by using the verbose option for the DHCP daemon. Verbose message output can help you troubleshoot DHCP problems. See "How to Generate Verbose DHCP Log Messages (DHCP Manager)" on page 349.

Another useful troubleshooting technique is transaction logging. Transactions provide information about every interchange between a DHCP server or BOOTP relay and clients. DHCP transactions include the following message types:

- ASSIGN – IP address assignment
- ACK – Server acknowledges that the client accepts the offered IP address, and sends configuration parameters
- EXTEND – Lease extension
- RELEASE – IP address release
- DECLINE – Client is declining address assignment
- INFORM – Client is requesting network configuration parameters but not an IP address
- NAK – Server does not acknowledge a client's request to use a previously used IP address
- ICMP_ECHO – Server detects potential IP address is already in use by another host

BOOTP relay transactions include the following message types:

- RELAY-CLNT – Message is being relayed from the DHCP client to a DHCP server
- RELAY–SRVR – Message is being relayed from the DHCP server to the DHCP client

DHCP transaction logging is disabled by default. When enabled, DHCP transaction logging uses the `local0` facility in `syslog` by default. DHCP transaction messages are generated with a `syslog` severity level of *notice*. This security level causes DHCP transactions to be logged to the file where other system notices are logged. However, because the `local` facility is used, the DHCP transaction messages can be logged separately from other notices. To log the transaction messages separately, you must edit the `syslog.conf` file to specify a separate log file. See the `syslog.conf`(4) man page for more information about the `syslog.conf` file.

You can disable or enable transaction logging, and you can specify a different `syslog` facility, from `local0` through `local7`, as explained in "How to Enable and Disable DHCP Transaction Logging (DHCP Manager)" on page 350. In the server system's `syslog.conf` file, you can also instruct `syslogd` to store the DHCP transaction messages in a separate file. See "How to Log DHCP Transactions to a Separate `syslog` File" on page 352 for more information.

## ▼ How to Generate Verbose DHCP Log Messages (DHCP Manager)

**1 In DHCP Manager, choose Modify from the Service menu.**

See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

The Modify Service Options dialog box opens and displays the Options tab. See Figure 15–3.

2 **Select Verbose Log Messages.**

3 **Select Restart Server.**

The Restart Server option is near the bottom of the dialog box.

4 **Click OK.**

The daemon runs in verbose mode for this session and each subsequent session until you reset this option. Verbose mode can reduce daemon efficiency because of the time that is taken to display messages.

## ▼ How to Generate Verbose DHCP Log Messages (Command Line)

1 **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Type the following command to set verbose mode:**

```
# /usr/sbin/dhcpconfig -P VERBOSE=true
```

The next time the DHCP server starts, the server runs in verbose mode until you turn off verbose mode.

To turn off verbose mode, type the following command:

```
# /usr/sbin/dhcpconfig -P VERBOSE=
```

This command sets the VERBOSE keyword to no value, which causes the keyword to be removed from the server's configuration file.

Verbose mode can reduce daemon efficiency because of the time that is taken to display messages.

## ▼ How to Enable and Disable DHCP Transaction Logging (DHCP Manager)

This procedure enables and disables transaction logging for all subsequent DHCP server sessions.

1. **In DHCP Manager, choose Modify from the Service menu.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Select Log Transactions to Syslog Facility.**

   To disable transaction logging, deselect this option.

3. **(Optional) Select a local facility from 0 to 7 to use for logging DHCP transactions.**

   By default, DHCP transactions are logged to the location where system notices are logged, which depends on how `syslogd` is configured. If you want the DHCP transactions to be logged to a file separate from other system notices, see "How to Log DHCP Transactions to a Separate `syslog` File" on page 352.

   Message files can quickly become very large when transaction logging is enabled.

4. **Select Restart Server.**

5. **Click OK.**

   The daemon logs transactions to the selected `syslog` facility for this session and each subsequent session until you disable logging.

## ▼ How to Enable and Disable DHCP Transaction Logging (Command Line)

1. **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

   For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Choose one of the following steps:**

   - **To enable DHCP transaction logging, type the following command:**

     ```
     # /usr/sbin/dhcpconfig -P LOGGING_FACILITY=syslog-local-facility
     ```

     *syslog-local-facility* is a number from 0 through 7. If you omit this option, 0 is used.

     By default, DHCP transactions are logged to the location where system notices are logged, which depends on how `syslogd` is configured. If you want the DHCP transactions to be logged to a file separate from other system notices, see "How to Log DHCP Transactions to a Separate `syslog` File" on page 352.

Message files can quickly become very large when transaction logging is enabled.

- **To disable DHCP transaction logging, type the following command:**

  ```
  # /usr/sbin/dhcpconfig -P LOGGING_FACILITY=
  ```

  Note that you supply no value for the parameter.

## How to Log DHCP Transactions to a Separate `syslog` File

1. **Become superuser or assume an equivalent role on the DHCP server system.**

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

   A role that is assigned to the DHCP Management profile might not be sufficient for this task. The role must have permission to edit `syslog` files.

2. **Edit the `/etc/syslog.conf` file on the server system to add a line of the following format:**

   `local`*n*`.notice` *path-to-logfile*

   *n* is the `syslog` facility number you specified for transaction logging, and *path-to-logfile* is the complete path to the file to use for logging transactions.

   For example, you might add the following line:

   ```
   local0.notice /var/log/dhcpsrvc
   ```

   See the `syslog.conf`(4) man page for more information about the `syslog.conf` file.

# Enabling Dynamic DNS Updates by a DHCP Server

DNS provides name-to-address and address-to-name services for the Internet. Once a DNS mapping is made, a system can be reached through its host name or its IP address. The system is also reachable from outside its domain.

The DHCP service can use DNS in two ways:

- The DHCP server can look up the host name that is mapped to an IP address that the server is assigning to the client. The server then returns the client's host name along with the client's other configuration information.
- The DHCP server can attempt to make a DNS mapping on a client's behalf, if the DHCP server is configured to update DNS. The client can supply its own host name when requesting DHCP service. If configured to make DNS updates, the DHCP server attempts to update DNS with the client's suggested host name. If the DNS update is successful, the DHCP server returns the requested host name to the client. If the DNS update is not successful, the DHCP server returns a different host name to the client.

You can enable the DHCP service to update the DNS service for DHCP clients that supply their own host names. For the DNS update feature to work, the DNS server, the DHCP server, and the DHCP client must be set up correctly. In addition, the requested host name must not be in use by another system in the domain.

The DHCP server's DNS update feature works if the following statements are true:

- The DNS server supports RFC 2136.
- The DNS software is based on BIND v8.2.2, patch level 5 or later, whether on the DHCP server system or the DNS server system.
- The DNS server is configured to accept dynamic DNS updates from the DHCP server.
- The DHCP server is configured to make dynamic DNS updates.
- DNS support is configured for the DHCP client's network on the DHCP server.
- The DHCP client is configured to supply a requested host name in its DHCP request message.
- The requested host name corresponds to a DHCP-owned address. The host name could also have no corresponding address.

## ▼ How to Enable Dynamic DNS Updating for DHCP Clients

**Note –** Be aware that dynamic DNS updates are a *security risk*.

By default, the Solaris DNS daemon (`in.named`) does not allow dynamic updates. Authorization for dynamic DNS updates is granted in the `named.conf` configuration file on the DNS server system. No other security is provided. You must carefully weigh the convenience of this facility for users against the security risk created when you enable dynamic DNS updates.

1 **On the DNS server, edit the `/etc/named.conf` file as superuser.**

2 **Find the `zone` section for the appropriate domain in the `named.conf` file.**

3 **Add the DHCP server's IP addresses to the `allow-update` keyword.**

If the `allow-update` keyword does not exist, insert the keyword.

For example, if the DHCP server resides at addresses `10.0.0.1` and `10.0.0.2`, a `named.conf` file for the `dhcp.domain.com` zone should be modified as follows:

```
zone "dhcp.domain.com" in {
             type master;
             file "db.dhcp";
             allow-update { 10.0.0.1; 10.0.0.2; };
};

zone "10.IN-ADDR.ARPA" in {
             type master;
             file "db.10";
             allow-update { 10.0.0.1; 10.0.0.2; };
};
```

Note that `allow-update` for both zones must be enabled to allow the DHCP server to update both A and PTR records on the DNS server.

4 **On the DHCP server, start DHCP Manager.**

```
# /usr/sadm/admin/bin/dhcpmgr &
```

See "How to Start and Stop DHCP Manager" on page 342 for more detailed information.

5 **Choose Modify from the Service menu.**

The Modify Service Options dialog box opens.

6 **Select Update DNS Host Information Upon Client Request.**

7 **Specify the number of seconds to wait for a response from the DNS server before timing out, then click OK.**

The default value of 15 seconds should be adequate. If you have time out problems, you can increase the value later.

8 **Click the Macros tab, and ensure that the correct DNS domain is specified.**

The `DNSdmain` option must be passed with the correct domain name to any client that expects dynamic DNS update support. By default, `DNSdmain` is specified in the server macro, which is used as the configuration macro bound to each IP address.

**9 Set up the DHCP client to specify its host name when requesting DHCP service.**

If you use the Solaris DHCP client, see "How to Enable a Solaris DHCPv4 Client to Request a Specific Host Name" on page 430. If your client is not a Solaris DHCP client, see the documentation for your DHCP client for information about how to specify a host name.

# Client Host Name Registration

If you let the DHCP server generate host names for the IP addresses that you place in the DHCP service, the DHCP server can register those host names in NIS+, `/etc/inet/hosts`, or DNS name services. Host name registration cannot be done in NIS because NIS does not provide a protocol to allow programs to update and propagate NIS maps.

---

**Note –** The DHCP server can update DNS with generated host names only if the DNS server and the DHCP server are running on the same system.

---

If a DHCP client provides its host name and the DNS server is configured to allow dynamic updates from the DHCP server, the DHCP server can update DNS on the client's behalf. Dynamic updates can be done even if the DNS and DHCP servers are running on different systems. See "Enabling Dynamic DNS Updates by a DHCP Server" on page 352 for more information about enabling this feature.

The following table summarizes client host name registration for DHCP client systems with the various name services.

**TABLE 15–2** Client Host Name Registration in Name Services

| | Who Registers Host Name | |
|---|---|---|
| **Name Service** | **DHCP-Generated Host Name** | **DHCP Client-Supplied Host Name** |
| NIS | NIS Administrator | NIS Administrator |
| NIS+ | DHCP tools | DHCP tools |
| `/etc/hosts` | DHCP tools | DHCP tools |
| DNS | DHCP tools, if the DNS server runs on the same system as the DHCP server | DHCP server, if configured for dynamic DNS updates |
| | DNS Administrator, if the DNS server runs on a different system | DNS Administrator, if DHCP server is not configured for dynamic DNS updates |

Solaris DHCP clients can request particular host names in DHCP requests if configured to do so as described in "How to Enable a Solaris DHCPv4 Client to Request a Specific Host Name" on page 430. Refer to the vendor documentation for other DHCP clients to determine if the capability is supported.

## Customizing Performance Options for the DHCP Server

You can change options that affect the performance of the DHCP server. These options are described in the following table.

**TABLE 15–3** Options Affecting DHCP Server Performance

| Server Option | Description | Keyword |
|---|---|---|
| Maximum number of BOOTP relay agent hops | If a request has traveled through more than a given number of BOOTP relay agents, the request is dropped. The default maximum number of relay agent hops is four. This number is likely to be sufficient for most networks. A network might need more than four hops if DHCP requests pass through several BOOTP relay agents before reaching a DHCP server. | `RELAY_HOPS=`*integer* |
| Detect duplicate addresses | By default, the server pings an IP address before offering the address to a client. A lack of response to the ping verifies that the address is not already in use. You can disable this feature to decrease the time that the server takes to make an offer. However, disabling the feature creates the risk of having duplicate IP addresses in use. | `ICMP_VERIFY=TRUE/FALSE` |
| Reload `dhcptab` automatically at specified intervals | The server can be set to automatically read the `dhcptab` at the interval, in minutes, that you specify. If your network configuration information does not change frequently, and you do not have multiple DHCP servers, you do not need to reload the `dhcptab` automatically. Also, note that DHCP Manager gives you the option to have the server reload the `dhcptab` after you make a change to the data. | `RESCAN_INTERVAL=`*min* |
| Cache offers of IP addresses for specified intervals | After a server offers an IP address to a client, the offer is cached. While the offer is cached, the server does not offer the address again. You can change the number of seconds for which the offer is cached. The default is 10 seconds. On slow networks, you might need to increase the offer time. | `OFFER_CACHE_TIMEOUT=`*sec* |

The following procedures describe how to change these options.

## ▼ How to Customize DHCP Performance Options (DHCP Manager)

**1 In DHCP Manager, choose Modify from the Service menu.**

See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2 **Change the desired options.**

See Table 15–3 for information about the options.

3 **Select Restart Server.**

4 **Click OK.**

## ▼ How to Customize DHCP Performance Options (Command Line)

If you change options with this procedure, the changed options are used only after the DHCP server is restarted.

1 **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Modify one or more performance options:**

```
# /usr/sbin/dhcpconfig -P keyword=value,keyword=value...
```

*keyword=value* can be any of the following keywords:

| | |
|---|---|
| `RELAY_HOPS=`*integer* | Specifies the maximum number of relay agent hops that can occur before the daemon drops the DHCP or BOOTP datagram. |
| `ICMP_VERIFY=TRUE/FALSE` | Enables or disables automatic duplicate IP address detection. Setting this keyword to `FALSE` is not recommended. |
| `RESCAN_INTERVAL=`*minutes* | Specifies the interval in minutes that the DHCP server should use to schedule the automatic rereading of the `dhcptab` information. |
| `OFFER_CACHE_TIMEOUT=`*seconds* | Specifies the number of seconds the DHCP server should cache the offers that are extended to discovering DHCP clients. The default setting is 10 seconds. |

**Example 15–1** Setting DHCP Performance Options

The following is an example of how to specify all the command options.

```
# dhcpconfig -P RELAY_HOPS=2,ICMP_VERIFY=TRUE,\
RESCAN_INTERVAL=30,OFFER_CACHE_TIMEOUT=20
```

# Adding, Modifying, and Removing DHCP Networks (Task Map)

When you configure a DHCP server, you must also configure at least one network in order to use the DHCP service. You can add more networks at any time.

The following task map lists tasks that you can perform when working with DHCP networks. The task map includes links to procedures for carrying out the tasks.

| Task | Description | For Instructions |
|---|---|---|
| Enable or disable the DHCP service on server network interfaces | The default behavior is to monitor all network interfaces for DHCP requests. If you do not want all interfaces to accept DHCP requests, you can remove an interface from the list of monitored interfaces. | "How to Specify Network Interfaces for DHCP Monitoring (DHCP Manager)" on page 359 |
| Add a new network to the DHCP service. | Places a network under DHCP management, for the purpose of managing IP addresses on the network. | "How to Add a DHCP Network (DHCP Manager)" on page 361<br>"How to Add a DHCP Network (`dhcpconfig`)" on page 362 |
| Change parameters of a DHCP-managed network. | Modifies the information that is passed to clients of a particular network. | "How to Modify the Configuration of a DHCP Network (DHCP Manager)" on page 364<br>"How to Modify the Configuration of a DHCP Network (`dhtadm`)" on page 365 |
| Delete a network from the DHCP service. | Removes a network so that IP addresses on the network are no longer managed by DHCP. | "How to Remove a DHCP Network (DHCP Manager)" on page 366<br>"How to Remove a DHCP Network (`pntadm`)" on page 367 |

## Specifying Network Interfaces for DHCP Monitoring

By default, both `dhcpconfig` and DHCP Manager's Configuration Wizard configure the DHCP server to monitor all the server system's network interfaces. If you add a new network interface to the server system, the DHCP server automatically monitors the new interface when you boot the system. You can then add any networks to be monitored through the network interface.

However, you can also specify which network interfaces should be monitored, and which interfaces should be ignored. You might want to ignore an interface if you do not want to offer DHCP service on that network.

If you specify that any interface should be ignored, and then install a new interface, the DHCP server ignores the new interface. You must add the new interface to the server's list of monitored interfaces. You can specify interfaces with DHCP Manager or the `dhcpconfig` utility.

This section includes procedures that enable you to specify which network interfaces DHCP should monitor or ignore. The DHCP Manager procedure uses the Interfaces tab of the DHCP Manager's Modify Service Options dialog box, which is shown in the following figure.

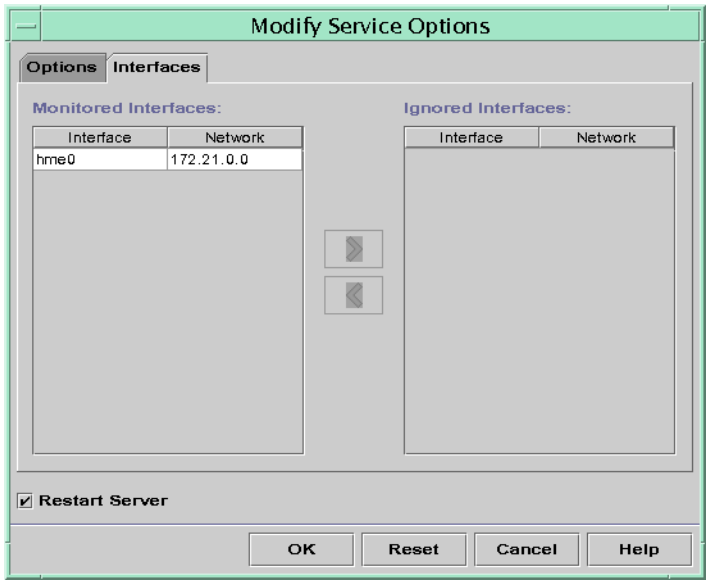



**FIGURE 15–4** Interfaces Tab of Modify Service Options Dialog Box in DHCP Manager

## How to Specify Network Interfaces for DHCP Monitoring (DHCP Manager)

1. **In DHCP Manager, choose Modify from the Service menu.**

   The Modify Service Options dialog box is displayed.

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Select the Interfaces tab.**

3 **Select the appropriate network interface.**

4 **Click the arrow buttons to move the interface to the appropriate list.**

For example, to ignore an interface, select the interface in the Monitored Interfaces list, and then click the right arrow button. The interface is then shown in the Ignored Interfaces list.

5 **Select Restart Server, and click OK.**

The changes you make persist across reboots.

## ▼ How to Specify Network Interfaces for DHCP Monitoring (`dhcpconfig`)

1 **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Type the following command on the DHCP server system:**

```
# /usr/sbin/dhcpconfig -P INTERFACES=int,int,...
```

*int, int,...* is a list of interfaces to monitor. The interface names must be separated by commas.

For example, you would use the following command to monitor only `ge0` and `ge1`:

```
#/usr/sbin/dhcpconfig -P INTERFACES=ge0,ge1
```

Interfaces that you want to ignore should be omitted from the `dhcpconfig` command line.

The changes you make with this command persist across reboots.

# Adding DHCP Networks

When you use DHCP Manager to configure the server, the first network is also configured at the same time. The first network is usually the local network on the server system's primary interface. If you want to configure additional networks, use the DHCP Network Wizard in DHCP Manager.

If you use the `dhcpconfig -D` command to configure the server, you must separately configure all networks that you want to use the DHCP service. See "How to Add a DHCP Network (`dhcpconfig`)" on page 362 for more information.

The following figure shows the initial dialog box for the DHCP Network Wizard in DHCP Manager.

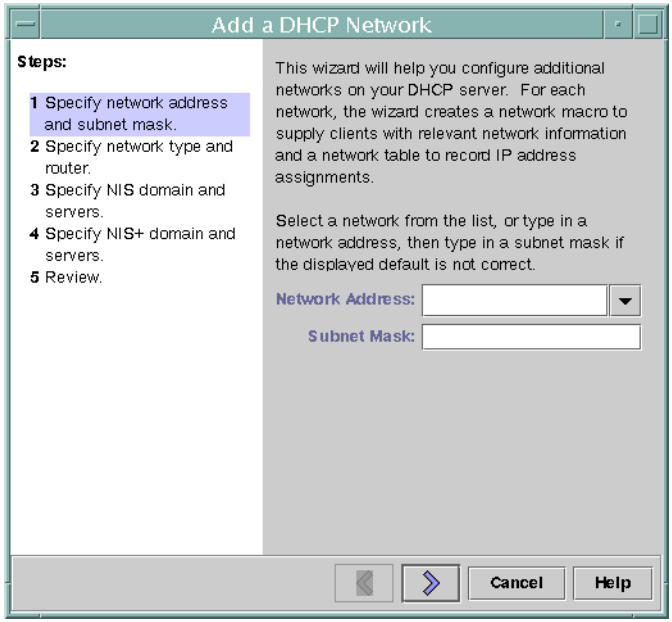



**FIGURE 15–5** DHCP Manager's Network Wizard

When you configure a new network, DHCP Manager creates the following components:

- A network table in the data store. The new network is shown in the network list within the Addresses tab of DHCP Manager.
- A network macro that contains information needed by clients that reside on this network. The network macro's name matches the IP address of the network. The network macro is added to the `dhcptab` table in the data store.

## ▼ How to Add a DHCP Network (DHCP Manager)

**1 In DHCP Manager, click the Addresses tab.**

Any networks already configured for DHCP service are listed.

See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

**2 Choose Network Wizard from the Edit menu.**

3 **Select options, or type requested information. Use the decisions that you made during the planning phase to determine what information to specify.**

Planning is described in "Planning DHCP Configuration of Your Remote Networks" on page 326.

If you have difficulty with the wizard, click Help in the wizard window. Your web browser displays help for the DHCP Network Wizard.

4 **Click Finish to complete the network configuration when you have finished specifying the requested information.**

The Network Wizard creates an empty network table, which is listed in the left pane of the window.

The Network Wizard also creates a network macro whose name matches the IP address of the network.

5 **(Optional) Select the Macros tab and select the network macro to view the macro's contents.**

You can confirm that the information that you provided in the wizard has been inserted as values for options in the network macro.

**See Also** You must add addresses for the network before the network's IP addresses can be managed under DHCP. See "Adding IP Addresses to the DHCP Service" on page 374 for more information.

If you leave the network table empty, the DHCP server can still provide configuration information to clients. See "Setting Up DHCP Clients to Receive Information Only (Task Map)" on page 407 for more information.

## ▼ How to Add a DHCP Network (`dhcpconfig`)

1 **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Type the following command on the DHCP server system:**

```
# /usr/sbin/dhcpconfig -N network-address
```

*network-address* is the IP address of the network you want to add to the DHCP service. See the `dhcpconfig`(1M) man page for suboptions you can use with the `-N` option.

If you do not use suboptions, `dhcpconfig` uses network files to obtain information about the network.

**See Also** You must add addresses for the network before the network's IP addresses can be managed under DHCP. See "Adding IP Addresses to the DHCP Service" on page 374 for more information.

If you leave the network table empty, the DHCP server can still provide configuration information to clients. See "Setting Up DHCP Clients to Receive Information Only (Task Map)" on page 407 for more information.

# Modifying DHCP Network Configurations

After you add a network to the DHCP service, you can modify the configuration information that you originally supplied. The configuration information is stored in the network macro used to pass information to clients on the network. You must modify the network macro to change the network configuration.

The following figure shows the Macros tab of DHCP Manager.

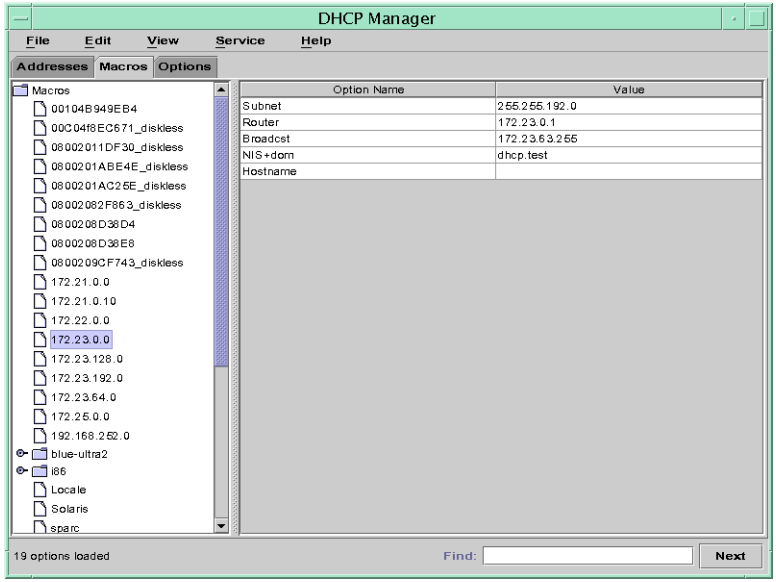



**FIGURE 15–6** DHCP Manager's Macros Tab

## ▼ How to Modify the Configuration of a DHCP Network (DHCP Manager)

1. **In DHCP Manager, select the Macros tab.**

   All macros that are defined for this DHCP server are listed in the left pane.

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Select the network macro whose name matches the network configuration that you are changing.**

   The network macro name is the network IP address.

3. **Choose Properties from the Edit menu.**

   The Macro Properties dialog box displays a table of the options included in the macro.

4. **Select the option that you want to modify.**

   The option name and its value are displayed in text fields near the top of the dialog box.

5. **(Optional) Modify the option name, or choose the Select button to display a list of option names.**

   The Select Option dialog box displays a list of all DHCP standard options, with a brief description of each option.

6. **(Optional) Select an option name in the Select Option dialog box, and click OK.**

   The new option name is displayed in the Option Name field.

7. **Type the new value for the option, and click Modify.**

8. **(Optional) You can also add options to the network macro by choosing Select in the dialog box.**

   See "Modifying DHCP Macros" on page 388 for more general information about modifying macros.

9. **Select Notify DHCP Server of Change, and click OK.**

   This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

## ▼ How to Modify the Configuration of a DHCP Network (dhtadm)

1. **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

   For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Determine which macro includes information for all clients of the network.**

   The network macro's name matches the network IP address.

   If you don't know which macro includes this information, you can display the `dhcptab` table to list all macros by using the command `dhtadm -P`.

3. **Type a command of the following format to change the value of the option you want to change:**

   **`# dhtadm -M -m`** *macro-name* **`-e 'symbol=value' -g`**

   See the `dhtadm`(1M) man page for more information about `dhtadm` command-line options.

**Example 15–2** Using the `dhtadm` Command to Modify a DHCP Macro

For example, to change the `10.25.62.0` macro's lease time to 57600 seconds and the NIS domain to `sem.example.com`, you would type the following commands:

```
# dhtadm -M -m 10.25.62.0 -e 'LeaseTim=57600' -g

# dhtadm -M -m 10.25.62.0 -e 'NISdmain=sem.example.com' -g
```

The `-g` option causes the DHCP daemon to reread the `dhcptab` table and put the changes into effect.

# Removing DHCP Networks

DHCP Manager enables you to remove multiple networks at once. You have the option to automatically remove the hosts table entries associated with the DHCP-managed IP addresses on those networks as well. The following figure shows DHCP Manager's Delete Networks dialog box.

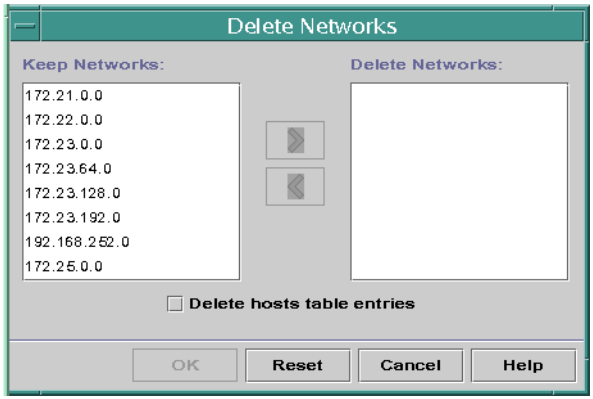



**FIGURE 15–7** Delete Networks Dialog Box in DHCP Manager

The `pntadm` command requires you to delete each IP address entry from a network before you delete that network. You can delete only one network at a time.

## ▼ How to Remove a DHCP Network (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Choose Delete Networks from the Edit menu.**

   The Delete Networks dialog box opens.

3. **In the Keep Networks list, select the networks that you want to delete.**

   Press the Control key while you click with the mouse to select multiple networks. Press the Shift key while you click to select a range of networks.

4. **Click the right arrow button to move the selected networks to the Delete Networks list.**

5. **If you want to remove the host table entries for this network's DHCP addresses, select Delete Host Table Entries.**

   Note that deleting host table entries does not delete the host registrations at the DNS server for these addresses. Entries are deleted only in the local name service.

6. **Click OK.**

### ▼ How to Remove a DHCP Network (`pntadm`)

Note that this procedure deletes the network's IP addresses from the DHCP network table before removing the network. The addresses are deleted to ensure that the host names are removed from the `hosts` file or database.

1. **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

   For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Type a command following this format to remove an IP address and its host name from the name service:**

   **`# pntadm -D -y`** *IP-address*

   For example, to remove IP address `10.25.52.1`, you would type the following command:

   **`# pntadm -D -y 10.25.52.1`**

   The `-y` option specifies to delete the host name.

3. **Repeat the `pntadm -D -y` command for each address in the network.**

   You might want to create a script to run the `pntadm` command if you are deleting many addresses.

4. **After all addresses are deleted, type the following command to delete the network from the DHCP service.**

   **`# pntadm -R`** *network-IP-address*

   For example, to remove network `10.25.52.0`, you would type the following command:

   **`# pntadm -R 10.25.52.0`**

   See the `pntadm`(1M) man page for more information about using the `pntadm` utility.

# Supporting BOOTP Clients With the DHCP Service (Task Map)

To support BOOTP clients on your DHCP server, you must set up your DHCP server to be BOOTP compatible. If you want to specify which BOOTP clients can use your DHCP, you can register BOOTP clients in the DHCP server's network table. Alternatively, you can reserve a number of IP addresses for automatic allocation to BOOTP clients.

**Note –** BOOTP addresses are permanently assigned, whether or not you explicitly assign a permanent lease to the address.

The following task map lists tasks that you might need to perform to support BOOTP clients. The task map contains links to the procedures used to carry out the tasks.

| Task | Description | For Instructions |
|---|---|---|
| Set up automatic BOOTP support. | Provides IP address for any BOOTP client on a DHCP-managed network, or on a network connected by a relay agent to a DHCP-managed network.<br>You must reserve a pool of addresses for exclusive use by BOOTP clients. This option might be more useful if the server must support a large number of BOOTP clients. | "How to Set Up Support of Any BOOTP Client (DHCP Manager)" on page 368 |
| Set up manual BOOTP support. | Provides IP address for only those BOOTP clients that have been manually registered with the DHCP service.<br>This option requires you to bind a client's ID to a particular IP address that has been marked for BOOTP clients. This option is useful for a small number of BOOTP clients, or when you want to restrict the BOOTP clients that can use the DHCP server. | "How to Set Up Support of Registered BOOTP Clients (DHCP Manager)" on page 369 |

## ▼ How to Set Up Support of Any BOOTP Client (DHCP Manager)

1. **In DHCP Manager, select Modify from the Service menu.**

   The Modify Service Options dialog box opens.

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **In the BOOTP Compatibility section of the dialog box, select Automatic.**

3. **Select Restart Server, and click OK.**

4. **Select the Addresses tab.**

5. **Select addresses that you want to reserve for BOOTP clients.**

   Select a range of addresses by clicking the first address, pressing the Shift key, and clicking the last address. Select multiple nonconcurrent addresses by pressing the Control key while clicking each address.

6. **Select Properties from the Edit menu.**

   The Modify Multiple Addresses dialog box opens.

7. **In the BOOTP section, select Assign All Addresses Only to BOOTP Clients.**

   All other options should be set to Keep Current Settings.

8. **Click OK.**

   Any BOOTP client can now obtain an address from this DHCP server.

## ▼ How to Set Up Support of Registered BOOTP Clients (DHCP Manager)

1. **In DHCP Manager, select Modify from the Service menu.**

   The Modify Service Options dialog box opens.

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **In the BOOTP Compatibility section of the dialog box, select Manual.**

3. **Select Restart Server, and click OK.**

4. **Select the Addresses tab.**

5. **Select an address that you want to assign to a particular BOOTP client.**

6. **Choose Properties from the Edit menu.**

   The Address Properties dialog box opens.

7. **In the Address Properties dialog box, select the Lease tab.**

8. **In the Client ID field, type the client's identifier.**

   For a BOOTP Solaris client on an Ethernet network, the client ID is a string that is derived from the client's hexadecimal Ethernet address. The client ID includes a prefix that indicates the

Address Resolution Protocol (ARP) type for Ethernet (01). For example, a BOOTP client with the Ethernet address `8:0:20:94:12:1e` would use the client ID `0108002094121E`.

---

**Tip –** As superuser on a Solaris client system, type the following command to obtain the Ethernet address for the interface:

```
# ifconfig -a
```

---

**9 Select Reserved to reserve the IP address for this client.**

**10 Select Assign Only to BOOTP Clients, and click OK.**

In the Addresses tab, BOOTP is displayed in the Status field, and the client ID you specified is listed in the Client ID field.

# Working With IP Addresses in the DHCP Service (Task Map)

You can use DHCP Manager or the `pntadm` command to add IP addresses, modify address properties, and remove addresses from the DHCP service. Before you work with IP addresses, you should refer to Table 15–4 to become familiar with IP address properties. The table provides information for users of DHCP Manager and `pntadm`.

---

**Note –** Table 15–4 includes examples of using `pntadm` to specify IP address properties while adding and modifying IP addresses. Refer also to the `pntadm`(1M) man page for more information about `pntadm`.

---

The following task map lists tasks that you must perform to add, modify, or remove IP addresses. The task map also contains links to the procedures used to carry out the tasks.

| Task | Description | For Instructions |
|---|---|---|
| Add single or multiple IP addresses to the DHCP service. | Adds IP addresses on networks that are already managed by the DHCP service by using DHCP Manager. | "How to Add a Single IP Address (DHCP Manager)" on page 375<br>"How to Duplicate an Existing IP Address (DHCP Manager)" on page 376<br>"How to Add Multiple IP Addresses (DHCP Manager)" on page 376<br>"How to Add IP Addresses (`pntadm`)" on page 377 |

| Task | Description | For Instructions |
|---|---|---|
| Change properties of an IP address. | Changes any of the IP address properties described in Table 15–4. | "How to Modify IP Address Properties (DHCP Manager)" on page 379<br>"How to Modify IP Address Properties (pntadm)" on page 379 |
| Remove IP addresses from the DHCP service. | Prevents the use of specified IP addresses by DHCP. | "How to Mark IP Addresses as Unusable (DHCP Manager)" on page 380<br>"How to Mark IP Addresses as Unusable (pntadm)" on page 381<br>"How to Delete IP Addresses From DHCP Service (DHCP Manager)" on page 382<br>"How to Delete IP Addresses From the DHCP Service (pntadm)" on page 383 |
| Assign a consistent IP address to a DHCP client. | Sets up a client to receive the same IP address each time the client requests its configuration. | "How to Assign a Consistent IP Address to a DHCP Client (DHCP Manager)" on page 384<br>"How to Assign a Consistent IP Address to a DHCP Client (pntadm)" on page 385 |

The following table lists and describes the properties of IP addresses.

**TABLE 15–4** IP Address Properties

| Property | Description | How to Specify in `pntadm` Command |
|---|---|---|
| Network address | The address of the network that contains the IP address that you are working with.<br>The network address is displayed in the Networks list within the Addresses tab in DHCP Manager. | The network address must be the last argument on the `pntadm` command line used to create, modify, or delete an IP address.<br>For example, to add an IP address to network `10.21.0.0`, you would type:<br>**`pntadm -A`** *ip-address options* **`10.21.0.0`** |
| IP address | The address you are working with, whether you are creating, modifying, or deleting the address.<br>The IP address is displayed in the first column of the DHCP Manager's Addresses tab. | The IP address must accompany the `-A`, `-M`, and `-D` options to the `pntadm` command.<br>For example, to modify IP address `10.21.5.12`, you would type:<br>**`pntadm -M 10.21.5.12`** *options* **`10.21.0.0`** |

**TABLE 15–4** IP Address Properties *(Continued)*

| Property | Description | How to Specify in `pntadm` Command |
|---|---|---|
| Client name | The host name mapped to the IP address in the hosts table. This name can be automatically generated by DHCP Manager when addresses are created. If you create a single address, you can supply the name. | Specify the client name with the `-h` option.<br>For example, to specify client name `carrot12` for `10.21.5.12`, you would type:<br>**`pntadm -M 10.21.5.12 -h carrot12 10.21.0.0`** |
| Owned by server | The DHCP server that manages the IP address and responds to the DHCP client's request for IP address allocation. | Specify the owning server name with the `-s` option.<br>For example to specify server `blue2` to own `10.21.5.12`, you would type:<br>**`pntadm -M 10.21.5.12 -s blue2 10.21.0.0`** |
| Configuration macro | The macro that the DHCP server uses to obtain network configuration options from the `dhcptab` table. Several macros are created automatically when you configure a server, and when you add networks. See "About DHCP Macros" on page 311 for more information about macros. When addresses are created, a server macro is also created. The server macro is assigned as the configuration macro for each address. | Specify the macro name with the `-m` option.<br>For example, to assign the server macro `blue2` to address `10.21.5.12`, you would type:<br>**`pntadm -M 10.21.5.12 -m blue2 10.21.0.0`** |
| Client ID | A text string that is unique within the DHCP service.<br>If the client ID is listed as 00, the address is not allocated to any client. If you specify a client ID when modifying the properties of an IP address, the address is bound exclusively to that client.<br>The client ID is determined by the vendor of the DHCP client. If your client is not a Solaris DHCP client, consult your DHCP client documentation for more information. | Specify the client ID with the `-i` option.<br>For example, to assign client ID `08002094121E` to address `10.21.5.12`, you would type:<br>**`pntadm -M 10.21.5.12 -i 0108002094121E 10.21.0.0`** |

**TABLE 15–4** IP Address Properties *(Continued)*

| Property | Description | How to Specify in `pntadm` Command |
|---|---|---|
| | For Solaris DHCP clients, the client ID is derived from the client's hexadecimal hardware address. The client ID includes a prefix that represents the ARP code for the type of network, such as 01 for Ethernet. The ARP codes are assigned by the Internet Assigned Numbers Authority (IANA) in the ARP Parameters section of the Assigned Numbers standard at `http://www.iana.com/numbers.html`<br><br>For example, a Solaris client with the hexadecimal Ethernet address `8:0:20:94:12:1e` uses the client ID `0108002094121E`. The client ID is listed in DHCP Manager and `pntadm` when a client is currently using an address.<br><br>**Tip:** As superuser on the Solaris client system, type the following command to obtain the Ethernet address for the interface: `ifconfig -a` | |
| Reserved | The setting that specifies the address is reserved exclusively for the client indicated by the client ID, and the DHCP server cannot reclaim the address. If you choose this option, you manually assign the address to the client. | Specify that the address is reserved, or manual, with the `-f` option.<br><br>For example, to specify that IP address `10.21.5.12` is reserved for a client, you would type:<br><br>**`pntadm -M 10.21.5.12 -f MANUAL 10.21.0.0`** |
| Lease type or policy | The setting that determines how DHCP manages the use of IP addresses by clients. A lease is either dynamic or permanent. See "Dynamic and Permanent Lease Types" on page 325 for a complete explanation. | Specify that the address is permanently assigned with the `-f` option. Addresses are dynamically leased by default.<br><br>For example, to specify that IP address `10.21.5.12` has a permanent lease, you would type:<br><br>**`pntadm -M 10.21.5.12 -f PERMANENT 10.21.0.0`** |
| Lease expiration date | The date when the lease expires, applicable only when a dynamic lease is specified. The date is specified in *mm/dd/yyyy* format. | Specify a lease expiration date with the `-e` option.<br><br>For example, to specify an expiration date of January 1, 2006, you would type:<br><br>**`pntadm -M 10.21.5.12 -e 01/01/2006 10.21.0.0`** |
| BOOTP setting | The setting that marks the address as reserved for BOOTP clients. See "Supporting BOOTP Clients With the DHCP Service (Task Map)" on page 367 for more information about supporting BOOTP clients. | Reserve an address for BOOTP clients with the `-f` option.<br><br>For example, to reserve IP address `10.21.5.12` for BOOTP clients, you would type:<br><br>**`pntadm -M 10.21.5.12 -f BOOTP 10.21.0.0`** |

**TABLE 15–4** IP Address Properties *(Continued)*

| Property | Description | How to Specify in `pntadm` Command |
|---|---|---|
| Unusable setting | The setting that marks the address to prevent assignment of the address to any client. | Mark an address as unusable with the `-f` option.<br>For example, to mark IP address `10.21.5.12` as unusable, you would type:<br>**`pntadm -M 10.21.5.12 -f UNUSABLE 10.21.0.0`** |

# Adding IP Addresses to the DHCP Service

Before you add IP addresses, you must add the network that owns the addresses to the DHCP service. See "Adding DHCP Networks" on page 360 for information about adding networks.

You can add addresses with DHCP Manager or the `pntadm` command.

On networks that are already managed by the DHCP service, you can add addresses in several ways with DHCP Manager:

- **Add a single IP address** – Place one new IP address under DHCP management.
- **Duplicate an existing IP address** – Copy the properties of an existing IP address managed by DHCP, and supply a new IP address and client name.
- **Add a range of multiple IP addresses** – Use the Address Wizard to place a series of IP addresses under DHCP management.

The following figure shows the Create Address dialog box. The Duplicate Address dialog box is identical to the Create Address dialog box, except that the text fields display the values for an existing address.

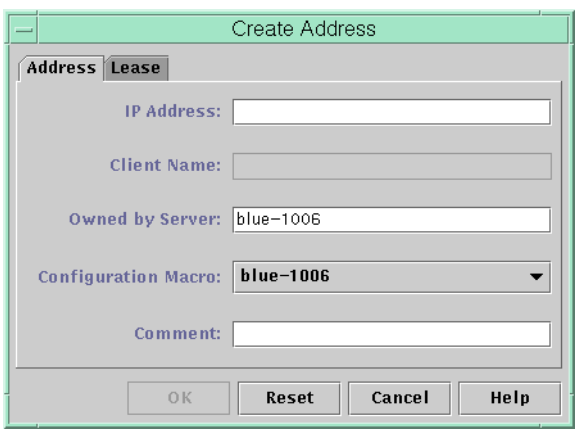



**FIGURE 15–8** Create Address Dialog Box in DHCP Manager

The following figure shows the first dialog of the Add Addresses to Network wizard, used to add a range of IP addresses.

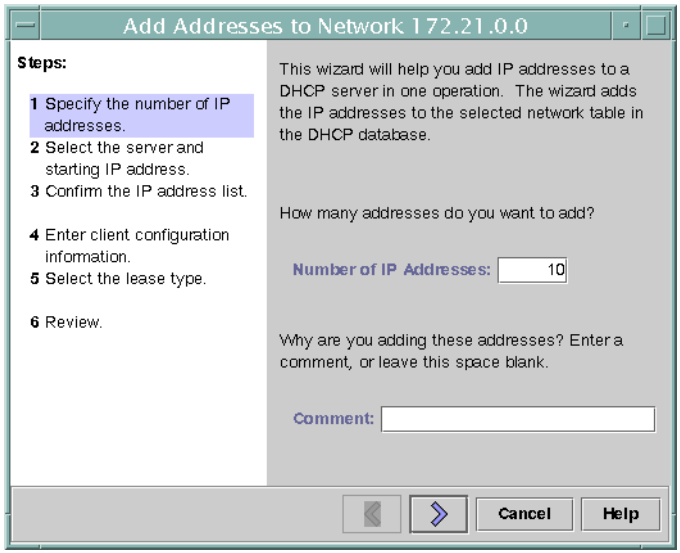



**FIGURE 15–9** Add Addresses to Network Wizard in DHCP Manager

## How to Add a Single IP Address (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Select the network where the new IP address is to be added.**

3. **Choose Create from the Edit menu.**

   The Create Address dialog box opens.

4. **Select or type values for the address settings on the Address and Lease tabs.**

   Select the Help button to open a web browser to display help for the dialog box. Also, see Table 15–4 for detailed information about the settings.

5. **Click OK.**

## How to Duplicate an Existing IP Address (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Select the network where the new IP address is located.**

3. **Select the address with properties that you want to duplicate.**

4. **Choose Duplicate from the Edit menu.**

5. **Specify the new IP address in the IP Address field.**

6. **(Optional) Specify a new client name for the address.**

   You cannot use the same name that is used by the address that you are duplicating.

7. **(Optional) Modify other option values, if necessary.**

   Most other option values should remain the same.

8. **Click OK.**

## How to Add Multiple IP Addresses (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Select the network where the new IP addresses are to be added.**

3. **Choose Address Wizard from the Edit menu.**

   The Add Addresses to Network dialog box prompts you to provide values for the IP address properties. See Table 15–4 for more information about the properties, or select the Help button in the dialog box. "Making Decisions for IP Address Management (Task Map)" on page 322 includes more extensive information.

4. **Click the right arrow button as you finish each screen, and click Finish on the last screen.**

   The Addresses tab is updated with the new addresses.

### ▼ How to Add IP Addresses (`pntadm`)

1. **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

   For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Add IP addresses by typing a command of the following format:**

   `# pntadm -A` *ip-address options network-address*

   Refer to the `pntadm`(1M) man page for a list of options you can use with `pntadm -A`. In addition, Table 15–4 shows some sample `pntadm` commands that specify options.

---

**Note –** You can write a script to add multiple addresses with `pntadm`. See Example 18–1 for an example.

---

## Modifying IP Addresses in the DHCP Service

You can modify any of the address properties described in Table 15–4 by using DHCP Manager or the `pntadm -M` command. See the `pntadm`(1M) man page for more information about `pntadm -M`.

The following figure shows the Address Properties dialog box that you use to modify IP address properties.

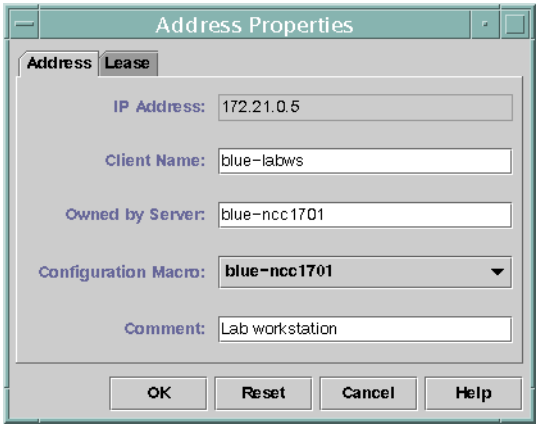



**FIGURE 15–10** Address Properties Dialog Box in DHCP Manager

The following figure shows the Modify Multiple Addresses dialog box that you use to modify multiple IP addresses.

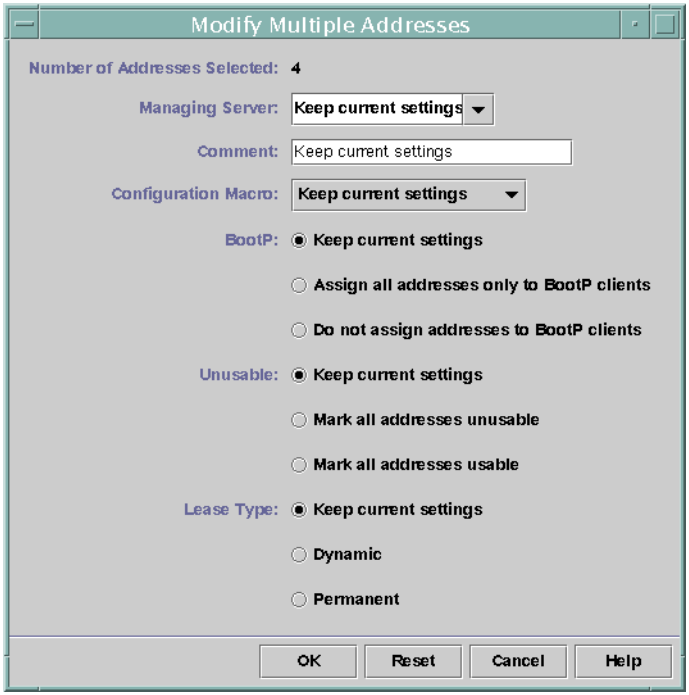



**FIGURE 15–11** Modify Multiple Addresses Dialog Box in DHCP Manager

## How to Modify IP Address Properties (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Select the IP address's network.**

3. **Select one or more IP addresses to modify.**

   If you want to modify more than one address, press the Control key while you click with the mouse to select multiple addresses. You can also press the Shift key while you click to select a block of addresses.

4. **Choose Properties from the Edit menu.**

   The Address Properties dialog box or the Modify Multiple Address dialog box opens.

5. **Change the appropriate properties.**

   Click the Help button, or refer to Table 15–4 for information about the properties.

6. **Click OK.**

## How to Modify IP Address Properties (`pntadm`)

1. **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

   For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Modify IP address properties by typing a command of the following format:**

   `# pntadm -M` *ip-address options network-address*

   Many options can be used with the `pntadm` command, which are documented in the `pntadm`(1M) man page.

   Table 15–4 shows some sample `pntadm` commands that specify options.

## Removing IP Addresses From the DHCP Service

At times, you might want the DHCP service to stop managing a particular IP address or group of addresses. The method that you use to remove an address from DHCP depends on whether you want the change to be temporary or permanent.

- To temporarily prevent the use of addresses, you can mark the addresses as unusable in the Address Properties dialog box as described in "Marking IP Addresses as Unusable by the DHCP Service" on page 380.
- To permanently prevent the use of addresses by DHCP clients, delete the addresses from the DHCP network tables, as described in "Deleting IP Addresses From the DHCP Service" on page 381.

## Marking IP Addresses as Unusable by the DHCP Service

You can use the `pntadm -M` command with the `-f UNUSABLE` option to mark addresses as unusable.

In DHCP Manager, you use the Address Properties dialog box, shown in Figure 15–10, to mark individual addresses. You use the Modify Multiple Addresses dialog box, show in Figure 15–11, to mark multiple addresses, as described in the following procedure.

### ▼ How to Mark IP Addresses as Unusable (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Select the IP address's network.**

3. **Select one or more IP addresses to mark as unusable.**

   If you want to mark more than one address as unusable, press the Control key while you click with the mouse to select multiple addresses. You can also press the Shift key while you click to select a block of addresses.

4. **Choose Properties from the Edit menu.**

   The Address Properties dialog box or the Modify Multiple Address dialog box opens.

5. **If you are modifying one address, select the Lease tab.**

6. **Select Address is Unusable.**

   If you are editing multiple addresses, select Mark All Addresses Unusable.

7. **Click OK.**

## ▼ How to Mark IP Addresses as Unusable (`pntadm`)

1. **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

   For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Mark IP addresses as unusable by typing a command of the following format:**

   `# pntadm -M` *ip-address* `-f UNUSABLE` *network-address*

   For example, to mark address `10.64.3.3` as unusable, type:

   ```
   pntadm -M 10.64.3.3 -f UNUSABLE 10.64.3.0
   ```

# Deleting IP Addresses From the DHCP Service

You should delete IP addresses from the DHCP network tables if you no longer want the address to be managed by DHCP. You can use the `pntadm -D` command or DHCP Manager's Delete Address dialog box.

The following figure shows the Delete Address dialog box.

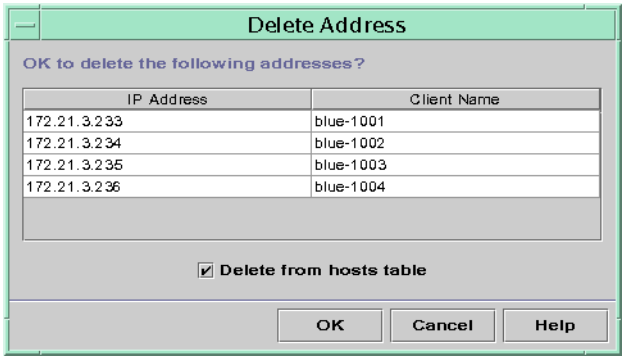



**FIGURE 15–12** Delete Address Dialog Box in DHCP Manager

## ▼ How to Delete IP Addresses From DHCP Service (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Select the IP address's network.**

3. **Select one or more IP addresses to delete.**

   If you want to delete more than one address, press the Control key while you click with the mouse to select multiple addresses. You can also press the Shift key while you click to select a block of addresses.

4. **Choose Delete from the Edit menu.**

   The Delete Address dialog box lists the address that you selected so that you can confirm the deletion.

5. **If you want to delete the host names from the hosts table, select Delete From Hosts Table.**

   If the host names were generated by DHCP Manager, you might want to delete the names from the hosts table.

6. **Click OK.**

### ▼ How to Delete IP Addresses From the DHCP Service (pntadm)

1. **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

   For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Delete IP addresses by typing a command of the following format:**

   ```
   # pntadm -D ip-address options network-address
   ```

   If you include the `-y` option, the host name is deleted from the name service that maintains the host name.

   For example, to delete address `10.64.3.3` from network `10.64.3.0`, and delete the corresponding host name, type:

   ```
   pntadm -D 10.64.3.3 -y 10.64.3.0
   ```

## Assigning a Reserved IP Address to a DHCP Client

The Solaris DHCP service attempts to provide the same IP address to a client that has previously obtained an address through DHCP. However, sometimes an address has already been reassigned to another client.

Routers, NIS or NIS+ servers, DNS servers, and other hosts that are critical to the network should not be DHCP clients. Hosts that provide services to the network should not rely on the network to obtain their IP addresses. Clients such as print servers or file servers should have consistent IP addresses as well. These clients can receive their network configurations and also be assigned a consistent IP address from the DHCP server.

You can set up the DHCP server to supply the same IP address to a client each time the client requests its configuration. You reserve the IP address for the client by manually assigning the client's ID to the address that you want the client to use. You can set up the reserved address to use either a dynamic lease or a permanent lease. If the client's address uses a dynamic lease, you can easily track the use of the address. A diskless client is an example of a client that should use a reserved address with a dynamic lease. If the client's address uses a permanent lease, you cannot track address use. Once a client obtains a permanent lease, the client does not contact the server again. The client can obtain updated configuration information only by releasing the IP address and restarting the DHCP lease negotiation.

You can use the `pntadm -M` command or DHCP Manager's Address Properties dialog box to set up lease properties.

The following figure shows the Lease tab of the Address Properties dialog box, which is used to modify the lease.

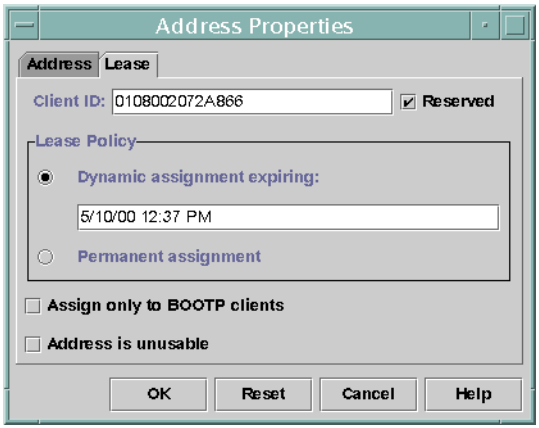



**FIGURE 15–13** Address Properties Lease Tab in DHCP Manager

## How to Assign a Consistent IP Address to a DHCP Client (DHCP Manager)

1. **In DHCP Manager, select the Addresses tab.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Select the appropriate network.**

3. **Double-click the IP address that you want to the client to use.**

   The Address Properties window opens.

4. **Select the Lease tab.**

5. **In the Client ID field, type the client ID.**

   The client ID is derived from the client's hardware address. See the Client ID entry in Table 15–4 for more information.

6. **Select the Reserved option to prevent the IP address from being reclaimed by the server.**

7 **In the Lease Policy area of the window, select Dynamic or Permanent assignment.**

Select Dynamic if you want the client to negotiate to renew leases, which enables you to track when the address is used. Because you selected Reserved, the address cannot be reclaimed even when a dynamic lease is assigned. You do not need to specify an expiration date for this lease. The DHCP server calculates the expiration date by using the lease time.

If you select Permanent, you cannot track the use of the IP address unless you enable transaction logging.

8 **Click OK.**

### ▼ How to Assign a Consistent IP Address to a DHCP Client (`pntadm`)

1 **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Set the lease flags by typing a command of the following format:**

```
# pntadm -M ip-address -i client-id -f MANUAL+BOOTP network-address
```

For example, to enable the Solaris DHCP client whose MAC address is `08:00:20:94:12:1E` to always receive IP address `10.21.5.12`, you would type:

```
pntadm -M 10.21.5.12 -i 0108002094121E -f MANUAL+BOOTP 10.21.0.0
```

**Tip –** Refer to the Client ID entry in Table 15–4 for more information about how to determine client identifiers.

## Working With DHCP Macros (Task Map)

*DHCP macros* are containers of DHCP options. The Solaris DHCP service uses macros to gather options that should be passed to clients. DHCP Manager and the `dhcpconfig` utility create a number of macros automatically when you configure the server. See "About DHCP Macros" on page 311 for background information about macros. See Chapter 14, "Configuring the DHCP Service (Tasks)," for information about macros created by default.

You might find that when changes occur on your network, you need to make changes to the configuration information that is passed to clients. To change configuration information, you need to work with DHCP macros. You can view, create, modify, duplicate, and delete DHCP macros.

When you work with macros, you must know about DHCP standard options, which are described in the `dhcp_inittab`(4) man page.

The following task map lists tasks to help you view, create, modify, and delete DHCP macros.

| Task | Description | For Instructions |
|---|---|---|
| View DHCP macros. | Display a list of all the macros that are defined on the DHCP server. | "How to View Macros Defined on a DHCP Server (DHCP Manager)" on page 387<br>"How to View Macros Defined on a DHCP Server (`dhtadm`)" on page 388 |
| Create DHCP macros. | Create new macros to support DHCP clients. | "How to Create a DHCP Macro (DHCP Manager)" on page 393<br>"How to Create a DHCP Macro (`dhtadm`)" on page 394 |
| Modify values that are passed in macros to DHCP clients. | Change macros by modifying existing options, adding options to macros, or removing options from macros. | "How to Change Values for Options in a DHCP Macro (DHCP Manager)" on page 389<br>"How to Change Values for Options in a DHCP Macro (`dhtadm`)" on page 390<br>"How to Add Options to a DHCP Macro (DHCP Manager)" on page 390<br>"How to Add Options to a DHCP Macro (`dhtadm`)" on page 391<br>"How to Delete Options From a DHCP Macro (DHCP Manager)" on page 391<br>"How to Delete Options From a DHCP Macro (`dhtadm`)" on page 392 |
| Delete DHCP macros. | Remove DHCP macros that are no longer used. | "How to Delete a DHCP Macro (DHCP Manager)" on page 395<br>"How to Delete a DHCP Macro (`dhtadm`)" on page 395 |

The following figure shows the Macros tab in the DHCP Manager window.

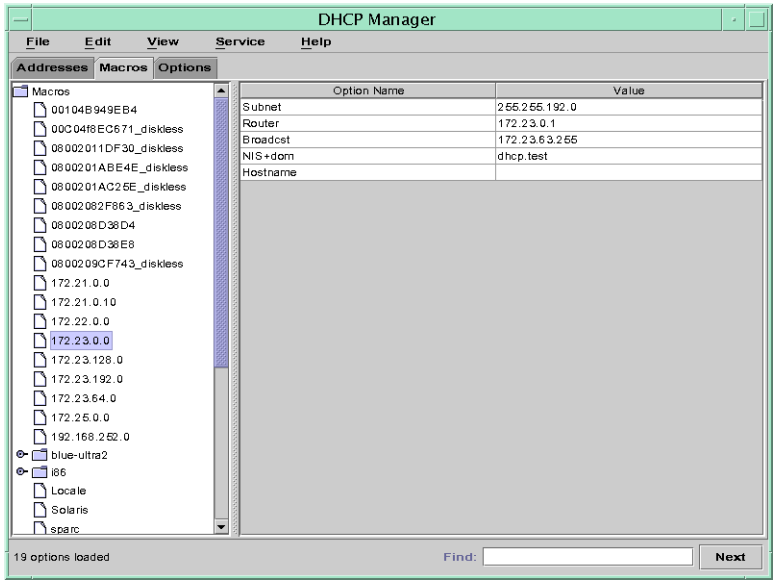



**FIGURE 15–14** DHCP Manager's Macros Tab

## ▼ How to View Macros Defined on a DHCP Server (DHCP Manager)

1. **In DHCP Manager, select the Macros tab.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

   The Macros area on the left side of the window displays, in alphabetical order, all the macros defined on the DHCP server. Macros preceded by a folder icon include references to other macros, whereas macros preceded by a document icon do not reference other macros.

2. **To open a macro folder, click the handle icon to the left of the folder icon.**

   The macros that are included in the selected macro are listed.

3. **To view the content of a macro, click the macro name.**

   Options and their assigned values are displayed.

### ▼ How to View Macros Defined on a DHCP Server (`dhtadm`)

1 **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Display the macros by typing the following command:**

```
# dhtadm -P
```

This command prints to standard output the formatted contents of the `dhcptab` table, including all macros and symbols defined on the DHCP server.

## Modifying DHCP Macros

You might need to modify macros when some aspect of your network changes and one or more DHCP clients need to know about the change. For example, you might add a router or an NIS server, create a new subnet, or change the lease policy.

Before you modify a macro, determine the name of the DHCP option you want to change, add, or delete. The standard DHCP options are listed in the DHCP Manager help and in the `dhcp_inittab`(4) man page.

You can use the `dhtadm -M -m` command or DHCP Manager to modify macros. See the `dhtadm`(1M) man page for more information about `dhtadm`.

The following figure shows DHCP Manager's Macro Properties dialog box.

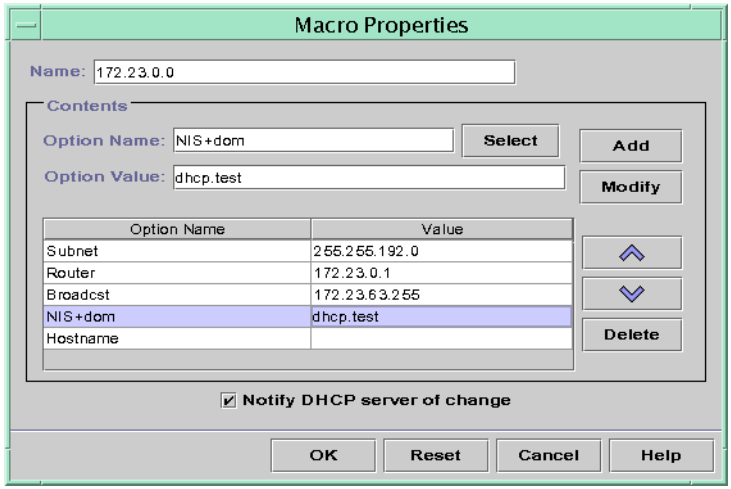



**FIGURE 15–15** Macro Properties Dialog Box in DHCP Manager

## How to Change Values for Options in a DHCP Macro (DHCP Manager)

1. **In DHCP Manager, select the Macros tab.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Select the macro that you want to change.**

3. **Choose Properties from the Edit menu.**

   The Macro Properties dialog box opens.

4. **In the table of Options, select the option that you want to change.**

   The option's name and its value are displayed in the Option Name and Option Value fields.

5. **In the Option Value field, select the old value and type the new value for the option.**

6. **Click Modify.**

   The new value is displayed in the options table.

7. **Select Notify DHCP Server of Change.**

   This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

8 **Click OK.**

## ▼ How to Change Values for Options in a DHCP Macro (`dhtadm`)

1 **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Change option values by typing a command of the following format:**

```
# dhtadm -M -m macroname -e 'option=value:option=value' -g
```

For example, to change the lease time and the Universal Time Offset in the macro `bluenote`, you would type:

```
# dhtadm -M -m bluenote -e 'LeaseTim=43200:UTCOffst=28800' -g
```

## ▼ How to Add Options to a DHCP Macro (DHCP Manager)

1 **In DHCP Manager, select the Macros tab.**

See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2 **Select the macro that you want to change.**

3 **Choose Properties from the Edit menu.**

The Macro Properties dialog box opens.

4 **In the Option Name field, specify the name of an option by using one of the following methods:**

- **Click the Select button next to the Option Name field to select an option to add to the macro.**

  The Select Option dialog box displays an alphabetized list of names of standard category options and descriptions. If you want to add an option that is not in the standard category, use the Category list to select a category.

  See "About DHCP Macros" on page 311 for more information about macro categories.

- **Type `Include` if you want to include a reference to an existing macro in the new macro.**

5 **Type the value for the option in the Option Value field.**

If you typed **`Include`** as the option name, you must specify the name of an existing macro in the Option Value field.

6 **Click Add.**

The option is added to the bottom of the list of options in this macro. To change the option's position in the macro, select the option and click the arrow buttons to move the option up or down in the list.

7 **Select Notify DHCP Server of Change.**

This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

8 **Click OK.**

## ▼ How to Add Options to a DHCP Macro (`dhtadm`)

1 **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Add options to a macro by typing a command of the following format:**

```
# dhtadm -M -m macroname -e 'option=value' -g
```

For example, to add the ability to negotiate leases in the macro `bluenote`, you would type the following command:

```
# dhtadm -M -m bluenote -e 'LeaseNeg=_NULL_VALUE' -g
```

Note that if an option does not require a value, you must use `_NULL_VALUE` as the value for the option.

## ▼ How to Delete Options From a DHCP Macro (DHCP Manager)

1 **In DHCP Manager, select the Macros tab.**

See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2 **Select the macro that you want to change.**

3 **Choose Properties from the Edit menu.**

The Macro Properties dialog box opens.

4 **Select the option that you want to remove from the macro.**

5 **Click Delete.**

The option is removed from the list of options for this macro.

6 **Select Notify DHCP Server of Change.**

This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

7 **Click OK.**

## ▼ How to Delete Options From a DHCP Macro (`dhtadm`)

1 **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Delete an option from a macro by typing a command of the following format:**

```
# dhtadm -M -m macroname -e 'option=' -g
```

For example, to remove the ability to negotiate leases in the macro `bluenote`, you would type the following command:

```
# dhtadm -M -m bluenote -e 'LeaseNeg=' -g
```

If an option is specified with no value, the option is removed from the macro.

# Creating DHCP Macros

You might want to add new macros to your DHCP service to support clients with specific needs. You can use the `dhtadm -A -m` command or DHCP Manager's Create Macro dialog box to add macros. See the `dhtadm`(1M) man page for more information about the `dhtadm` command.

The following figure shows DHCP Manager's Create Macro dialog box.

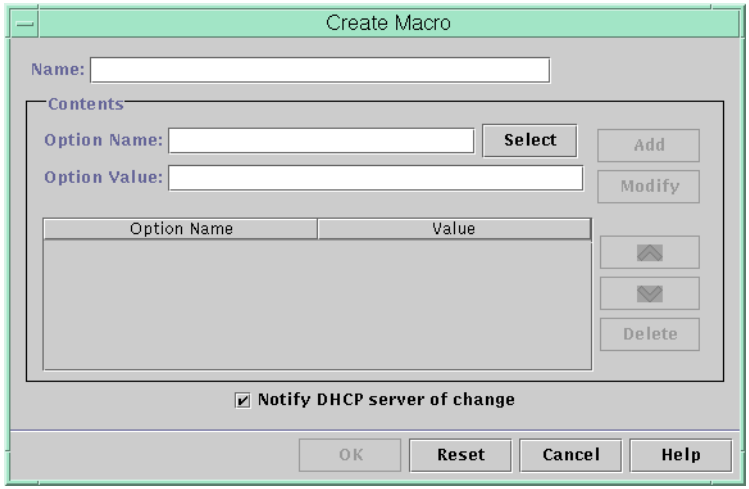



**FIGURE 15–16** Create Macro Dialog Box in DHCP Manager

## How to Create a DHCP Macro (DHCP Manager)

1. **In DHCP Manager, select the Macros tab.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Choose Create from the Edit menu.**

   The Create Macro dialog box opens.

3. **Type a unique name for the macro.**

   The name can be up to 128 alphanumeric characters. If you use a name that matches a vendor class identifier, network address, or client ID, the macro is processed automatically for appropriate clients. If you use a different name, the macro is not processed automatically. The macro must be assigned to a specific IP address or included in another macro that is processed automatically. See "Macro Processing by the DHCP Server" on page 311 for more detailed information.

4. **Click the Select button, which is next to the Option Name field.**

   The Select Option dialog box displays an alphabetized list of names of standard category options and their descriptions. If you want to add an option that is not in the standard category, use the Category list. Select the category that you want from the Category list. See "About DHCP Options" on page 310 for more information about option categories.

5. **Select the option to add to the macro, and click OK.**

   The Macro Properties dialog box displays the selected option in the Option Name field.

6 **Type the value for the option in the Option Value field, and click Add.**

The option is added to the bottom of the list of options in this macro. To change the option's position in the macro, select the option and click the arrow buttons to move the option up or down in the list.

7 **Repeat Step 5 and Step 6 for each option you want to add to the macro.**

8 **Select Notify DHCP Server of Change when you are finished adding options.**

This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

9 **Click OK.**

## ▼ How to Create a DHCP Macro (`dhtadm`)

1 **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Create a macro by typing a command of the following format:**

```
# dhtadm -A -m macroname -d ':option=value:option=value:option=value:' -g
```

There is no limit to the number of *option=value* pairs that can be included in the argument to `-d`. The argument must begin and end with colons, with colons between each *option=value* pair. The complete string must be enclosed in quotation marks.

For example, to create the macro `bluenote`, type the following command:

```
# dhtadm -A -m bluenote -d ':Router=10.63.6.121\
:LeaseNeg=_NULL_VALUE:DNSserv=10.63.28.12:' -g
```

Note that if an option does not require a value, you must use `_NULL_VALUE` as the value for the option.

## Deleting DHCP Macros

You might want to delete a macro from the DHCP service. For example, if you delete a network from the DHCP service, you can also delete the associated network macro.

You can use the `dhtadm -D -m` command or DHCP Manager to delete macros.

### ▼ How to Delete a DHCP Macro (DHCP Manager)

1. **In DHCP Manager, select the Macros tab.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Select the macro to delete.**

   The Delete Macro dialog box prompts you to confirm that you want to delete the specified macro.

3. **Select Notify DHCP Server of Change.**

   This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

4. **Click OK.**

### ▼ How to Delete a DHCP Macro (`dhtadm`)

1. **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

   For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Delete a macro by typing a command of the following format:**

   ```
   # dhtadm -D -m macroname -g
   ```

   For example, to delete the macro `bluenote`, you would type the following command:

   ```
   # dhtadm -D -m bluenote -g
   ```

# Working With DHCP Options (Task Map)

Options are keywords for network configuration parameters that the DHCP server can pass to clients. In the Solaris DHCP service, you cannot create, delete, or modify the standard DHCP options. The standard options are defined by the DHCP protocol, so the options cannot change. You can only perform tasks on options that you create for your site. For this reason, when you first set up your DHCP service, the Options tab in DHCP Manager is empty until you create options for your site.

If you create options on the DHCP server, you must also add information about the options on the DHCP client. For the Solaris DHCP client, you must edit the `/etc/dhcp/inittab` file to add entries for the new options. See the `dhcp_inittab`(4) man page for more information about this file.

If you have DHCP clients that are not Solaris clients, refer to the documentation for those clients for information about adding options or symbols. See "About DHCP Options" on page 310 for more information about options in Solaris DHCP.

You can use either DHCP Manager or the `dhtadm` command to create, modify, or delete options.

---

**Tip –** Options are called *symbols* in the DHCP literature. The `dhtadm` command and its related man page also refer to options as symbols.

---

The following task map lists tasks that you must perform to create, modify, and delete DHCP options. The task map contains links to procedures for the tasks.

| Task | Description | For Instructions |
|---|---|---|
| Create DHCP options. | Add new options for information not covered by a standard DHCP option. | "How to Create DHCP Options (DHCP Manager)" on page 400<br>"How to Create DHCP Options (`dhtadm`)" on page 401<br>"Modifying the Solaris DHCP Client's Option Information" on page 405 |
| Modify DHCP options. | Change properties of DHCP options you have created. | "How to Modify DHCP Option Properties (DHCP Manager)" on page 402<br>"How to Modify DHCP Option Properties (`dhtadm`)" on page 403 |

| Task | Description | For Instructions |
|---|---|---|
| Delete DHCP options. | Remove DHCP options that you have created. | "How to Delete DHCP Options (DHCP Manager)" on page 404<br>"How to Delete DHCP Options (`dhtadm`)" on page 404 |

Before you create DHCP options, you should be familiar with the option properties listed in the following table.

**TABLE 15–5** DHCP Option Properties

| Option Property | Description |
|---|---|
| Category | The *category* of an option must be one of the following:<br>■ Vendor – Options specific to a client's vendor platform, either hardware or software.<br>■ Site – Options specific to your site.<br>■ Extend – Newer options that have been added to the DHCP protocol, but not yet implemented as standard options in Solaris DHCP. |
| Code | The *code* is a unique number that you assign to an option. The same code cannot be used for any other option within its option category. The code must be appropriate for the option category:<br>■ Vendor – Code values of 1–254 for each vendor class<br>■ Site – Code values of 128–254<br>■ Extend – Code values of 77–127 |

**TABLE 15–5** DHCP Option Properties *(Continued)*

| Option Property | Description |
|---|---|
| Data type | The *data type* specifies what kind of data can be assigned as a value for the option. The valid data types are described in the following list.<br>■ ASCII – Text string value.<br>■ BOOLEAN – No value is associated with the Boolean data type. The presence of the option indicates that a condition is true, while the absence of the option indicates that a condition is false. For example, the `Hostname` option is Boolean. The presence of `Hostname` in a macro causes the DHCP server to look up the host name associated with the assigned address.<br>■ IP – One or more IP addresses, in dotted decimal format (*xxx.xxx.xxx.xxx*).<br>■ OCTET – Uninterpreted ASCII representation of binary data. For example, a client ID uses the octet data type. Valid characters are 0–9, A–F, and a–f. Two ASCII characters are needed to represent an 8-bit quantity.<br>■ UNUMBER8, UNUMBER16, UNUMBER32, UNUMBER64, SNUMBER8, SNUMBER16, SNUMBER32, or SNUMBER64 – Numeric value. An initial U or S indicates whether the number is unsigned or signed. The digits at the end indicate how many bits are in the number. |
| Granularity | The *granularity* specifies how many "instances" of the data type are needed to represent a complete option value. For example, a data type of IP and a granularity of 2 would mean that the option value must contain two IP addresses. |
| Maximum | The maximum number of values that can be specified for the option. For example, suppose the maximum is 2, the granularity is 2, and the data type is IP. In this case, the option value could contain a maximum of two pairs of IP addresses. |

**TABLE 15–5** DHCP Option Properties *(Continued)*

| Option Property | Description |
|---|---|
| Vendor client classes | This option is available only when the option category is Vendor. Vendor client classes identify the client classes with which the Vendor option is associated. The class is an ASCII string that represents the client machine type or operating system. For example, the class string for some models of Sun workstations is `SUNW.Sun-Blade-100`. This type of option enables you to define configuration parameters that are passed to all clients of the same class, and *only* clients of that class.<br><br>You can specify multiple client classes. Only those DHCP clients with a client class value that matches a class that you specify receive the options scoped by that class.<br><br>The client class is determined by the vendor of the DHCP client. For DHCP clients that are not Solaris clients, refer to the vendor documentation for the DHCP client for the client class.<br><br>For Solaris clients, the Vendor client class can be obtained by typing the `uname -i` command on the client. To specify the Vendor client class, substitute periods for any commas in the string returned by the `uname` command. For example, if the string `SUNW,Sun-Blade-100` is returned by the `uname -i` command, you should specify the Vendor client class as `SUNW.Sun-Blade-100`. |

# Creating DHCP Options

If you need to pass client information for which there is not already an existing option in the DHCP protocol, you can create an option. See the `dhcp_inittab`(4) man page for a list of all the options that are defined in Solaris DHCP before you create your own option.

You can use the `dhtadm -A -s` command or DHCP Manager's Create Option dialog box to create new options.

The following figure shows DHCP Manager's Create Option dialog box.

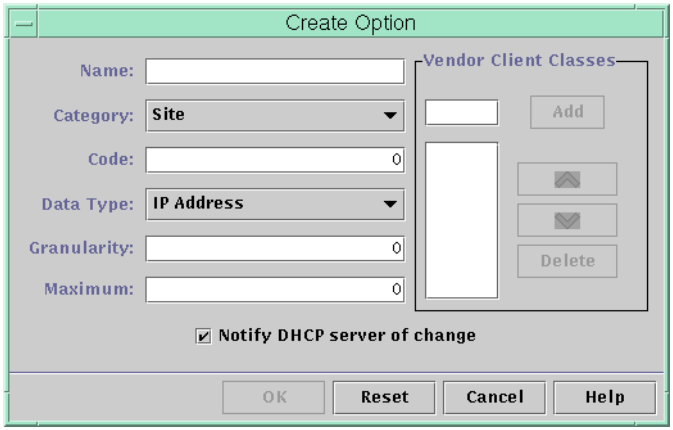



**FIGURE 15–17** Create Option Dialog Box in DHCP Manager

## ▼ How to Create DHCP Options (DHCP Manager)

1. **In DHCP Manager, select the Options tab.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Choose Create from the Edit menu.**

   The Create Options dialog box opens.

3. **Type a short descriptive name for the new option.**

   The name can contain up to 128 alphanumeric characters and spaces.

4. **Type or select values for each setting in the dialog box.**

   Refer to Table 15–5 for information about each setting, or view the DHCP Manager help.

5. **Select Notify DHCP Server of Change if you are finished creating options.**

   This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

6. **Click OK.**

   You can now add the option to macros, and assign a value to the option to pass to clients.

## ▼ How to Create DHCP Options (`dhtadm`)

1. **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

   For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Create a DHCP option by typing a command using the following format:**

   **`# dhtadm -A -s`** *option-name* **`-d '`***category,code,data-type,granularity,maximum***`' -g`**

   | | |
   |---|---|
   | *option-name* | Is an alphanumeric string of 128 characters of less. |
   | *category* | Is one of the following: `Site`, `Extend`, or `Vendor=`*list-of-classes*. *list-of-classes* is a space-separated list of vendor client classes to which the option applies. See Table 15–5 for information about how to determine the vendor client class. |
   | *code* | Is a numeric value that is appropriate to the option category, as explained in Table 15–5. |
   | *data-type* | Is specified by a keyword that indicates the type of data that is passed with the option, as explained in Table 15–5. |
   | *granularity* | Is specified as a nonnegative number, as explained in Table 15–5. |
   | *maximum* | Is a nonnegative number, as explained in Table 15–5. |

### Example 15–3 Creating a DHCP Option With `dhtadm`

The following command would create an option called `NewOpt`, which is a Site category option. The option's code is 130. The option's value can be set to a single 8-bit unsigned integer.

```
# dhtadm -A -s NewOpt -d 'Site,130,UNUMBER8,1,1' -g
```

The following command would create an option called `NewServ`, which is a Vendor category option that applies to clients whose machine type is `SUNW,Sun-Blade-100` or `SUNW,Sun-Blade-1000`. The option's code is 200. The option's value can be set to one IP address.

```
# dhtadm -A -s NewServ -d 'Vendor=SUNW.Sun-Blade-100 \
SUNW.Sun-Blade-1000,200,IP,1,1' -g
```

## Modifying DHCP Options

If you have created options for your DHCP service, you can change the properties for these options. You can use the `dhtadm -M -s` command or DHCP Manager's Option Properties dialog box to modify options.

Note that you should modify the Solaris DHCP client's option information to reflect the same modification that you make to the DHCP service. See "Modifying the Solaris DHCP Client's Option Information" on page 405.

The following figure shows DHCP Manager's Option Properties dialog box.

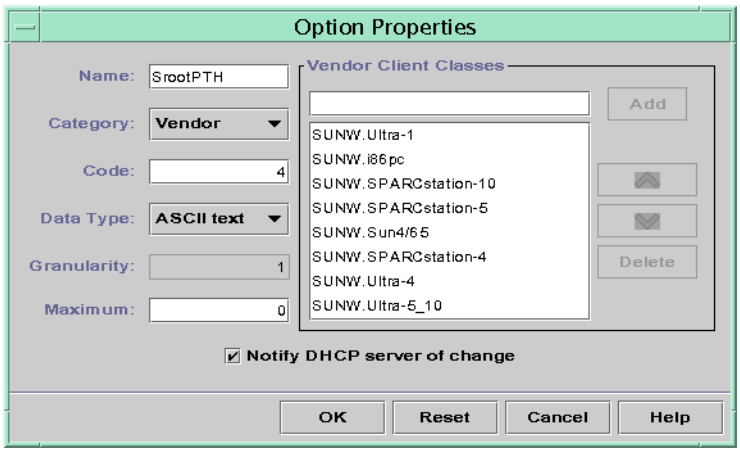



**FIGURE 15–18** Option Properties Dialog Box in DHCP Manager

## ▼ How to Modify DHCP Option Properties (DHCP Manager)

1. **In DHCP Manager, select the Options tab.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Select the option that you want to modify.**

3. **Choose Properties from the Edit menu.**

   The Option Properties dialog box opens.

4. **Edit the properties as needed.**

   See Table 15–5 for information about the properties, or view the DHCP Manager help.

5 **Select Notify DHCP Server of Change when you are finished with options.**

The change is made to the `dhcptab` table. The DHCP server is signaled to reread the `dhcptab` table to put the changes into effect.

6 **Click OK.**

## ▼ How to Modify DHCP Option Properties (`dhtadm`)

1 **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Modify an option by typing a command using the following format:**

```
# dhtadm -M -s option-name -d 'category,code,data-type,granularity,maximum' -g
```

| | |
|---|---|
| *option-name* | Specifies the name of the option that you want to change. |
| *category* | Can be `Site`, `Extend`, or `Vendor=`*list-of-classes*. *list-of-classes* is a space-separated list of vendor client classes to which the option applies. For example, `SUNW.Sun-Blade-100 SUNW.Ultra-80 SUNWi86pc`. |
| *code* | Specifies a numeric value that is appropriate to the option category, as explained in Table 15–5. |
| *data-type* | Specifies a keyword that indicates the type of data that is passed with the option, as explained in Table 15–5. |
| *granularity* | Is a nonnegative number, as explained in Table 15–5. |
| *maximum* | Is a nonnegative number, as explained in as explained in Table 15–5. |

Note that you must specify all of the DHCP option properties with the `-d` switch, not just the properties that you want to change.

**Example 15–4** Modifying a DHCP Option With `dhtadm`

The following command would modify an option called `NewOpt`. The option is a Site category option. The option's code is 135. The option's value can be set to a single 8-bit unsigned integer.

```
# dhtadm -M -s NewOpt -d 'Site,135,UNUMBER8,1,1'
```

The following command would modify an option called `NewServ`, which is a Vendor category option. The option now applies to clients whose machine type is `SUNW,Sun-Blade-100` or `SUNW,i86pc`. The option's code is 200. The option's value can be set to one IP address.

```
# dhtadm -M -s NewServ -d 'Vendor=SUNW.Sun-Blade-100 \
SUNW.i86pc,200,IP,1,1' -g
```

## Deleting DHCP Options

You cannot delete standard DHCP options. However, if you have defined options for your DHCP service, you can delete these options by using DHCP Manager or the `dhtadm` command.

### ▼ How to Delete DHCP Options (DHCP Manager)

1. **In DHCP Manager, select the Options tab.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Select the option that you want to delete.**

3. **Choose Delete from the Edit menu.**

   The Delete Option dialog box opens.

4. **Select Notify DHCP Server of Change if you are finished deleting options.**

   This selection tells the DHCP server to reread the `dhcptab` table to put the change into effect immediately after you click OK.

5. **Click OK.**

### ▼ How to Delete DHCP Options (`dhtadm`)

1. **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

   For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Delete a DHCP option by typing a command using the following format:**

```
# dhtadm -D -s option-name -g
```

## Modifying the Solaris DHCP Client's Option Information

If you add a new DHCP option to your DHCP server, you must add a complementary entry to each DHCP client's option information. If you have a DHCP client that is not a Solaris DHCP client, refer to that client's documentation for information about adding options or symbols.

On a Solaris DHCP client, you must edit the `/etc/dhcp/inittab` file and add an entry for each option that you add to the DHCP server. If you later modify the option on the server, you must also modify the entry in the client's `/etc/dhcp/inittab` file.

Refer to the `dhcp_inittab`(4) man page for detailed information about the syntax of the `/etc/dhcp/inittab` file.

---

**Note –** If you added DHCP options to the `dhcptags` file in a previous Solaris release, you must add the options to the `/etc/dhcp/inittab` file. See "DHCP Option Information" on page 471 for more information.

---

# Supporting Solaris Network Installation With the DHCP Service

You can use DHCP to install the Solaris Operating System on certain client systems on your network. Only `sun4u`-based systems and x86 systems that meet the hardware requirements for running the Solaris OS can use this feature. For information about using DHCP to automatically configure client systems for the network as they boot, see Chapter 2, "Preconfiguring System Configuration Information (Tasks)," in *Solaris Express Installation Guide: Network-Based Installations*.

DHCP also supports Solaris client systems that boot and install remotely from servers across a wide area network (WAN) using HTTP. This method of remote booting and installing is called the *WAN boot installation* method. Using WAN boot, you can install the Solaris OS on SPARC based systems over a large public network where the network infrastructure might be untrustworthy. You can use WAN boot with security features to protect data confidentiality and installation image integrity.

Before you can use DHCP for booting and installing client systems remotely using WAN boot, the DHCP server must be configured to supply the following information to clients:

- The proxy server's IP address
- The location of the `wanboot-cgi` program

For details about configuring the DHCP server to provide this information, see Chapter 2, "Preconfiguring System Configuration Information (Tasks)," in *Solaris Express Installation Guide: Network-Based Installations*. For information about booting and installing client systems with a DHCP server across a WAN, see Chapter 9, "WAN Boot (Overview)," in *Solaris Express Installation Guide: Network-Based Installations*.

For information about supporting diskless clients, see "Supporting Remote Boot and Diskless Boot Clients (Task Map)" on page 406.

# Supporting Remote Boot and Diskless Boot Clients (Task Map)

The Solaris DHCP service can support Solaris client systems that mount their operating system files remotely from another machine (the OS server). Such clients are often called *diskless clients*. Diskless clients can be thought of as persistent remote boot clients. Each time a diskless client boots, the client must obtain the name and IP address of the server that hosts the client's operating system files. The diskless client can then boot remotely from those files.

Each diskless client has its own root partition on the OS server, which is shared to the client host name. The DHCP server must always return the same IP address to a diskless client. That address must remain mapped to the same host name in the name service, such as DNS. When a diskless client receives a consistent IP address, the client uses a consistent host name, and can access its root partition on the OS server.

In addition to providing the IP address and host name, the DHCP server can supply the location of the diskless client's operating system files. However, you must create options and macros to pass the information in a DHCP message packet.

The following task map lists the tasks required to support diskless clients or any other persistent remote boot clients. The task map also provides links to procedures to help you carry out the tasks.

| Task | Description | For Instructions |
|---|---|---|
| Set up OS services on a Solaris server. | Use the `smosservice` command to create operating system files for clients. | Chapter 7, "Managing Diskless Clients (Tasks)," in *System Administration Guide: Basic Administration*<br><br>Also, see the `smosservice`(1M) man page. |

| Task | Description | For Instructions |
|---|---|---|
| Set up the DHCP service to support network boot clients. | Use DHCP Manager or the `dhtadm` command to create new Vendor options and macros, which the DHCP server can use to pass booting information to the clients.<br>If you already created the options for network install clients, you need only to create macros for the Vendor client types of the diskless clients. | Chapter 2, "Preconfiguring System Configuration Information (Tasks)," in *Solaris Express Installation Guide: Network-Based Installations* |
| Assign reserved IP addresses to the diskless clients. | Use DHCP Manager to mark address as reserved, or use the `pntadm` command to mark addresses as `MANUAL` for diskless clients. | "Assigning a Reserved IP Address to a DHCP Client" on page 383 |
| Set up diskless clients for OS service. | Use the `smdiskless` command to add operating system support on the OS server for each client. Specify the IP addresses that you reserved for each client. | Chapter 7, "Managing Diskless Clients (Tasks)," in *System Administration Guide: Basic Administration*<br>Also, see the `smdiskless`(1M) man page. |

# Setting Up DHCP Clients to Receive Information Only (Task Map)

In some networks, you might want the DHCP service to provide only configuration information to clients. Client systems that need information, not leases, can use the DHCP client to issue an `INFORM` message. The `INFORM` message asks the DHCP server to send the appropriate configuration information to the client.

You can set up the Solaris DHCP server to support clients that need information only. You need to create an empty network table that corresponds to the network that is hosting the clients. The table must exist so that the DHCP server can respond to clients from that network.

The following task map lists the tasks required to support information-only clients. The task map also includes links to procedures to help you carry out the tasks.

| Task | Description | For Instructions |
|---|---|---|
| Create an empty network table. | Use DHCP Manager or the `pntadm` command to create a network table for the information-only clients' network. | "Adding DHCP Networks" on page 360 |

| Task | Description | For Instructions |
|---|---|---|
| Create macros to contain information that is needed by clients. | Use DHCP Manager or the `dhtadm` command to create macros to pass the required information to clients. | "Creating DHCP Macros" on page 392 |
| Have the DHCP client issue an `INFORM` message. | Use the `ifconfig` *int* `dhcp inform` command to make the DHCP client issue an `INFORM` message. | "DHCP Client Startup" on page 421<br>"`ifconfig` Command Options Used With the DHCP Client" on page 426<br>`ifconfig`(1M)man page |

# Converting to a New DHCP Data Store

Solaris DHCP provides a utility to convert the DHCP configuration data from one data store to another data store. Several reasons might exist for converting to a new data store. For example, you might have more DHCP clients, requiring higher performance or higher capacity from the DHCP service. You also might want to share the DHCP server duties among multiple servers. See "Choosing the DHCP Data Store" on page 320 for a comparison of the relative benefits and drawbacks of each type of data store.

---

**Note –** If you upgraded from a Solaris release that is older than the Solaris 8 7/01 release, you should read this note.

When you run any Solaris DHCP tool after Solaris installation, you are prompted to convert to the new data store. The conversion is required because the format of the data stored in both files and NIS+ changed in the Solaris 8 7/01 release. If you do not convert to the new data store, the DHCP server continues to read the old data tables. However, the server can only extend leases for existing clients. You cannot register new DHCP clients or use DHCP management tools with the old data tables.

---

The conversion utility is also useful for sites that are converting from a Sun provided data store to a third-party data store. The conversion utility looks up entries in the existing data store and adds new entries that contain the same data to the new data store. Data store access is implemented in separate modules for each data store. This modular approach enables the conversion utility to convert DHCP data from any data store format to any other data store format. Each data store must have a module that the DHCP service can use. See *Solaris DHCP Service Developer's Guide* for more information about how to write a module to support a third-party data store.

The data store conversion can be accomplished with DHCP Manager through the Data Store Conversion wizard, or with the `dhcpconfig -C` command.

The initial dialog box of the Data Store Conversion wizard is shown in the following figure.

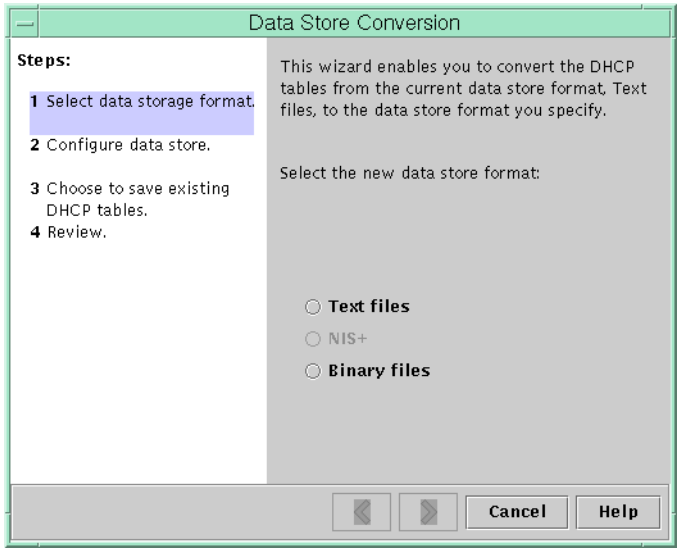



**FIGURE 15–19** Data Store Conversion Wizard Dialog Box in DHCP Manager

Before the conversion begins, you must specify whether to save the old data store's tables (`dhcptab` and network tables). The conversion utility then stops the DHCP server, converts the data store, and restarts the server when the conversion has completed successfully. If you did not specify to save the old tables, the utility deletes the tables after determining the conversion is successful. The process of converting can be time-consuming. The conversion runs in the background with a meter to inform you of its progress.

## ▼ How to Convert the DHCP Data Store (DHCP Manager)

1. **In DHCP Manager, choose Convert Data Store from the Service menu.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

   The Data Store Conversion wizard opens.

2. **Answer the wizard's prompts.**

   If you have trouble providing the requested information, click Help to view detailed information about each dialog box.

3. **Review your selections, and then click Finish to convert the data store.**

   The DHCP server restarts when the conversion is complete. The server immediately uses the new data store.

### ▼ How to Convert the DHCP Data Store (`dhcpconfig -C`)

1 **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Convert the data store by typing a command of the following format:**

```
# /usr/sbin/dhcpconfig -C -r resource -p path
```

*resource* is the new data store type, such as `SUNWbinfiles`

*path* is the path to the data, such as `/var/dhcp`

Note that if you want to keep the original data in the old data store after the conversion, specify the `-k` option. For example, to convert your data store to `SUNWbinfiles` and save the old data store, you would type:

```
# /usr/sbin/dhcpconfig -C -r SUNWbinfiles -p /var/dhcp -k
```

See the `dhcpconfig`(1M) man page for more information about the `dhcpconfig` utility.

# Moving Configuration Data Between DHCP Servers (Task Map)

DHCP Manager and the `dhcpconfig` utility enable you to move some or all the DHCP configuration data from one Solaris DHCP server to another server. You can move entire networks and all the IP addresses, macros, and options associated with the networks. Alternatively, you can select specific IP addresses, macros, and options to move. You can also copy macros and options without removing the macros and options from the first server.

You might want to move data if you are going to do any of the following tasks:

- Add a server to share DHCP duties.
- Replace the DHCP server's system.
- Change the path for the data store, while still using the same data store.

The following task map identifies the procedures that you must perform when you move DHCP configuration data.

| Task | Description | For Instructions |
|---|---|---|
| 1. Export the data from the first server. | Select the data that you want to move to another server, and create a file of exported data. | "How to Export Data From a DHCP Server (DHCP Manager)" on page 412<br>"How to Export Data From a DHCP Server (`dhcpconfig -X`)" on page 413 |
| 2. Import the data to the second server. | Copy exported data to another DHCP server's data store. | "How to Import Data on a DHCP Server (DHCP Manager)" on page 414<br>"How to Import Data on a DHCP Server (`dhcpconfig -I`)" on page 414 |
| 3. Modify the imported data for the new server environment. | Change server-specific configuration data to match the new server's information. | "How to Modify Imported DHCP Data (DHCP Manager)" on page 415<br>"How to Modify Imported DHCP Data (`pntadm`, `dhtadm`)" on page 416 |

In DHCP Manager, you use the Export Data wizard and the Import Data wizard to move the data from one server to the other server. You then modify macros in the Macros tab. The following figures show the initial dialog boxes for the wizards.

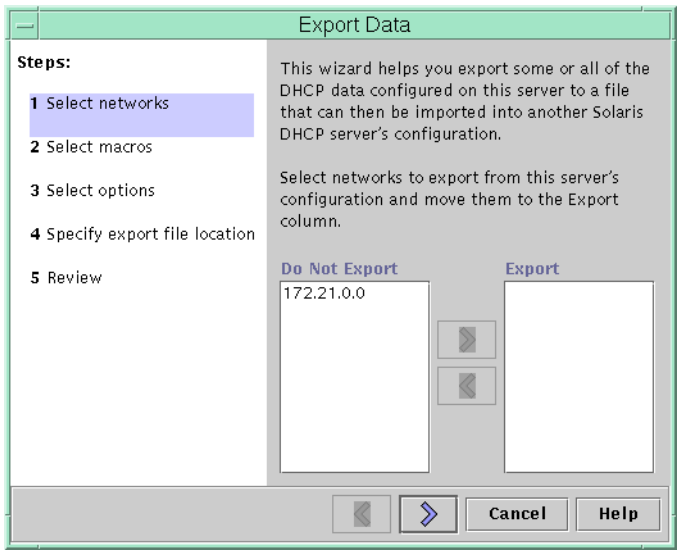



**FIGURE 15–20** Export Data Wizard Dialog Box in DHCP Manager

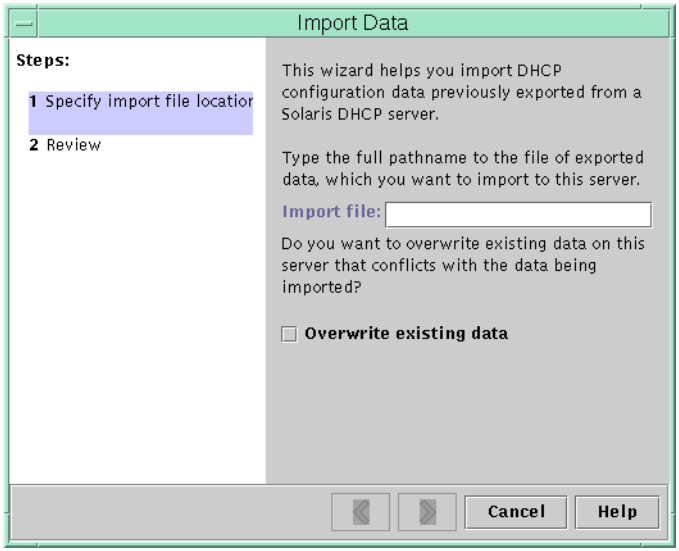



FIGURE 15–21 Import Data Wizard Dialog Box in DHCP Manager

## ▼ How to Export Data From a DHCP Server (DHCP Manager)

1. **Start DHCP Manager on the server from which you want to move or copy data.**

   See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2. **Choose Export Data from the Service menu.**

   The Export Data wizard opens as shown in Figure 15–20.

3. **Answer the wizard's prompts.**

   If you have difficulty, click Help for detailed information about the prompts.

4. **Move the export file to a file system that is accessible to the DHCP server that must import the data.**

**See Also** Import the data as described in "How to Import Data on a DHCP Server (DHCP Manager)" on page 414.

## ▼ How to Export Data From a DHCP Server (`dhcpconfig -X`)

1. **Log in to the server from which you want to move or copy data.**

2. **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

   For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

3. **Export the data.**

   You can export all of the DHCP data, or specific parts of the data.

   - **To export specific addresses, macros, and options, type a command that uses the following format:**

     ```
     # dhcpconfig -X filename -a network-addresses -m macros -o options
     ```

     *filename* is the full path name that you want to use to store the compressed exported data. You specify particular network addresses, DHCP macros, and DHCP options in comma-separated lists. The following example shows how to export specific networks, macros, and options.

     ```
     # dhcpconfig -X /var/dhcp/0dhcp1065_data \
     -a 10.63.0.0,10.62.0.0 \
     -m 10.63.0.0,10.62.0.0,SUNW.Sun-Blade-100 -o Sterm
     ```

   - **To export all DHCP data, type a command that uses the `ALL` keyword.**

     ```
     # dhcpconfig -X filename -a ALL -m ALL -o ALL
     ```

     *filename* is the full path name that you want to use to store the compressed exported data. The keyword `ALL` can be used with the command options to export all the network addresses, macros, or options. The following example shows how to use the `ALL` keyword.

     ```
     # dhcpconfig -X /var/dhcp/dhcp1065_data -a ALL -m ALL -o ALL
     ```

---

**Tip** – You can omit the export of a particular kind of data by not specifying the `dhcpconfig` command option for that type of data. For example, if you do not specify the `-m` option, no DHCP macros are exported.

---

See the `dhcpconfig`(1M) man page for more information about the `dhcpconfig` command.

4 **Move the export file to a location that is accessible to the server that must import the data.**

**See Also** Import the data as described in "How to Import Data on a DHCP Server (`dhcpconfig -I`)" on page 414.

## ▼ How to Import Data on a DHCP Server (DHCP Manager)

1 **Start DHCP Manager on the server to which you want to move data that you previously exported from a DHCP server.**

See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2 **Choose Import Data from the Service menu.**

The Import Data wizard opens, as shown in Figure 15–21.

3 **Answer the wizard's prompts.**

If you have difficulty, click Help for detailed information about the prompts.

4 **Modify the imported data, if necessary.**

See "How to Modify Imported DHCP Data (DHCP Manager)" on page 415

## ▼ How to Import Data on a DHCP Server (`dhcpconfig -I`)

1 **Log in to the server to which you want to import the data.**

2 **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

3 **Import the data by typing a command of the following format:**

```
# dhcpconfig -I filename
```

*filename* is the name of the file that contains the exported data.

4 **Modify the imported data, if necessary.**

See "How to Modify Imported DHCP Data (`pntadm`, `dhtadm`)" on page 416.

## ▼ How to Modify Imported DHCP Data (DHCP Manager)

1 **Start DHCP Manager on the server to which you imported data.**

See "How to Start and Stop DHCP Manager" on page 342 for information about DHCP Manager.

2 **Examine imported data for network-specific information that needs modification.**

For example, if you moved networks, you must open the Addresses tab and change the owning server of addresses in the imported networks. You might also need to open the Macros tab to specify the correct domain names for NIS, NIS+ or DNS in some macros.

3 **Open the Addresses, tab and select a network that you imported.**

4 **To select all the addresses, click the first address, press and hold the Shift key, and click the last address.**

5 **From the Edit menu, choose Properties.**

The Modify Multiple Addresses dialog box opens.

6 **At the Managing Server prompt, select the new server's name.**

7 **At the Configuration Macro prompt, select the macro that should be used for all clients on this network, and then click OK.**

8 **Open the Macros tab.**

9 **Use the Find button to locate the options that are likely to need modified values.**

The Find button is located at the bottom of the window.

`DNSdmain`, `DNSserv`, `NISservs`, `NIS+serv`, and `NISdmain` are examples of options that might need modification on the new server.

10 **Change the options in the appropriate macros.**

See "How to Modify DHCP Option Properties (DHCP Manager)" on page 402 for the procedure for changing options.

## ▼ How to Modify Imported DHCP Data (`pntadm`, `dhtadm`)

1. **Log in to the server to which you imported data.**

2. **Become superuser or assume a role or user name that is assigned to the DHCP Management profile.**

   For more information about the DHCP Management profile, see "Setting Up User Access to DHCP Commands" on page 343.

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

3. **Examine the network tables for data that needs to be modified.**

   If you moved networks, use the `pntadm -P` *network-address* command to print out the network tables for the networks you moved.

4. **Modify IP address information by using the `pntadm` command.**

   You might need to change the owning server and the configuration macro for imported addresses. For example, to change the owning server (`10.60.3.4`) and macro (`dhcpsrv-1060`) for address `10.63.0.2`, you would use the following command:

   ```
   pntadm -M 10.63.0.2 -s 10.60.3.4 -m dhcpsrv-1060 10.60.0.0
   ```

   If you have a large number of addresses, you should create a script file that contains commands to modify each address. Execute the script with the `pntadm -B` command, which runs `pntadm` in batch mode. See the `pntadm`(1M) man page.

5. **Examine the `dhcptab` macros for options with values that need modification.**

   Use the `dhtadm -P` command to print the entire `dhcptab` table to your screen. Use `grep` or some other tool to search for options or values that you might want to change.

6. **Modify options in macros, if necessary, by using the `dhtadm -M` command.**

   For example, you might need to modify some macros to specify the correct domain names and servers for NIS, NIS+ or DNS. For example, the following command changes the values of `DNSdmain` and `DNSserv` in the macro `mymacro`:

   ```
   dhtadm -M -m mymacro -e 'DNSserv=dnssrv2:DNSdmain=example.net' -g
   ```

◆ ◆ ◆ CHAPTER 16

# Configuring and Administering the DHCP Client

This chapter discusses the Dynamic Host Configuration Protocol (DHCP) client that is part of the Solaris Operating System. The chapter explains how the client's DHCPv4 and DHCPv6 protocols work, and how you can affect the behavior of the client.

One protocol, DHCPv4, has long been part of the Solaris Operating System (Solaris OS), and enables DHCP servers to pass configuration parameters such as IPv4 network addresses to IPv4 nodes.

The other protocol, DHCPv6, enables DHCP servers to pass configuration parameters such as IPv6 network addresses to IPv6 nodes. DHCPv6 is a stateful counterpart to "IPv6 Stateless Address Autoconfiguration" (RFC 2462), and can be used separately or concurrently with the stateless to obtain configuration parameters.

This chapter contains the following information:


- "About the Solaris DHCP Client" on page 417
- "Enabling and Disabling a Solaris DHCP Client" on page 425
- "DHCP Client Administration" on page 426
- "DHCP Client Systems With Multiple Network Interfaces" on page 429
- "DHCPv4 Client Host Names" on page 430
- "DHCP Client Systems and Name Services" on page 431
- "DHCP Client Event Scripts" on page 436


## About the Solaris DHCP Client

The Solaris DHCP client is the `dhcpagent` daemon, part of the Solaris OS. When you install the Solaris OS, you are prompted to use DHCP to configure network interfaces. If you specify Yes for DHCPv4, then that protocol is enabled on your system during Solaris installation. There are no install time options specifically for DHCPv6. A related question, though, is about IPv6. If you enable IPv6, then DHCPv6 is also enabled on a local network that supports DHCPv6.

You do not need to do anything else with the Solaris client to use DHCP. The DHCP server's configuration determines what information is given to DHCP client systems that use the DHCP service.

If a client system is already running the Solaris OS, but not using DHCP, you can reconfigure the client system to use DHCP. You can also reconfigure a DHCP client system so that it stops using DHCP and uses static network information that you provide. See "Enabling and Disabling a Solaris DHCP Client" on page 425 for more information.

## DHCPv6 Server

There is no DHCPv6 server available through Sun Microsystems for the Solaris OS. Servers available from third parties are compatible with Sun's DHCPv6, and if there is a DHCPv6 server on the network, Sun's DHCPv6 client will use it.

See "Solaris DHCP Server" on page 304for information on the Sun DHCPv4 server.

## Differences Between DHCPv4 and DHCPv6

The two major differences between DHCPv4 and DHCPv6 are the following:

- **The administrative model**
  - DHCPv4–The administrator enables DHCP for each interface. Administration is on a per-logical interface basis.
  - DHCPv6–Explicit configuration is not necessary. This protocol is enabled on a given physical interface.
- **Protocol details**
  - DHCPv4–The DHCP server supplies the subnet mask for each address. A hostname option sets the system-wide node name.
  - DHCPv6–The subnet mask is supplied by Router Advertisements, not the DHCPv6 server. There is no DHCPv6 hostname option.

## The Administrative Model

**DHCPv4** requires explicit client configuration. You must set up the DHCPv4 system for addressing when desired, and this is typically done during initial system installation or dynamically through the use of `ifconfig`(1M) options.

**DHCPv6** does not require explicit client configuration. Instead, using DHCP is a property of the network, and the signal to use it is carried in Router Advertisement messages from local routers. The DHCP client automatically creates and destroys logical interfaces as needed.

The DHCPv6 mechanism is very similar administratively to the existing IPv6 stateless (automatic) address configuration. For stateless address configuration, you would set a flag on the local router to indicate that, for a given set of prefixes, each client should automatically configure an address on its own by using the advertised prefix plus a local interface token or random number. For DHCPv6, the same prefixes are required, but the addresses are acquired and managed through a DHCPv6 server instead of being assigned "randomly."

### MAC Address and Client ID

**DHCPv4** uses the MAC address and an optional Client ID to identify the client for purposes of assigning an address. Each time the same client arrives on the network, it gets the same address, if possible.

**DHCPv6** uses basically the same scheme, but makes the Client ID mandatory and imposes structure on it. The Client ID in DHCPv6 consists of two parts: a DHCP Unique Identifier (DUID) and an Identity Association Identifier (IAID). The DUID identifies the client **system** (rather than just an interface, as in DHCPv4), and the IAID identifies the interface on that system.

As described in RFC 3315, an identity association is the means used for a server and a client to identify, group, and manage a set of related IPv6 addresses. A client must associate at least one distinct IA with each of its network interfaces, and then uses the assigned IAs to obtain configuration information from a server for that interface. For additional information about IAs, see the next section, "Protocol Details."

DUID+IAID can also be used with DHCPv4. These can be concatenated together unambiguously so that they can serve as the Client ID. For compatibility reasons, this is not done for regular IPv4 interfaces. However, for logical interfaces ("hme0:1"), DUID+IAID is used if no Client ID is configured.

Unlike IPv4 DHCP, DHCPv6 does not provide a "client name" option, so there is no way to name your systems based on DHCPv6 alone. Instead, if you need to know the DNS name that goes with an address provided by DHCPv6, use DNS reverse-resolution (address-to-name query via the `getaddrinfo(3SOCKET)` function) to find the corresponding name information. One implication of this is that if you are using only DHCPv6 and want a node to have a specific name, you must set `/etc/nodename` on your system.

## Protocol Details

With DHCPv4, the DHCP server supplies the subnet mask to be used with the assigned address. With DHCPv6, the subnet mask (also known as "prefix length") is assigned by the Router Advertisements, and is not controlled by the DHCP server.

DHCPv4 carries a Hostname option that is used to set the system-wide node name. DHCPv6 has no such option.

To configure a Client ID for DHCPv6 you must specify a DUID, rather than allowing the system to choose one automatically. You can do this globally for the daemon, or on a per-interface basis. Use the following format to set the global DUID (note the initial dot):

```
.v6.CLIENT_ID=<DUID>
```

To set a particular interface to use a given DUID (and make the system appear to be multiple independent clients to a DHCPv6 server):

```
hme0.v6.CLIENT ID=<DUID>
```

Each Identity Association (IA) holds one type of address. For example, an identity association for temporary addresses (IA_TA) holds temporary addresses, while an identity association for non-temporary addresses (IA_NA), carries assigned addresses that are permanent. The version of DHCPv6 described in this guide provides only IA_NA associations.

The Solaris OS assigns exactly one IAID to each interface, on demand, and the IAID is stored in a file in the root file system so that it remains constant for the life of the machine.

## Logical Interfaces

In the DHCPv4 client, each logical interface is independent and is an administrative unit. In addition to the zeroth logical interface (which defaults to the interface MAC address as an identifier), the user may configure specific logical interfaces to run DHCP by specifying a CLIENT_ID in the `dhcpagent` configuration file. For example:

```
hme0:1.CLIENT_ID=orangutan
```

DHCPv6 works differently. The zeroth logical interface on an IPv6 interface, unlike IPv4, is always a link-local. A link-local is used to automatically assign an IP address to a device in an IP network when there is no other assignment method available, such as a DHCP server. The zeroth logical interface cannot be under DHCP control, so although DHCPv6 is run on the zeroth logical interface (known, also, as the “physical” interface), it assigns addresses only on non-zero logical interfaces.

In response to a DHCPv6 client request, the DHCPv6 server returns a list of addresses for the client to configure.

## Option Negotiation

In DHCPv6 there is an Option Request Option, which provides a hint to the server of what the client prefers to see. If all possible options were sent from the server to the client, so much information could be sent that some of it would have to be dropped on the way to the client. The server might use the hint to choose among the options to include in the reply. Alternatively, the

server could ignore the hint and choose other items to include. On Solaris OS, for example, the preferred options might include the Solaris DNS address domain or the NIS address domain, but would probably not include the net bios server.

The same type of hint is also provided for DHCPv4, but without the special Option Request Option. Instead DHCPv4 uses the `PARAM_REQUEST_LIST` in `/etc/default/dhcpagent`.

## Configuration Syntax

Configure the DHCPv6 client in much the same way as the existing DHCPv4 client, using `/etc/default/dhcpagent`.

The syntax is augmented with a ".v6" marker between the interface name (if any) and the parameter to be configured. For example, the global IPv4 option request list is set like this:

```
PARAM_REQUEST_LIST=1,3,6,12,15,28,43
```

An individual interface can be configured to omit the hostname option like this:

```
hme0.PARAM_REQUEST_LIST=1,3,6,15,28,43
```

To set a global request list for DHCPv6, note the leading dot:

```
.v6.PARAM_REQUEST_LIST=23,24
```

Or, to set an individual interface, follow this example:

```
hme0.v6.PARAM_REQUEST_LIST=21,22,23,24
```

For reference, here is an actual `/etc/default/dhcpagent` file for DHCPv6 configuration:

```
# The default DHCPv6 parameter request list has preference (7), unicast (12),
# DNS addresses (23), DNS search list (24), NIS addresses (27), and
# NIS domain (29).  This may be changed by altering the following parameter-
# value pair.  The numbers correspond to the values defined in RFC 3315 and
# the IANA dhcpv6-parameters registry.
.v6.PARAM_REQUEST_LIST=7,12,23,24,27,29
```

## DHCP Client Startup

In most cases, there is nothing you need to do for DHCPv6 client startup. The `in.ndpd` daemon starts up DHCPv6 automatically when it is needed. You might need to touch `/etc/hostname6.$IFNAME` to configure an interface to be plumbed for IPv6 at boot time. However, the installer already does this if you enable IPv6 on your system at install time.

For DHCPv4, however, you must request the client startup, if that was not done during Solaris installation. See "How to Enable the Solaris DHCP Client" on page 425.

The `dhcpagent` daemon obtains configuration information that is needed by other processes involved in booting the system. For this reason, the system startup scripts start `dhcpagent` early in the boot process and wait until the network configuration information from the DHCP server arrives.

Although the default is to run DHCPv6, you can choose to not have DHCPv6 run. After DHCPv6 starts running, you can stop it with the `ifconfig` command. You can also disable DHCPv6 so that it does not start on reboot, by modifying the `/etc/inet/ndpd.conf` file.

For example, to immediately shut down DHCPv6 on the interface named "hme0."

```
ex# echo ifdefault StatefulAddrConf false >> /etc/inet/ndpd.conf
ex# pkill -HUP -x in.ndpd
ex# ifconfig hme0 inet6 dhcp release
```

The presence of the file `/etc/dhcp.`*interface* (for example, `/etc/dhcp.ce0` on a Sun Fire™ 880 system) indicates to the startup scripts that DHCPv4 is to be used on the specified interface. Upon finding a `dhcp.`*interface* file, the startup scripts start `dhcpagent`.

After startup, `dhcpagent` waits until it receives instructions to configure a network interface. The startup scripts issue the `ifconfig` *interface* `dhcp start` command, which instructs `dhcpagent` to start DHCPv4 as described in "How DHCP Works" on page 301. If commands are contained within the `dhcp.`*interface* file, they are appended to the `dhcp start` option of `ifconfig`. See the `ifconfig`(1M) man page for more information about options used with the `ifconfig` *interface* `dhcp` command.

# DHCPv6 Communication

Unlike DHCPv4, which is invoked by manual configuration, DHCPv6 is invoked by Router Advertisements (RAs). Depending on how the router is configured, the system automatically invokes DHCPv6 on the interface on which the Router Advertisement message was received and uses DHCP to get an address and other parameters, or the system requests only data other than an address (for example, DNS servers) with DHCPv6.

The `in.ndpd` daemon receives the Router Advertisement message. It does this automatically on all interfaces plumbed for IPv6 on the system. When `in.ndpd` sees an RA that specifies that DHCPv6 should run, it invokes it.

To prevent `in.ndpd` from starting up DHCPv6, you can change the `/etc/inet/ndpd.conf` file.

You can also stop DHCPv6 after it starts by using one of the following versions of `ifconfig`:

```
ifconfig <interface> inet6 dhcp drop
```

or:

```
ifconfig <interface> inet6 dhcp release
```

# How DHCP Client Protocols Manage Network Configuration Information

DHCPv4 and DHCPv6 client protocols manage network configuration information in different ways. The key difference is that with DHCPv4 the negotiation is for the lease of a single address and some options to go with it. With DHCPv6, the negotiation is over a batch of addresses and a batch of options.

For background information on the interaction between DHCPv4 client and server, see Chapter 12, "About Solaris DHCP (Overview)."

## How the DHCPv4 Client Manages Network Configuration Information

After the information packet is obtained from a DHCP server, `dhcpagent` configures the network interface and brings up the interface. The daemon controls the interface for the duration of the lease time for the IP address, and maintains the configuration data in an internal table. The system startup scripts use the `dhcpinfo` command to extract configuration option values from the internal table. The values are used to configure the system and enable it to communicate on the network.

The `dhcpagent` daemon waits passively until a period of time elapses, usually half the lease time. The daemon then requests an extension of the lease from a DHCP server. If the system notifies `dhcpagent` that the interface is down or that the IP address has changed, the daemon does not control the interface until instructed by the `ifconfig` command to do so. If `dhcpagent` finds that the interface is up and the IP address has not changed, the daemon sends a request to the server for a lease renewal. If the lease cannot be renewed, `dhcpagent` takes down the interface at the end of the lease time.

Each time `dhcpagent` performs an action related to the lease, the daemon looks for an executable file called `/etc/dhcp/eventhook`. If an executable file with this name is found, `dhcpagent` invokes the executable. See "DHCP Client Event Scripts" on page 436 for more information about using the event executable.

## How the DHCPv6 Client Manages Network Configuration Information

DHCPv6 communication between client and server begins with the client sending out a Solicit message, to locate servers. In response, all servers available for DHCP service send an Advertise message. The server message contains multiple IA_NA (Identity Association Non-Temporary Address) records plus other options (such as DNS server addresses) that the server can supply.

A client can request particular addresses (and multiples of them) by setting up its own IA_NA/IAADDR records in its Request message. A client typically requests specific addresses if it has old addresses recorded and it would like the server to provide the same ones, if possible. Regardless of what the client does (even if it requests no addresses at all), the server can supply any number of addresses to the client for a single DHCPv6 transaction.

This is a the message dialog that takes place between the clients and servers.

- A client sends a Solicit message to locate servers.
- Servers send an Advertise message to indicate they are available for DHCP service.
- A client sends a Request message to request configuration parameters, including IP addresses, from servers with the greatest preference values. Server preference values are set by the administrator and extend from 0, at the lowest end, to 255 at the highest.
- The server sends a Reply message that contains the address leases and configuration data.

If the preference value in the Advertise message is 255, the DHCPv6 client immediately selects that server. If the most preferred server does not respond, or fails to give a successful Reply to the Request message, then the client continues looking for less-preferred servers (in order) until there are no more Advertise messages on hand. At that point, the client starts over by again sending Solicit messages.

The chosen server sends a Reply message containing assigned addresses and configuration parameters in response to a Solicit or Request message.

## DHCP Client Shutdown

At shutdown, the client sends a Release message to the server that assigned addresses to the client, to indicate that the client will no longer use one or more of the assigned addresses. When the DHCPv4 client system shuts down normally, `dhcpagent` writes the current configuration information to the file `/etc/dhcp/`*interface*`.dhc`, or for DHCPv6, to `/etc/dhcp/`*interface*`.dh6`. By default, the lease is saved rather than released, so the DHCP server does not know that the IP address is not in active use, which enables the client to easily regain the address on next boot. This default action is the same as the `ifconfig <interface> dhcp drop` command.

If the lease in that file is still valid when the system reboots, `dhcpagent` sends an abbreviated request to use the same IP address and network configuration information. For DHCPv4, this is the Request message. For DHCPv6, the message is Confirm.

If the DHCP server permits this request, `dhcpagent` can use the information that it wrote to disk when the system shut down. If the server does not permit the client to use the information, `dhcpagent` initiates the DHCP protocol sequence described in "How DHCP Works" on page 301. As a result, the client obtains new network configuration information.

# Enabling and Disabling a Solaris DHCP Client

To enable the DHCP client on a system that is already running the Solaris OS and is not using DHCP, you must first unconfigure the system. When the system boots, you must issue some commands to set up the system and enable the DHCP client.

---

**Note –** In many deployments it is common practice to have crucial parts of the infrastructure set up with static IP addresses, rather than using DHCP. Determining which devices on your network, for example routers and certain servers, should be client and which should not, is beyond the scope of this guide.

---

## ▼ How to Enable the Solaris DHCP Client

This procedure is necessary only if DHCPv4 was not enabled during Solaris installation. It is never necessary for DHCPv6.

1. **Become superuser on the client system.**

2. **If this system uses preconfiguration instead of interactive configuration, edit the `sysidcfg` file. Add the `dhcp` subkey to the `network_interface` keyword in the `sysidcfg` file.**

   For example, `network_interface=hme0 {dhcp}`. See the `sysidcfg`(4) man page for more information.

3. **Unconfigure and shut down the system.**

   ```
   # sys-unconfig
   ```

   See the `sys-unconfig`(1M) man page for more information about the configuration information that is removed by this command.

4. **Reboot the system after shutdown is complete.**

   If the system uses preconfiguration, the `dhcp` subkey in the `sysidcfg` file configures the system to use the DHCP client as the system boots.

   If the system does not use preconfiguration, you are prompted for system configuration information by `sysidtool` programs when the system reboots. See the `sysidtool`(1M) man page for more information.

5. **When prompted to use DHCP to configure network interfaces, specify Yes.**

## ▼ How to Disable a Solaris DHCP Client

1. **Become superuser on the client system.**

2 **If you used a `sysidcfg` file to preconfigure the system, remove the `dhcp` subkey from the `network_interface` keyword.**

3 **Unconfigure and shut down the system.**

```
# sys-unconfig
```

See the `sys-unconfig`(1M) man page for more information about the configuration information that is removed by this command.

4 **Reboot the system after shutdown is complete.**

If the system uses preconfiguration, you are not prompted for configuration information, and the DHCP client is not configured.

If the system does not use preconfiguration, you are prompted for system configuration information by `sysidtool` programs when the system reboots. See the `sysidtool`(1M) man page for more information.

5 **When prompted to use DHCP to configure network interfaces, specify No.**

# DHCP Client Administration

The Solaris DHCP client software does not require administration under normal system operation. The `dhcpagent` daemon automatically starts when the system boots, renegotiates leases, and stops when the system shuts down. You should not manually start and stop the `dhcpagent` daemon directly. Instead, as superuser on the client system, you can use the `ifconfig` command to affect `dhcpagent`'s management of the network interface, if necessary.

## `ifconfig` Command Options Used With the DHCP Client

This section summarizes the command options, which are documented in the `ifconfig`(1M) man page. The only difference between the DHCPv4 and the DHCPv6 versions of these commands is the “inet6” keyword. Include the “inet6” keyword for DHCPv6, but leave it out when running DHCPv4.

The `ifconfig` command enables you to do the following:

- **Start the DHCP client** – The command `ifconfig` *interface* `[inet6] dhcp start` initiates the interaction between `dhcpagent` and the DHCP server to obtain an IP address and a new set of configuration options. This command is useful when you change information that you want a client to use immediately, such as when you add IP addresses or change the subnet mask.
- **Request network configuration information only** – The command `ifconfig` *interface* `[inet6] dhcp inform` causes `dhcpagent` to issue a request for network configuration parameters, with the exception of the IP address. This command is useful when the network interface has a static IP address, but the client system needs updated network options. For example, this command is useful if you do not use DHCP to manage IP addresses, but you do use it to configure hosts on the network.
- **Request a lease extension** – The command `ifconfig` *interface* `[inet6] dhcp extend` causes `dhcpagent` to issue a request to renew the lease. The client does automatically request to renew leases. However, you might want to use this command if you change the lease time and want clients to use the new lease time immediately, rather than waiting for the next attempt at lease renewal.
- **Release the IP address** – The command `ifconfig` *interface* `[inet6] dhcp release` causes `dhcpagent` to relinquish the IP address used by the network interface. Release of the IP address happens automatically when the lease expires. You might want to issue this command with a laptop, for example, when leaving a network and planning to start the system on a new network. See also the `/etc/default/dhcpagent` configuration file `RELEASE_ON_SIGTERM` property.
- **Drop the IP address** – The command `ifconfig` *interface* `[inet6] dhcp drop` causes `dhcpagent` to take down the network interface without informing the DHCP server and cache the lease in the file system. This command enables the client to use the same IP address when it reboots.
- **Ping the network interface** – The command `ifconfig` *interface* `[inet6] dhcp ping` lets you determine if the interface is under the control of DHCP.
- **View the DHCP configuration status of the network interface** – The command `ifconfig` *interface* `[inet6] dhcp status` displays the current state of the DHCP client. The display indicates the following items:
  - If an IP address has been bound to the client
  - The number of requests sent, received, and declined
  - If this interface is the primary interface
  - Times when the lease was obtained, when it expires, and when renewal attempts are scheduled to begin

  For example:

```
# ifconfig hme0 dhcp status
Interface  State         Sent  Recv  Declined  Flags
hme0       BOUND            1     1         0   [PRIMARY]
(Began,Expires,Renew)=(08/16/2005 15:27, 08/18/2005 13:31, 08/17/2005 15:24)

# ifconfig hme0 inet6 dhcp status
Interface  State         Sent  Recv  Declined  Flags
hme0       BOUND            1     0         0   [PRIMARY]
(Began,Expires,Renew)=(11/22/2006 20:39, 11/22/2006 20:41, 11/22/2006 20:40)
```

# Setting DHCP Client Configuration Parameters

The `/etc/default/dhcpagent` file on the client system contains tunable parameters for the `dhcpagent`. You can use a text editor to change several parameters that affect client operation. The `/etc/default/dhcpagent` file is well documented, so for more information, you should refer to the file as well as to the `dhcpagent`(1M) man page.

The `/etc/dhcp.`*interface* file is another location in which parameters affecting the DHCP client are set. Parameters set in this file are used by system startup scripts with the `ifconfig` command. This, however, affects only DHCPv4. There is no DHCPv6 equivalent.

By default, the DHCP client is configured as follows:

## For DHCPv4

- The client system does not require a particular host name.

  If you want a client to request a specific host name, see "DHCPv4 Client Host Names" on page 430.
- Default requests for the client are given in `/etc/default/dhcpagent`, and includes DNS Server, DNS domain, and broadcast address.

  The DHCP client's parameter file can be set up to request more options in the `PARAM_REQUEST_LIST` keyword in the `/etc/default/dhcpagent` file. The DHCP server can be configured to provide options that were not specifically requested. See "About DHCP Macros" on page 311 and "Working With DHCP Macros (Task Map)" on page 385 for information about using DHCP server macros to send information to clients.

## For DHCPv4 and DHCPv6

- The client system uses DHCP on one physical network interface.

  If you want to use DHCP on more than one physical network interface, see "DHCP Client Systems With Multiple Network Interfaces" on page 429.
- The client is not automatically configured as a name service client if the DHCP client was configured after the Solaris installation.

See "DHCP Client Systems and Name Services" on page 431 for information about using name services with DHCP clients.

# DHCP Client Systems With Multiple Network Interfaces

The DHCP client can simultaneously manage several different interfaces on one system. The interfaces can be physical interfaces or logical interfaces. Each interface has its own IP address and lease time. If more than one network interface is configured for DHCP, the client issues separate requests to configure them. The client maintains a separate set of network configuration parameters for each interface. Although the parameters are stored separately, some of the parameters are global in nature. The global parameters apply to the system as a whole, rather than to a particular network interface.

The host name, NIS domain name, and time zone are examples of global parameters. Global parameters usually have different values for each interface. However, only one value can be used for each global parameter associated with each system. To be sure that there is only one answer to a query for a global parameter, only the parameters for the primary network interface are used. You can insert the word `primary` in the `/etc/dhcp.`*interface* file for the interface that you want to be treated as the primary interface. If the `primary` keyword is not used, the first interface in alphabetical order is considered to be the primary interface.

The DHCP client manages leases for logical interfaces and physical interfaces identically, except for the following limitation on logical interfaces:

- The DHCP client does not manage the default routes that are associated with logical interfaces.

  The Solaris kernel associates routes with physical interfaces, not logical interfaces. When a physical interface's IP address is established, the necessary default routes should be placed in the routing table. If DHCP is used subsequently to configure a logical interface associated with that physical interface, the necessary routes should already be in place. The logical interface uses the same routes.

  When a lease expires on a physical interface, the DHCP client removes the default routes that are associated with the interface. When a lease expires on a logical interface, the DHCP client does not remove the default routes associated with the logical interface. The associated physical interface and possibly other logical interfaces might need to use the same routes.

  If you need to add or remove default routes that are associated with a DHCP-controlled interface, you can use the DHCP client event script mechanism. See "DHCP Client Event Scripts" on page 436.

# DHCPv4 Client Host Names

By default, the Solaris DHCPv4 client does not supply its own host name, because the client expects the DHCP server to supply the host name. The Solaris DHCPv4 server is configured to supply host names to DHCPv4 clients by default. When you use the Solaris DHCPv4 client and server together, these defaults work well. However, when you use the Solaris DHCPv4 client with some third-party DHCP servers, the client might not receive a host name from the server. If the Solaris DHCP client does not receive a host name through DHCP, the client system looks at the `/etc/nodename` file for a name to use as the host name. If the file is empty, the host name is set to `unknown`.

If the DHCP server supplies a name in the DHCP `Hostname` option, the client uses that host name, even if a different value is placed in the `/etc/nodename` file. If you want the client to use a specific host name, you can enable the client to request that name. See the following procedure.

---

**Note –** The following procedure does not work with all DHCP servers. Through this procedure you are requiring the client to send a specific host name to the DHCP server, and to expect the same name in return.

However, the DHCP server does not have to respect this request and many do not. They simply return a different name.

---

## ▼ How to Enable a Solaris DHCPv4 Client to Request a Specific Host Name

1. **On the client system, edit the `/etc/default/dhcpagent` file as superuser.**

2. **Find the `REQUEST_HOSTNAME` keyword in the `/etc/default/dhcpagent` file and modify the keyword as follows:**

   ```
   REQUEST_HOSTNAME=yes
   ```

   If a comment sign (`#`) is in front of `REQUEST_HOSTNAME`, remove the `#`. If the `REQUEST_HOSTNAME` keyword is not present, insert the keyword.

3. **Edit the `/etc/hostname.`*interface* file on the client system to add the following line:**

   `inet` *hostname*

   *hostname* is the name that you want the client to use.

4. **Type the following commands to have the client perform a full DHCP negotiation upon rebooting:**

   ```
   # ifconfig interface dhcp release
   # reboot
   ```

The DHCP data that is cached on the client is removed. The client restarts the protocol to request new configuration information, including a new host name. The DHCP server first makes sure that the host name is not in use by another system on the network. The server then assigns the host name to the client. If configured to do so, the DHCP server can update name services with the client's host name.

If you want to change the host name later, repeat Step 3 and Step 4.

# DHCP Client Systems and Name Services

Solaris systems support the following name services: DNS, NIS, NIS+, and a local file store (`/etc/inet/hosts`). Each name service requires some configuration before it is usable. The name service switch configuration file (see `nsswitch.conf`(4)) must also be set up appropriately to indicate the name services to be used.

Before a DHCP client system can use a name service, you must configure the system as a client of the name service. By default, and unless configured otherwise during system installation, only local files are used.

The following table summarizes issues that are related to each name service and DHCP. The table includes links to documentation that can help you set up clients for each name service.

**TABLE 16–1** Name Service Client Setup Information for DHCP Client Systems

| Name Service | Client Setup Information |
|---|---|
| NIS | If you are using Solaris DHCP to send Solaris network install information to a client system, you can use a configuration macro that contains the `NISservs` and `NISdmain` options. These options pass the IP addresses of NIS servers and the NIS domain name to the client. The client then automatically becomes an NIS client.<br><br>If a DHCP client system is already running the Solaris OS, the NIS client is not automatically configured on that system when the DHCP server sends NIS information to the client.<br><br>If the DHCP server is configured to send NIS information to the DHCP client system, you can see the values given to the client if you use the `dhcpinfo` command on the client as follows:<br><br>`# /sbin/dhcpinfo NISdmain`<br><br>`# /sbin/dhcpinfo NISservs`<br><br>**Note –** For DHCPv6, include `-v6`, and different protocol keywords in the command.<br><br>`# /sbin/dhcpinfo -v6 NISDomain`<br><br>`# /sbin/dhcpinfo -v6 NISServers`<br><br>Use the values returned for the NIS domain name and NIS servers when you set up the system as an NIS client.<br><br>You set up an NIS client for a Solaris DHCP client system in the standard way, as documented in Chapter 5, "Setting Up and Configuring NIS Service," in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.<br><br>**Tip –** You can write a script that uses `dhcpinfo` and `ypinit` to automate NIS client configuration on DHCP client systems. |
| NIS+ | If the NIS+ client for a DHCP client system is set up in the conventional way, then the DHCP server might give the client different addresses from time to time. This creates security issues, because NIS+ security includes IP address as part of the configuration. To assure that your client has the same address every time, set up the NIS+ client for a DHCP client system in a nonstandard way, which is documented in "Setting Up DHCP Clients as NIS+ Clients" on page 433.<br><br>If the DHCP client system has been manually assigned an IP address, the client's address is always the same. You can set up the NIS+ client in the standard way, which is documented in "Setting Up NIS+ Client Machines" in *System Administration Guide: Naming and Directory Services (NIS+)*. |

**TABLE 16–1** Name Service Client Setup Information for DHCP Client Systems *(Continued)*

| Name Service | Client Setup Information |
|---|---|
| `/etc/inet/hosts` | You must set up the `/etc/inet/hosts` file for a DHCP client system that is to use `/etc/inet/hosts` for its name service.<br><br>The DHCP client system's host name is added to its own `/etc/inet/hosts` file by the DHCP tools. However, you must manually add the host name to the `/etc/inet/hosts` files of other systems in the network. If the DHCP server system uses `/etc/inet/hosts` for name resolution, you must also manually add the client's host name on the system. |
| DNS | If the DHCP client system receives the DNS domain name through DHCP, the client system's `/etc/resolv.conf` file is configured automatically. The `/etc/nsswitch.conf` file is also automatically updated to append `dns` to the `hosts` line after any other name services in the search order. See *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* for more information about DNS. |

# Setting Up DHCP Clients as NIS+ Clients

You can use the NIS+ name service on Solaris systems that are DHCP clients. However, if your DHCP server can provide different addresses at different times, this partially circumvents one of the security-enhancing features of NIS+, the creation of Data Encryption Standard (DES) credentials. For the sake of security, configure the DHCP server to provide the same address all the time. When you set up an NIS+ client that is *not* using DHCP, you add unique DES credentials for the client to the NIS+ server. There are several ways to create credentials, such as using the `nisclient` script or the `nisaddcred` command.

NIS+ credential generation requires a client to have a static host name to create and store the credentials. If you want to use NIS+ and DHCP, you must create identical credentials to be used for all the host names of DHCP clients. In this way, no matter what IP address and associated host name that a DHCP client receives, the client can use the same DES credentials.

The following procedure shows you how to create identical credentials for all DHCP host names. This procedure is valid only if you know the host names that DHCP clients use. For example, when the DHCP server generates the host names, you know the possible host names that a client can receive.

## ▼ How to Set Up Solaris DHCP Clients as NIS+ Clients

A DHCP client system that is to be an NIS+ client must use credentials that belong to another NIS+ client system in the NIS+ domain. This procedure only produces credentials for the system, which apply only to the superuser logged in to the system. Other users who log in to the DHCP client system must have their own unique credentials in the NIS+ server. These credentials are created according to a procedure in the *System Administration Guide: Naming and Directory Services (NIS+)*.

1. **Create the credentials for a client by typing the following command on the NIS+ server:**

   ```
   # nisgrep nisplus-client-name cred.org_dir > /tmp/file
   ```

   This command writes the `cred.org_dir` table entry for the NIS+ client to a temporary file.

2. **Use the `cat` command to view the contents of the temporary file.**

   Or, use a text editor.

3. **Copy the credentials to use for DHCP clients.**

   You must copy the public key and private key, which are long strings of numbers and letters separated by colons. The credentials are to be pasted into the command issued in the next step.

4. **Add credentials for a DHCP client by typing the following command:**

   ```
   # nistbladm -a cname=" dhcp-client-name@nisplus-domain" auth_type=DES \
   auth_name="unix.dhcp-client-name@nisplus-domain" \
   public_data=copied-public-key \
   private_data=copied-private-key
   ```

   For the *copied-public-key*, paste the public key information that you copied from the temporary file. For the *copied-private-key*, paste the private key information that you copied from the temporary file.

5. **Remote copy files from the NIS+ client system to the DHCP client system by typing the following commands on the DHCP client system:**

   ```
   # rcp nisplus-client-name:/var/nis/NIS_COLD_START /var/nis
   # rcp nisplus-client-name:/etc/.rootkey /etc
   # rcp nisplus-client-name:/etc/defaultdomain /etc
   ```

   If you get a "permission denied" message, the systems might not be set up to allow remote copying. In this case, you can copy the files as a regular user to an intermediate location. As superuser, copy the files from the intermediate location to the proper location on the DHCP client system.

6. **Copy the correct name service switch file for NIS+ by typing the following command on the DHCP client system:**

   ```
   # cp /etc/nsswitch.nisplus /etc/nsswitch.conf
   ```

7. **Reboot the DHCP client system.**

   The DHCP client system should now be able to use NIS+ services.

**Example 16–1** Setting up a Solaris DHCP Client System as an NIS+ Client

The following example assumes that you have one system `nisei`, which is an NIS+ client in the NIS+ domain `dev.example.net`. You also have one DHCP client system, `dhow`, and you want `dhow` to be an NIS+ client.

```
     (First log in as superuser on the NIS+ server)
# nisgrep nisei cred.org_dir > /tmp/nisei-cred
# cat /tmp/nisei-cred
nisei.dev.example.net.:DES:unix.nisei@dev.example.net:46199279911a84045b8e0
c76822179138173a20edbd8eab4:90f2e2bb6ffe7e3547346dda624ec4c7f0fe1d5f37e21cff63830
c05bc1c724b
# nistbladm -a cname="dhow@dev.example.net." \
auth_type=DES auth_name="unix.dhow@dev.example.net" \
public_data=46199279911a84045b8e0c76822179138173a20edbd8eab4 \
private_data=90f2e2bb6ffe7e3547346dda624ec4c7f0fe1d5f37e21cff63830\
c05bc1c724b
# rlogin dhow
     (Log in as superuser on dhow)
# rcp nisei:/var/nis/NIS_COLD_START /var/nis
# rcp nisei:/etc/.rootkey /etc
# rcp nisei:/etc/defaultdomain /etc
# cp /etc/nsswitch.nisplus /etc/nsswitch.conf
# reboot
```

The DHCP client system `dhow` should now be able to use NIS+ services.

### Example 16–2 Adding Credentials With a Script

If you want to set up a large number of DHCP client systems as NIS+ clients, you can write a script. A script can quickly add the entries to the `cred.org_dir` NIS+ table. The following example shows a sample script.

```
#! /usr/bin/ksh
#
# Copyright (c) by Sun Microsystems, Inc. All rights reserved.
#
# Sample script for cloning a credential. Hosts file is already populated
# with entries of the form dhcp-[0-9][0-9][0-9]. The entry we're cloning
# is dhcp-001.
#
#
PUBLIC_DATA=6e72878d8dc095a8b5aea951733d6ea91b4ec59e136bd3b3
PRIVATE_DATA=3a86729b685e2b2320cd7e26d4f1519ee070a60620a93e48a8682c5031058df4
HOST="dhcp-"
DOMAIN="mydomain.example.com"

for
i in 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019
do
     print - ${HOST}${i}
     #nistbladm -r [cname="${HOST}${i}.${DOMAIN}."]cred.org_dir
     nistbladm -a cname="${HOST}${i}.${DOMAIN}." \
         auth_type=DES auth_name="unix.${HOST}${i}@${DOMAIN}" \
```

```
        public_data=${PUBLIC_DATA} private_data=${PRIVATE_DTA} cred.org_Dir
done

exit 0
```

# DHCP Client Event Scripts

You can set up the Solaris DHCP client to run an executable program or script that can perform any action that is appropriate for the client system. The program or script, which is called an *event script*, is automatically executed after certain DHCP lease events occur. The event script can be used to run other commands, programs, or scripts in response to specific lease events. You must provide your own event script to use this feature.

The following event keywords are used by `dhcpagent` to signify DHCP lease events:

| Event Keyword | Description |
|---|---|
| `BOUND` and `BOUND6` | The interface is configured for DHCP. The client receives the acknowledgement message (DHCPv4 `ACK`) or (DHCPv6 `Reply`) from the DHCP server, which grants the lease request for an IP address. The event script is invoked immediately after the interface is configured successfully. |
| `EXTEND` and `EXTEND6` | The client successfully extends a lease. The event script is invoked immediately after the client receives the acknowledgement message from the DHCP server for the renew request. |
| `EXPIRE` and `EXPIRE6` | The lease expires when the lease time is up. For DHCPv4, the event script is invoked immediately before the leased address is removed from the interface and the interface is marked as down. For DHCPv6, the event script is invoked just before the last remaining leased addresses are removed from the interface. |
| `DROP` and `DROP6` | The client drops the lease to remove the interface from DHCP control. The event script is invoked immediately before the interface is removed from DHCP control. |
| `RELEASE` and `RELEASE6` | The client relinquishes the IP address. The event script is invoked immediately before the client releases the address on the interface and sends the DHCPv4 `RELEASE` or DHCPv6 `Release` packet to the DHCP server. |
| `INFORM` and `INFORM6` | An interface acquires new or updated configuration information from a DHCP server through the DHCPv4 `INFORM` or the DHCPv6 `Information-Request` message. These events occur when the DHCP client obtains only configuration parameters from the server and does not obtain an IP address lease. |

| | |
|---|---|
| LOSS6 | During lease expiration, when one or more valid leases still remain, the event script is invoked just before expired addresses are removed. Those being removed are marked with the IFF_DEPRECATED flag. |

With each of these events, dhcpagent invokes the following command:

```
/etc/dhcp/eventhook interface event
```

where *interface* is the interface that is using DHCP and *event* is one of the event keywords described previously. For example, when the ce0 interface is first configured for DHCP, the dhcpagent invokes the event script as follows:

```
/etc/dhcp/eventhook ce0 BOUND
```

To use the event script feature, you must do the following:

- Name the executable file /etc/dhcp/eventhook.
- Set the owner of the file to be root.
- Set permissions to 755 (rwxr-xr-x).
- Write the script or program to perform a sequence of actions in response to any of the documented events. Because Sun might add new events, the program must silently ignore any events that are not recognized or do not require action. For example, the program or script might write to a log file when the event is RELEASE, and ignore all other events.
- Make the script or program noninteractive. Before the event script is invoked, stdin, stdout, and stderr are connected to /dev/null. To see the output or errors, you must redirect to a file.

The event script inherits its program environment from dhcpagent, and runs with root privileges. The script can use the dhcpinfo utility to obtain more information about the interface, if necessary. See the dhcpinfo(1) man page for more information.

The dhcpagent daemon waits for the event script to exit on all events. If the event script does not exit after 55 seconds, dhcpagent sends a SIGTERM signal to the script process. If the process still does not exit after three additional seconds, the daemon sends a SIGKILL signal to kill the process.

The dhcpagent(1M) man page includes one example of an event script.

Example 16–3 shows how to use a DHCP event script to keep the content of the /etc/resolv.conf file up to date. When the BOUND and EXTEND events occur, the script replaces the names of the domain server and name server. When the EXPIRE, DROP and RELEASE events occur, the script removes the names of the domain server and name server from the file.

**Note –** The example script assumes that DHCP is the authoritative source for the names of the domain server and the name server. The script also assumes that all interfaces under DHCP control return consistent and current information. These assumptions might not reflect conditions on your system.

**EXAMPLE 16–3** Event Script for Updating the `/etc/resolv.conf` File

```
#!/bin/ksh -p

PATH=/bin:/sbin export PATH
umask 0222

# Refresh the domain and name servers on /etc/resolv.conf

insert ()
{
    dnsservers=‘dhcpinfo -i $1 DNSserv‘
    if [ -n "$dnsservers" ]; then
        # remove the old domain and name servers
        if [ -f /etc/resolv.conf ]; then
            rm -f /tmp/resolv.conf.$$
            sed -e ’/^domain/d’ -e ’/^nameserver/d’ \
                /etc/resolv.conf > /tmp/resolv.conf.$$
        fi

        # add the new domain
        dnsdomain=‘dhcpinfo -i $1 DNSdmain‘
        if [ -n "$dnsdomain" ]; then
            echo "domain $dnsdomain" >> /tmp/resolv.conf.$$
        fi

        # add new name servers
        for name in $dnsservers; do
            echo nameserver $name >> /tmp/resolv.conf.$$
        done
        mv -f /tmp/resolv.conf.$$ /etc/resolv.conf
    fi
}

# Remove the domain and name servers from /etc/resolv.conf

remove ()
{
    if [ -f /etc/resolv.conf ]; then
        rm -f /tmp/resolv.conf.$$
```

**EXAMPLE 16–3** Event Script for Updating the `/etc/resolv.conf` File *(Continued)*

```
        sed -e '/^domain/d' -e '/^nameserver/d' \
            /etc/resolv.conf > /tmp/resolv.conf.$$
        mv -f /tmp/resolv.conf.$$ /etc/resolv.conf
    fi
}

case $2 in
BOUND | EXTEND)
    insert $1
    exit 0
    ;;
EXPIRE | DROP | RELEASE)
    remove
    exit 0
    ;;
*)
    exit 0
    ;;
esac
```

◆ ◆ ◆ CHAPTER 17

# Troubleshooting DHCP (Reference)

This chapter provides information to help you solve problems that you might encounter when you configure a DHCP server or client. The chapter also helps you with problems you might have in using DHCP after configuration is complete.

The chapter includes the following information:


- "Troubleshooting DHCP Server Problems" on page 441
- "Troubleshooting DHCP Client Configuration Problems" on page 447


See Chapter 14, "Configuring the DHCP Service (Tasks)," for information about configuring your DHCP server. See"Enabling and Disabling a Solaris DHCP Client" on page 425 for information about configuring your DHCP client.

## Troubleshooting DHCP Server Problems

The problems that you might encounter when you configure the server fall into the following categories:


- "NIS+ Problems and the DHCP Data Store" on page 441
- "IP Address Allocation Errors in DHCP" on page 444


### NIS+ Problems and the DHCP Data Store

If you use NIS+ as the DHCP data store, problems that you might encounter can be categorized as follows:


- "Cannot Select NIS+ as the DHCP Data Store" on page 442
- "NIS+ Is Not Adequately Configured for DHCP Data Store" on page 442
- "NIS+ Access Problems for the DHCP Data Store" on page 443

## Cannot Select NIS+ as the DHCP Data Store

If you try to use NIS+ as your data store, DHCP Manager might not offer NIS+ as a choice for the data store. If you use the `dhcpconfig` command, you might see a message stating that NIS+ does not appear to be installed and running. Both these symptoms mean that NIS+ has not been configured for this server, although NIS+ might be in use on the network. Before you can select NIS+ as a data store, the server system must be configured as an NIS+ client.

Before you set up the DHCP server system as an NIS+ client, the following statements must be true:

- The domain must have already been configured.
- The NIS+ domain's master server must be running.
- The master server's tables must be populated.
- The hosts table must have an entry for the new client system, the DHCP server system.

"Setting Up NIS+ Client Machines" in *System Administration Guide: Naming and Directory Services (NIS+)* provides detailed information about configuring an NIS+ client.

## NIS+ Is Not Adequately Configured for DHCP Data Store

After you successfully use NIS+ with DHCP, you might encounter errors if changes are made to NIS+. The changes could introduce configuration problems. Use the following explanations of problems and solutions to help you determine the cause of configuration problems.

**Problem:** Root object does not exist in the NIS+ domain.

**Solution:** Type the following command:

```
/usr/lib/nis/nisstat
```

This command displays statistics for the domain. If the root object does not exist, no statistics are returned.

Set up the NIS+ domain using the *System Administration Guide: Naming and Directory Services (NIS+)*.

**Problem:** NIS+ is not used for `passwd` and `publickey` information.

**Solution:** Type the following command to view the configuration file for the name service switch:

```
cat /etc/nsswitch.conf
```

Check the `passwd` and `publickey` entries for the "nisplus" keyword. Refer to the *System Administration Guide: Naming and Directory Services (NIS+)* for information about configuring the name service switch.

**Problem:** The domain name is empty.

**Solution:** Type the following command:

```
domainname
```

If the command lists an empty string, no domain name has been set for the domain. Use local files for your data store, or set up an NIS+ domain for your network. Refer to the *System Administration Guide: Naming and Directory Services (NIS+)*.

**Problem:** The `NIS_COLD_START` file does not exist.

**Solution:** Type the following command on the server system to determine if the file exists:

```
cat /var/nis/NIS_COLD_START
```

Use local files for your data store, or create an NIS+ client. Refer to the *System Administration Guide: Naming and Directory Services (NIS+)*.

## NIS+ Access Problems for the DHCP Data Store

NIS+ access problems might cause error messages about incorrect DES credentials, or inadequate permissions to update NIS+ objects or tables. Use the following explanations of problems and solutions to determine the cause of NIS+ access errors you receive.

**Problem:** The DHCP server system does not have create access to the `org_dir` object in the NIS+ domain.

**Solution:** Type the following command:

```
nisls -ld org_dir
```

The access rights are listed in the form `r---rmcdrmcdr---`, where the permissions apply respectively to nobody, owner, group, and world. The owner of the object is listed next.

Normally, the `org_dir` directory object provides full rights to both the owner and the group. Full rights consist of read, modify, create, and destroy. The `org_dir` directory object provides only read access to the world and nobody classes.

The DHCP server name must either be listed as the owner of the `org_dir` object, or be listed as a principal in the group. The group must have create access. List the group with the command:

```
nisls -ldg org_dir
```

Use the `nischmod` command to change the permissions for `org_dir` if necessary. For example, to add create access to the group, you would type the following command:

```
nischmod g+c org_dir
```

See the `nischmod`(1) man page for more information.

**Problem:** The DHCP server does not have access rights to create a table under the `org_dir` object.

Usually, this problem means the server system's principal name is not a member of the owning group for the `org_dir` object, or no owning group exists.

**Solution:** Type this command to find the owning group name:

**niscat -o org_dir**

Look for a line that is similar to:

```
Group : "admin.example.com."
```

List the principal names in the group using the command:

**nisgrpadm -l** *groupname*

For example, this command lists the principal names of the group `admin.example.com`:

```
nisgrpadm -l admin.example.com
```

The server system's name should be listed as an explicit member of the group or included as an implicit member of the group. If necessary, add the server system's name to the group using the `nisgrpadm` command.

For example, to add the server name `pacific` to the group `admin.example.com`, you would type the following command:

```
nisgrpadm -a admin.example.com pacific.example.com
```

See the `nisgrpadm`(1) man page for more information.

**Problem:** The DHCP server does not have valid Data Encryption Standard (DES) credentials in the NIS+ `cred` table.

**Solution:** If there is a credential problem, an error message states that the user does not have DES credentials in the NIS+ name service.

Use the `nisaddcred` command to add security credentials for the DHCP server system.

The following example shows how to add DES credentials for the system `mercury` in the domain `example.com`:

```
nisaddcred -p unix.mercury@example.com \
-P mercury.example.com. DES example.com.
```

The command prompts for the root password, which is required to generate an encrypted secret key.

See the `nisaddcred`(1M) man page for more information.

# IP Address Allocation Errors in DHCP

When a client attempts to obtain or verify an IP address, you might see problems logged to `syslog` or in server debugging mode output. The following list of common error messages indicates the possible causes and solutions.

`There is no` *n.n.n.n* `dhcp-network table for DHCP client's network`

**Cause:** A client is requesting a specific IP address or seeking to extend a lease on its current IP address. The DHCP server cannot find the DHCP network table for that address.

**Solution:** The DHCP network table might have been deleted mistakenly. You can recreate the network table by adding the network again using DHCP Manager or the `dhcpconfig` command.

`ICMP ECHO reply to OFFER candidate:` *n.n.n.n*`, disabling`

**Cause:** The IP address considered for offering to a DHCP client is already in use. This problem might occur if more than one DHCP server owns the address. The problem might also occur if an address was manually configured for a non-DHCP network client.

**Solution:** Determine the proper ownership of the address. Correct either the DHCP server database or the host's network configuration.

`ICMP ECHO reply to OFFER candidate:` *n.n.n.n*`. No corresponding dhcp network record.`

**Cause:** The IP address considered for offering to a DHCP client does not have a record in a network table. This error indicates that the IP address record was deleted from the DHCP network table after the address was selected. This error can only happen in the brief period before the duplicate address check is completed.

**Solution:** Use DHCP Manager or the `pntadm` command to view the DHCP network table. If the IP address is missing, create the address with DHCP Manager by choosing Create from the Edit menu on the Address tab. You can also use `pntadm` to create the IP address.

`DHCP network record for` *n.n.n.n*`is unavailable, ignoring request.`

**Cause:** The record for the requested IP address is not in the DHCP network table, so the server is dropping the request.

**Solution:** Use DHCP Manager or the `pntadm` command to view the DHCP network table. If the IP address is missing, create the address with DHCP Manager by choosing Create from the Edit menu on the Address tab. You can also use `pntadm` to create the address.

*n.n.n.n* `currently marked as unusable.`

**Cause:** The requested IP address cannot be offered because the address has been marked in the network table as unusable.

**Solution:** You can use DHCP Manager or the `pntadm` command to make the address usable.

*n.n.n.n* `was manually allocated. No dynamic address will be allocated.`

**Cause:** The client ID has been assigned a manually allocated address, and that address is marked as unusable. The server cannot allocate a different address to this client.

**Solution:** You can use DHCP Manager or the `pntadm` command to make the address usable, or manually allocate a different address to the client.

`Manual allocation (`*n.n.n.n, client ID*`) has` *n* `other records. Should have 0.`

**Cause:** The client that has the specified client ID has been manually assigned more than one IP address. A client should be assigned only one address. The server selects the last manually assigned address that is found in the network table.

**Solution:** Use DHCP Manager or the `pntadm` command to modify IP addresses to remove the additional manual allocations.

`No more IP addresses on` *n.n.n.n*`network.`

**Cause:** All IP addresses currently managed by DHCP on the specified network have been allocated.

**Solution:** Use DHCP Manager or the `pntadm` command to create new IP addresses for this network.

`Client:` *clientid* `lease on` *n.n.n.n* `expired.`

**Cause:** The lease was not negotiable and timed out.

**Solution:** The client should automatically restart the protocol to obtain a new lease.

`Offer expired for client:` *n.n.n.n*

**Cause:** The server made an IP address offer to the client, but the client took too long to respond and the offer expired.

**Solution:** The client should automatically issue another discover message. If this message also times out, increase the cache offer time out for the DHCP server. In DHCP Manager, choose Modify from the Service menu.

`Client:` *clientid* `REQUEST is missing requested IP option.`

**Cause:** The client's request did not specify the offered IP address, so the DHCP server ignored the request. This problem might occur if you use a third-party DHCP client that is not compliant with the updated DHCP protocol, RFC 2131.

**Solution:** Update the client software.

`Client:` *clientid* `is trying to renew` *n.n.n.n*`, an IP address it has not leased.`

**Cause:** The IP address for this client in the DHCP network table does not match the IP address that the client specified in its renewal request. The DHCP server does not renew the lease. This problem might occur if you delete a client's record while the client is still using the IP address.

**Solution:** Use DHCP Manager or the `pntadm` command to examine the network table, and correct the client's record, if necessary. The client ID should be bound to the specified IP address. If the client ID is not bound, edit the address properties to add the client ID.

`Client:` *clientid* `is trying to verify unrecorded address:` *n.n.n.n*`, ignored.`
**Cause:** The specified client has not been registered in the DHCP network table with this address, so the request is ignored by this DHCP server.

Another DHCP server on the network might have assigned this client the address. However, you might also have deleted the client's record while the client was still using the IP address.

**Solution:** Use DHCP Manager or the `pntadm` command to examine the network table on this server and any other DHCP servers on the network. Make corrections, if necessary.

You can also do nothing and allow the lease to expire. The client automatically requests a new address lease.

If you want the client to get a new lease immediately, restart the DHCP protocol on the client by typing the following commands:

```
ifconfig interface dhcp release
ifconfig interface dhcp start
```

# Troubleshooting DHCP Client Configuration Problems

The problems that you might encounter with a DHCP client fall into the following categories:

- "Problems Communicating With the DHCP Server" on page 447
- "Problems With Inaccurate DHCP Configuration Information" on page 456

## Problems Communicating With the DHCP Server

This section describes problems that you might encounter as you add DHCP clients to the network.

After you enable the client software and reboot the system, the client tries to reach the DHCP server to obtain its network configuration. If the client fails to reach the server, you might see error messages such as the following:

```
DHCP or BOOTP server not responding
```

Before you can determine the problem, you must gather diagnostic information from both the client and the server. To gather information, you can perform the following tasks:

1. "How to Run the DHCP Client in Debugging Mode" on page 448
2. "How to Run the DHCP Server in Debugging Mode" on page 448
3. "How to Use `snoop` to Monitor DHCP Network Traffic" on page 449

You can do these things separately or concurrently.

The information that you gather can help you determine if the problem is with the client, server, or a relay agent. Then, you can find a solution.

## How to Run the DHCP Client in Debugging Mode

If the client is not a Solaris DHCP client, refer to the client's documentation for information about how to run the client in debugging mode.

If you have a Solaris DHCP client, use the following steps.

1. **Become superuser on the DHCP client system.**

2. **Kill the DHCP client daemon.**

   ```
   # pkill -x dhcpagent
   ```

3. **Restart the daemon in debugging mode.**

   ```
   # /sbin/dhcpagent -d1 -f &
   ```

   The `-d` switch puts the DHCP client in debugging mode with level 1 verbosity. The `-f` switch causes output to be sent to the console instead of to `syslog`.

4. **Configure the interface to start DHCP negotiation.**

   ```
   # ifconfig interface dhcp start
   ```

   Replace *interface* with the name of the network interface of the client, such as `ge0`.

   When run in debugging mode, the client daemon displays messages to your screen while performing DHCP requests. See "Output from DHCP Client in Debugging Mode" on page 449 for information about client debugging mode output.

## How to Run the DHCP Server in Debugging Mode

1. **Become superuser on the server system.**

2. **Stop the DHCP server temporarily.**

   ```
   # svcadm disable -t svc:/network/dhcp-server
   ```

   You can also use DHCP Manager or `dhcpconfig` to stop the server.

3. **Restart the daemon in debugging mode.**

   ```
   # /usr/lib/inet/in.dhcpd -d -v
   ```

   You should also use any `in.dhcpd` command-line options that you normally use when you run the daemon. For example, if you run the daemon as a BOOTP relay agent, include the `-r` option with the `in.dhcpd -d -v` command.

When run in debugging mode, the daemon displays messages to your screen while processing DHCP or BOOTP requests. See "Output from the DHCP Server in Debugging Mode" on page 450 for information about server debugging mode output.

## ▼ How to Use `snoop` to Monitor DHCP Network Traffic

1. **Become superuser on the DHCP server system.**

2. **Start `snoop` to begin tracing network traffic across the server's network interface.**

   ```
   # /usr/sbin/snoop -d interface -o snoop-output-filename udp port 67 or udp port 68
   ```

   For example, you might type the following command:

   ```
   #  /usr/sbin/snoop -d hme0 -o /tmp/snoop.output udp port 67 or udp port 68
   ```

   `snoop` continues to monitor the interface until you stop `snoop` by pressing Control-C after you have the information that you need.

3. **Boot the client system, or restart the `dhcpagent` on the client system.**

   "How to Run the DHCP Client in Debugging Mode" on page 448 describes how to restart `dhcpagent`.

4. **On the server system, use `snoop` to display the output file with the contents of network packets:**

   ```
   # /usr/sbin/snoop -i snoop-output-filename -x0 -v
   ```

   For example, you might type the following command:

   ```
   # /usr/sbin/snoop -i /tmp/snoop.output -x0 -v
   ```

**See Also** See "DHCP `snoop` Output" on page 454 for information about interpreting the output.

## Output from DHCP Client in Debugging Mode

The following example shows normal output when a DHCP client in debugging mode sends its DHCP request and receives its configuration information from a DHCP server.

**EXAMPLE 17–1** Normal Output from the DHCP Client in Debugging Mode

```
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x27fc8 (DHCP filter)
/sbin/dhcpagent: debug: init_ifs: initted interface hme0
/sbin/dhcpagent: debug: insert_ifs: hme0: sdumax 1500, optmax 1260, hwtype 1, hwlen 6
/sbin/dhcpagent: debug: insert_ifs: inserted interface hme0
/sbin/dhcpagent: debug: register_acknak: registered acknak id 5
/sbin/dhcpagent: debug: unregister_acknak: unregistered acknak id 5
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x26018 (ARP reply filter)
```

**EXAMPLE 17–1** Normal Output from the DHCP Client in Debugging Mode *(Continued)*

```
/sbin/dhcpagent: info: setting IP netmask on hme0 to 255.255.192.0
/sbin/dhcpagent: info: setting IP address on hme0 to 10.23.3.233
/sbin/dhcpagent: info: setting broadcast address on hme0 to 10.23.63.255
/sbin/dhcpagent: info: added default router 10.23.0.1 on hme0
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x28054 (blackhole filter)
/sbin/dhcpagent: debug: configure_if: bound ifsp->if_sock_ip_fd
/sbin/dhcpagent: info: hme0 acquired lease, expires Tue Aug 10 16:18:33 2006
/sbin/dhcpagent: info: hme0 begins renewal at Tue Aug 10 15:49:44 2006
/sbin/dhcpagent: info: hme0 begins rebinding at Tue Aug 10 16:11:03 2006
```

If the client cannot reach the DHCP server, you might see debugging mode output that is similar to the output shown in the following example.

**EXAMPLE 17–2** Output Indicating a Problem from the DHCP Client in Debugging Mode

```
/sbin/dhcpagent: debug: set_packet_filter: set filter 0x27fc8 (DHCP filter)
/sbin/dhcpagent: debug: init_ifs: initted interface hme0
/sbin/dhcpagent: debug: select_best: no valid OFFER/BOOTP reply
/sbin/dhcpagent: debug: select_best: no valid OFFER/BOOTP reply
/sbin/dhcpagent: debug: select_best: no valid OFFER/BOOTP reply
```

If you see this message, the client request never reached the server, or the server cannot send a response to the client. Run `snoop` on the server as described in "How to Use snoop to Monitor DHCP Network Traffic" on page 449 to determine if packets from the client have reached the server.

## Output from the DHCP Server in Debugging Mode

Normal server debugging mode output shows server configuration information followed by information about each network interface as the daemon starts. After daemon startup, the debugging mode output shows information about requests the daemon processes. Example 17–3 shows debugging mode output for a DHCP server that has just started. The server extends the lease for a client that is using an address owned by another DHCP server that is not responding.

**EXAMPLE 17–3** Normal Output for DHCP Server in Debugging Mode

```
Daemon Version: 3.1
Maximum relay hops: 4
Transaction logging to console enabled.
Run mode is: DHCP Server Mode.
Datastore: nisplus
Path: org_dir.dhcp.test..:dhcp.test..:$
```

**EXAMPLE 17–3** Normal Output for DHCP Server in Debugging Mode *(Continued)*

```
DHCP offer TTL: 10
Ethers compatibility enabled.
BOOTP compatibility enabled.
ICMP validation timeout: 1000 milliseconds, Attempts: 2.
Monitor (0005/hme0) started...
Thread Id: 0005 - Monitoring Interface: hme0 *****
MTU: 1500      Type: DLPI
Broadcast: 10.21.255.255
Netmask: 255.255.0.0
Address: 10.21.0.2
Monitor (0006/nf0) started...
Thread Id: 0006 - Monitoring Interface: nf0 *****
MTU: 4352      Type: DLPI
Broadcast: 10.22.255.255
Netmask: 255.255.0.0
Address: 10.22.0.1
Monitor (0007/qfe0) started...
Thread Id: 0007 - Monitoring Interface: qfe0 *****
MTU: 1500      Type: DLPI
Broadcast: 10.23.63.255
Netmask: 255.255.192.0
Address: 10.23.0.1
Read 33 entries from DHCP macro database on Tue Aug 10 15:10:27 2006
Datagram received on network device: qfe0
Client: 0800201DBA3A is requesting verification of address owned by 10.21.0.4
Datagram received on network device: qfe0
Client: 0800201DBA3A is requesting verification of address owned by 10.21.0.4
Datagram received on network device: qfe0
Client: 0800201DBA3A is requesting verification of address owned by 10.21.0.4
Datagram received on network device: qfe0
Client: 0800201DBA3A maps to IP: 10.23.3.233
Unicasting datagram to 10.23.3.233 address.
Adding ARP entry: 10.23.3.233 == 0800201DBA3A
DHCP EXTEND 0934312543 0934316143 10.23.3.233 10.21.0.2
          0800201DBA3A SUNW.Ultra-5_10 0800201DBA3A
```

Example 17–4 shows debugging mode output from a DHCP daemon that starts as a BOOTP relay agent. The agent relays requests from a client to a DHCP server, and relays the server's responses to the client.

**EXAMPLE 17–4** Normal Output from BOOTP Relay in Debugging Mode

```
Relay destination: 10.21.0.4 (blue-servr2)     network: 10.21.0.0
Daemon Version: 3.1
Maximum relay hops: 4
```

**EXAMPLE 17–4** Normal Output from BOOTP Relay in Debugging Mode *(Continued)*

```
Transaction logging to console enabled.
Run mode is: Relay Agent Mode.
Monitor (0005/hme0) started...
Thread Id: 0005 - Monitoring Interface: hme0 *****
MTU: 1500      Type: DLPI
Broadcast: 10.21.255.255
Netmask: 255.255.0.0
Address: 10.21.0.2
Monitor (0006/nf0) started...
Thread Id: 0006 - Monitoring Interface: nf0 *****
MTU: 4352      Type: DLPI
Broadcast: 10.22.255.255
Netmask: 255.255.0.0
Address: 10.22.0.1
Monitor (0007/qfe0) started...
Thread Id: 0007 - Monitoring Interface: qfe0 *****
MTU: 1500      Type: DLPI
Broadcast: 10.23.63.255
Netmask: 255.255.192.0
Address: 10.23.0.1
Relaying request 0800201DBA3A to 10.21.0.4, server port.
BOOTP RELAY-SRVR 0934297685 0000000000 0.0.0.0 10.21.0.4 0800201DBA3A
N/A 0800201DBA3A
Packet received from relay agent: 10.23.0.1
Relaying reply to client 0800201DBA3A
Unicasting datagram to 10.23.3.233 address.
Adding ARP entry: 10.23.3.233 == 0800201DBA3A
BOOTP RELAY-CLNT 0934297688 0000000000 10.23.0.1 10.23.3.233 0800201DBA3A
N/A 0800201DBA3A
Relaying request 0800201DBA3A to 10.21.0.4, server port.
BOOTP RELAY-SRVR 0934297689 0000000000 0.0.0.0 10.21.0.4 0800201DBA3A
N/A 0800201DBA3A
Packet received from relay agent: 10.23.0.1
Relaying reply to client 0800201DBA3A
Unicasting datagram to 10.23.3.233 address.
Adding ARP entry: 10.23.3.233 == 0800201DBA3A
```

If there is a problem with DHCP, the debugging mode output might display warnings or error messages. Use the following list of DHCP server error messages to find solutions.

`ICMP ECHO reply to OFFER candidate:` *ip_address* `disabling`

**Cause:** Before the DHCP server offers an IP address to a client, the server pings the address to verify that the address is not in use. If a client replies, the address is in use.

**Solution:** Make sure the addresses that you configured are not already in use. You can use the `ping` command. See the `ping`(1M) man page for more information.

`No more IP addresses on` *network-address* `network.`

**Cause:** No IP addresses are available in the DHCP network table associated with the client's network.

**Solution:** Create more IP addresses with DHCP Manager or the `pntadm` command. If the DHCP daemon is monitoring multiple subnets, be sure the additional addresses are for the subnet where the client is located. See "Adding IP Addresses to the DHCP Service" on page 374 for more information.

`No more IP addresses for` *network-address* `network when you are running the DHCP daemon in BOOTP compatibility mode.`

**Cause:** BOOTP does not use a lease time, so the DHCP server looks for free addresses with the BOOTP flag set to allocate to BOOTP clients.

**Solution:** Use DHCP Manager to allocate BOOTP addresses. See "Supporting BOOTP Clients With the DHCP Service (Task Map)" on page 367.

`Request to access nonexistent per network database:` *database-name* `in datastore:` *datastore.*

**Cause:** During configuration of the DHCP server, a DHCP network table for a subnet was not created.

**Solution:** Use DHCP Manager or the `pntadm` command to create the DHCP network table and new IP addresses. See "Adding DHCP Networks" on page 360.

`There is no` *table-name* `dhcp-network table for DHCP client's network.`

**Cause:** During configuration of the DHCP server, a DHCP network table for a subnet was not created.

**Solution:** Use DHCP Manager or the `pntadm` command to create the DHCP network table and new IP addresses. See "Adding DHCP Networks" on page 360.

`Client using non_RFC1048 BOOTP cookie.`

**Cause:** A device on the network is trying to access an unsupported implementation of BOOTP.

**Solution:** Ignore this message, unless you need to configure this device. If you want to support the device, see "Supporting BOOTP Clients With the DHCP Service (Task Map)" on page 367 for more information.

## DHCP snoop Output

In the `snoop` output, you should see that packets are exchanged between the DHCP client system and the DHCP server system. The IP address for each system is indicated in each packet. IP addresses for any routers or relay agents in the packet's path are also included. If the systems do not exchange packets, the client system might not be able to contact the server system at all. The problem is then at a lower level.

To evaluate `snoop` output, you must know what the expected behavior is. For example, you must know if the request should be going through a BOOTP relay agent. You must also know the MAC addresses and the IP address of the systems involved so that you can determine if those values are as expected. If there is more than one network interface, you must know the addresses of the network interfaces as well.

The following example shows normal `snoop` output for a DHCP acknowledgement message sent from the DHCP server on `blue-servr2` to a client whose MAC address is `8:0:20:8e:f3:7e`. In the message, the server assigns the client the IP address `192.168.252.6` and the host name `white-6`. The message also includes a number of standard network options and several vendor-specific options for the client.

**EXAMPLE 17–5** Sample `snoop` Output for One Packet

```
ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 26 arrived at 14:43:19.14
ETHER:  Packet size = 540 bytes
ETHER:  Destination = 8:0:20:8e:f3:7e, Sun
ETHER:  Source      = 8:0:20:1e:31:c1, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:   Total length = 526 bytes
IP:   Identification = 64667
IP:   Flags = 0x4 IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 254 seconds/hops
IP:   Protocol = 17 (UDP)
IP:   Header checksum = 157a
IP:   Source address = 10.21.0.4, blue-servr2
```

**EXAMPLE 17–5** Sample snoop Output for One Packet *(Continued)*

```
IP:   Destination address = 192.168.252.6, white-6
IP:   No options
IP:   UDP:  ----- UDP Header -----
UDP:
UDP:  Source port = 67
UDP:  Destination port = 68 (BOOTPC)
UDP:  Length = 506
UDP:  Checksum = 5D4C
UDP:
DHCP: ----- Dynamic Host Configuration Protocol -----
DHCP:
DHCP: Hardware address type (htype) =  1 (Ethernet (10Mb))
DHCP: Hardware address length (hlen) = 6 octets
DHCP: Relay agent hops = 0
DHCP: Transaction ID = 0x2e210f17
DHCP: Time since boot = 0 seconds
DHCP: Flags = 0x0000
DHCP: Client address (ciaddr) = 0.0.0.0
DHCP: Your client address (yiaddr) = 192.168.252.6
DHCP: Next server address (siaddr) = 10.21.0.2
DHCP: Relay agent address (giaddr) = 0.0.0.0
DHCP: Client hardware address (chaddr) = 08:00:20:11:E0:1B
DHCP:
DHCP: ----- (Options) field options -----
DHCP:
DHCP: Message type = DHCPACK
DHCP: DHCP Server Identifier = 10.21.0.4
DHCP: Subnet Mask = 255.255.255.0
DHCP: Router at = 192.168.252.1
DHCP: Broadcast Address = 192.168.252.255
DHCP: NISPLUS Domainname = dhcp.test
DHCP: IP Address Lease Time = 3600 seconds
DHCP: UTC Time Offset = -14400 seconds
DHCP: RFC868 Time Servers at = 10.21.0.4
DHCP: DNS Domain Name = sem.example.com
DHCP: DNS Servers at = 10.21.0.1
DHCP: Client Hostname = white-6
DHCP: Vendor-specific Options (166 total octets):
DHCP:   (02) 04 octets  0x8194AE1B (unprintable)
DHCP:   (03) 08 octets  "pacific"
DHCP:   (10) 04 octets  0x8194AE1B (unprintable)
DHCP:   (11) 08 octets  "pacific"
DHCP:   (15) 05 octets  "xterm"
DHCP:   (04) 53 octets  "/export/s2/base.s2s/latest/Solaris_8/Tools/Boot"
DHCP:   (12) 32 octets  "/export/s2/base.s2s/latest"
DHCP:   (07) 27 octets  "/platform/sun4u/kernel/unix"
```

**EXAMPLE 17–5** Sample snoop Output for One Packet *(Continued)*

```
DHCP:   (08) 07 octets  "EST5EDT"
  0: 0800 208e f37e 0800 201e 31c1 0800 4500    .. .ó~.. .1...E.
 16: 020e fc9b 4000 fe11 157a ac15 0004 c0a8    ....@....z......
 32: fc06 0043 0044 01fa 5d4c 0201 0600 2e21    ...C.D..]L.....!
 48: 0f17 0000 0000 0000 0000 c0a8 fc06 ac15    ................
 64: 0002 0000 0000 0800 2011 e01b 0000 0000    ........ .......
 80: 0000 0000 0000 0000 0000 0000 0000 0000    ................
 96: 0000 0000 0000 0000 0000 0000 0000 0000    ................
112: 0000 0000 0000 0000 0000 0000 0000 0000    ................
128: 0000 0000 0000 0000 0000 0000 0000 0000    ................
144: 0000 0000 0000 0000 0000 0000 0000 0000    ................
160: 0000 0000 0000 0000 0000 0000 0000 0000    ................
176: 0000 0000 0000 0000 0000 0000 0000 0000    ................
192: 0000 0000 0000 0000 0000 0000 0000 0000    ................
208: 0000 0000 0000 0000 0000 0000 0000 0000    ................
224: 0000 0000 0000 0000 0000 0000 0000 0000    ................
240: 0000 0000 0000 0000 0000 0000 0000 0000    ................
256: 0000 0000 0000 0000 0000 0000 0000 0000    ................
272: 0000 0000 0000 6382 5363 3501 0536 04ac    ......c.Sc5..6..
288: 1500 0401 04ff ffff 0003 04c0 a8fc 011c    ................
304: 04c0 a8fc ff40 0964 6863 702e 7465 7374    .....@.dhcp.test
320: 3304 0000 0e10 0204 ffff c7c0 0404 ac15    3...............
336: 0004 0f10 736e 742e 6561 7374 2e73 756e    ....sem.example.
352: 2e63 6f6d 0604 ac15 0001 0c07 7768 6974    com.........whit
368: 652d 362b a602 0481 94ae 1b03 0861 746c    e-6+.........pac
384: 616e 7469 630a 0481 94ae 1b0b 0861 746c    ific.........pac
400: 616e 7469 630f 0578 7465 726d 0435 2f65    ific...xterm.5/e
416: 7870 6f72 742f 7332 382f 6261 7365 2e73    xport/sx2/bcvf.s
432: 3238 735f 776f 732f 6c61 7465 7374 2f53    2xs_btf/latest/S
448: 6f6c 6172 6973 5f38 2f54 6f6f 6c73 2f42    olaris_x/Tools/B
464: 6f6f 740c 202f 6578 706f 7274 2f73 3238    oot. /export/s2x
480: 2f62 6173 652e 7332 3873 5f77 6f73 2f6c    /bcvf.s2xs_btf/l
496: 6174 6573 7407 1b2f 706c 6174 666f 726d    atest../platform
512: 2f73 756e 346d 2f6b 6572 6e65 6c2f 756e    /sun4u/kernel/un
528: 6978 0807 4553 5435 4544 54ff              ix..EST5EDT.
```

# Problems With Inaccurate DHCP Configuration Information

If a DHCP client receives inaccurate information in its network configuration information, look at the DHCP server data. You must examine the option values in the macros that the DHCP server processes for this client. Examples of inaccurate information might be the wrong NIS domain name or router IP address.

Use the following general guidelines to help you determine the source of the inaccurate information:

- Look at the macros defined on the server as described in "How to View Macros Defined on a DHCP Server (DHCP Manager)" on page 387. Review the information in "Order of Macro Processing" on page 312, and determine which macros are processed automatically for this client.
- Look at the network table to determine what macro (if any) is assigned to the client's IP address as the configuration macro. See "Working With IP Addresses in the DHCP Service (Task Map)" on page 370 for more information.
- Take note of any options that occur in more than one macro. Make sure the value that you want for an option is set in the last processed macro.
- Edit the appropriate macro or macros to assure that the correct value is passed to the client. See "Modifying DHCP Macros" on page 388.

# Problems With the DHCP Client-Supplied Host Name

This section describes problems that you might experience with DHCP clients that supply their own host names to be registered with DNS.

## DHCP Client Does Not Request a Host Name

If your client is not a Solaris DHCP client, consult the client's documentation to determine how to configure the client to request a host name. For Solaris DHCP clients, see "How to Enable a Solaris DHCPv4 Client to Request a Specific Host Name" on page 430.

## DHCP Client Does Not Get Requested Host Name

The following list includes describes possible problems a client might have in getting its requested hostname, and suggested solutions.

**Problem:** Client accepted an offer from a DHCP server that does not issue DNS updates.

**Solution:** If two DHCP servers are available to the client, the servers should both be configured to provide DNS updates. See "Enabling Dynamic DNS Updates by a DHCP Server" on page 352 for information about configuring the DHCP server and the DNS server.

To determine whether the DHCP server is configured to provide DNS updates:

1. Determine the IP address of the client's DHCP server. On the client system, use `snoop` or another application for capturing network packets. See "How to Use snoop to Monitor DHCP Network Traffic" on page 449, and perform the procedure on the client instead of the server. In the `snoop` output, look for the DHCP Server Identifier to get the IP address of the server.
2. Log in to the DHCP server system to verify that the system is configured to make DNS updates. Type the following command as superuser:

```
dhcpconfig -P
```

If `UPDATE_TIMEOUT` is listed as a server parameter, the DHCP server is configured to make DNS updates.

3. On the DNS server, look at the `/etc/named.conf` file. Find the `allow-update` keyword in the `zone` section of the appropriate domain. If the server allows DNS updates by the DHCP server, the DHCP server's IP address is listed in the `allow-update` keyword.

**Problem:** Client is using `FQDN` option to specify host name. Solaris DHCP does not currently support the `FQDN` option because the option is not officially in the DHCP protocol.

**Solution:** On the server, use `snoop` or another application for capturing network packets. See "How to Use snoop to Monitor DHCP Network Traffic" on page 449. In the `snoop` output, look for the `FQDN` option in a packet from the client.

Configure the client to specify host name using `Hostname` option. `Hostname` is option code 12. Refer to client documentation for instructions.

For a Solaris client, see "How to Enable a Solaris DHCPv4 Client to Request a Specific Host Name" on page 430

**Problem:** DHCP server that makes an address offer to the client does not know the client's DNS domain.

**Solution:** On the DHCP server look for the `DNSdmain` option with a valid value. Set the `DNSdmain` option to the correct DNS domain name in a macro that is processed for this client. `DNSdmain` is usually contained in the network macro. See "Modifying DHCP Macros" on page 388 for information about changing values of options in a macro.

**Problem:** The host name requested by client corresponds to an IP address that is not managed by the DHCP server. The Solaris DHCP server does not perform DNS updates for IP addresses that the server does not manage.

**Solution:** Check `syslog` for one of the following messages from the DHCP server:

- `There is no` *n.n.n.n* `dhcp-network table for DHCP client's network.`
- `DHCP network record for` *n.n.n.n* `is unavailable, ignoring request.`

Configure the client to request a different name. See "How to Enable a Solaris DHCPv4 Client to Request a Specific Host Name" on page 430. Choose a name that is mapped to an address managed by the DHCP server. You can see address mappings in DHCP Manager's Addresses tab. Alternatively, choose an address that is not mapped to any IP address.

**Problem:** The host name requested by client corresponds to an IP address that is currently not available for use. The address might be in use, leased to another client, or under offer to another client.

**Solution:** Check `syslog` for the following message from the DHCP server: `ICMP ECHO reply to OFFER candidate:` *n.n.n.n*.

Configure the client to choose a name corresponding to a different IP address. Alternatively, reclaim the address from the client that uses the address.

**Problem:** DNS server is not configured to accept updates from the DHCP server.

**Solution:** Examine the `/etc/named.conf` file on the DNS server. Look for the DHCP server's IP address with the `allow-update` keyword in the appropriate `zone` section for the DHCP server's domain. If the IP address is not present, the DNS server is not configured to accept updates from the DHCP server.

See "How to Enable Dynamic DNS Updating for DHCP Clients" on page 353 for information about configuring the DNS server.

If the DHCP server has multiple interfaces, you might need to configure the DNS server to accept updates from all of the DHCP server's addresses. Enable debugging on the DNS server to see whether the updates are reaching the DNS server. If the DNS server received update requests, examine the debugging mode output to determine why the updates did not occur. See the `in.named`.1M man page for information about DNS debugging mode.

**Problem:** DNS updates might not have completed in the allotted time. DHCP servers do not return host names to clients if the DNS updates have not completed by the configured time limit. However, attempts to complete the DNS updates continue.

**Solution:** Use the `nslookup` command to determine whether the updates completed successfully. See the `nslookup`(1M) man page.

For example, suppose the DNS domain is `hills.example.org`, and the DNS server's IP address is `10.76.178.11`. The host name that the client wants to register is `cathedral`. You could use the following command to determine if `cathedral` has been registered with that DNS server:

```
nslookup cathedral.hills.example.org 10.76.178.11
```

If the updates completed successfully, but not in the allotted time, you need to increase the time out value. See "How to Enable Dynamic DNS Updating for DHCP Clients" on page 353. In this procedure, you should increase the number of seconds to wait for a response from the DNS server before timing out.

◆ ◆ ◆ CHAPTER 18

# DHCP Commands and Files (Reference)

This chapter explains the relationships between the DHCP commands and the DHCP files. However, the chapter does not explain how to use the commands.

The chapter contains the following information:


- "DHCP Commands" on page 461
- "Files Used by the DHCP Service" on page 469
- "DHCP Option Information" on page 471


## DHCP Commands

The following table lists the commands that you can use to manage DHCP on your network.

**TABLE 18–1** Commands Used in DHCP

| Command | Description | Man Page |
|---|---|---|
| `dhtadm` | Used to make changes to the options and macros in the `dhcptab`. This command is most useful in scripts that you create to automate changes to your DHCP information. Use `dhtadm` with the `-P` option, and pipe the output through the `grep` command for a quick way to search for particular option values in the `dhcptab` table. | `dhtadm`(1M) |
| `pntadm` | Used to make changes to the DHCP network tables that map client IDs to IP addresses and optionally associate configuration information with IP addresses. | `pntadm`(1M) |
| `dhcpconfig` | Used to configure and unconfigure DHCP servers and BOOTP relay agents. Also used to convert to a different data store format, and to import and export DHCP configuration data. | `dhcpconfig`(1M) |

**TABLE 18–1** Commands Used in DHCP *(Continued)*

| Command | Description | Man Page |
|---|---|---|
| `in.dhcpd` | The DHCP server daemon. The daemon is started when the system is started. You should not start the server daemon directly. Use DHCP Manager, the `svcadm` command, or `dhcpconfig` to start and stop the daemon. The daemon should be invoked directly only to run the server in debug mode to troubleshoot problems. | `in.dhcpd`(1M) |
| `dhcpmgr` | The DHCP Manager, a graphical user interface (GUI) tool used to configure and manage the DHCP service. DHCP Manager is the recommended Solaris DHCP management tool. | `dhcpmgr`(1M) |
| `ifconfig` | Used at system boot to assign IP addresses to network interfaces, configure network interface parameters, or both. On a Solaris DHCP client, `ifconfig` starts DHCP to get the parameters (including the IP address) needed to configure a network interface. | `ifconfig`(1M) |
| `dhcpinfo` | Used by system startup scripts on Solaris client systems to obtain information (such as the host name) from the DHCP client daemon, `dhcpagent`. You can also use `dhcpinfo` in scripts or at the command line to obtain specified parameter values. | `dhcpinfo`(1) |
| `snoop` | Used to capture and display the contents of packets being passed across the network. `snoop` is useful for troubleshooting problems with the DHCP service. | `snoop`(1M) |
| `dhcpagent` | The DHCP client daemon, which implements the client side of the DHCP protocol. | `dhcpagent`(1M) |

# Running DHCP Commands in Scripts

The `dhcpconfig`, `dhtadm`, and `pntadm` commands are optimized for use in scripts. In particular, the `pntadm` command is useful for creating a large number of IP address entries in a DHCP network table. The following sample script uses `pntadm` in batch mode to create IP addresses.

**EXAMPLE 18–1** `addclient.ksh` Script With the `pntadm` Command

```
#! /usr/bin/ksh
#
# This script utilizes the pntadm batch facility to add client entries
# to a DHCP network table. It assumes that the user has the rights to
# run pntadm to add entries to DHCP network tables.

#
# Based on the nsswitch setting, query the netmasks table for a netmask.
# Accepts one argument, a dotted IP address.
#
get_netmask()
```

**EXAMPLE 18–1** addclient.ksh Script With the pntadm Command *(Continued)*

```
{
    MTMP=‘getent netmasks ${1} | awk ’{ print $2 }’‘
    if [ ! -z "${MTMP}" ]
    then
        print - ${MTMP}
    fi
}

#
# Based on the network specification, determine whether or not network is
# subnetted or supernetted.
# Given a dotted IP network number, convert it to the default class
# network.(used to detect subnetting). Requires one argument, the
# network number. (e.g. 10.0.0.0) Echos the default network and default
# mask for success, null if error.
#
get_default_class()
{
    NN01=${1%%.*}
    tmp=${1#*.}
    NN02=${tmp%%.*}
    tmp=${tmp#*.}
    NN03=${tmp%%.*}
    tmp=${tmp#*.}
    NN04=${tmp%%.*}
    RETNET=""
    RETMASK=""

    typeset -i16 ONE=10#${1%%.*}
    typeset -i10 X=$((${ONE}&16#f0))
    if [ ${X} -eq 224 ]
    then
        # Multicast
        typeset -i10 TMP=$((${ONE}&16#f0))
        RETNET="${TMP}.0.0.0"
        RETMASK="240.0.0.0"
    fi
    typeset -i10 X=$((${ONE}&16#80))
    if [ -z "${RETNET}" -a ${X} -eq 0 ]
    then
        # Class A
        RETNET="${NN01}.0.0.0"
        RETMASK="255.0.0.0"
    fi
    typeset -i10 X=$((${ONE}&16#c0))
    if [ -z "${RETNET}" -a ${X} -eq 128 ]
```

**EXAMPLE 18–1** addclient.ksh Script With the pntadm Command *(Continued)*

```
    then
        # Class B
        RETNET="${NN01}.${NN02}.0.0"
        RETMASK="255.255.0.0"
    fi
    typeset -i10 X=$((${ONE}&16#e0))
    if [ -z "${RETNET}" -a ${X} -eq 192 ]
    then
        # Class C
        RETNET="${NN01}.${NN02}.${NN03}.0"
        RETMASK="255.255.255.0"
    fi
    print - ${RETNET} ${RETMASK}
    unset NNO1 NNO2 NNO3 NNO4 RETNET RETMASK X ONE
}

#
# Given a dotted form of an IP address, convert it to its hex equivalent.
#
convert_dotted_to_hex()
{
    typeset -i10 one=${1%%.*}
    typeset -i16 one=${one}
    typeset -Z2 one=${one}
    tmp=${1#*.}

    typeset -i10 two=${tmp%%.*}
    typeset -i16 two=${two}
    typeset -Z2 two=${two}
    tmp=${tmp#*.}

    typeset -i10 three=${tmp%%.*}
    typeset -i16 three=${three}
    typeset -Z2 three=${three}
    tmp=${tmp#*.}

    typeset -i10 four=${tmp%%.*}
    typeset -i16 four=${four}
    typeset -Z2 four=${four}

     hex=‘print - ${one}${two}${three}${four} | sed -e ’s/#/0/g’‘
     print - 16#${hex}
     unset one two three four tmp
}

#
```

**EXAMPLE 18–1** addclient.ksh Script With the pntadm Command *(Continued)*

```
# Generate an IP address given the network address, mask, increment.
#
get_addr()
{
    typeset -i16 net=‘convert_dotted_to_hex ${1}‘
    typeset -i16 mask=‘convert_dotted_to_hex ${2}‘
    typeset -i16 incr=10#${3}

    # Maximum legal value - invert the mask, add to net.
    typeset -i16 mhosts=~${mask}
    typeset -i16 maxnet=${net}+${mhosts}

    # Add the incr value.
    let net=${net}+${incr}

    if [ $((${net} < ${maxnet})) -eq 1 ]
    then
        typeset -i16 a=${net}\&16#ff000000
        typeset -i10 a="${a}>>24"

        typeset -i16 b=${net}\&16#ff0000
        typeset -i10 b="${b}>>16"

        typeset -i16 c=${net}\&16#ff00
        typeset -i10 c="${c}>>8"

        typeset -i10 d=${net}\&16#ff
        print - "${a}.${b}.${c}.${d}"
    fi
    unset net mask incr mhosts maxnet a b c d
}

# Given a network address and client address, return the index.
client_index()
{
    typeset -i NNO1=${1%%.*}
    tmp=${1#*.}
    typeset -i NNO2=${tmp%%.*}
    tmp=${tmp#*.}
    typeset -i NNO3=${tmp%%.*}
    tmp=${tmp#*.}
    typeset -i NNO4=${tmp%%.*}

    typeset -i16 NNF1
    let NNF1=${NNO1}
    typeset -i16 NNF2
```

**EXAMPLE 18–1** `addclient.ksh` Script With the `pntadm` Command *(Continued)*

```
let NNF2=${NNO2}
typeset -i16 NNF3
let NNF3=${NNO3}
typeset -i16 NNF4
let NNF4=${NNO4}
typeset +i16 NNF1
typeset +i16 NNF2
typeset +i16 NNF3
typeset +i16 NNF4
NNF1=${NNF1#16\#}
NNF2=${NNF2#16\#}
NNF3=${NNF3#16\#}
NNF4=${NNF4#16\#}
if [ ${#NNF1} -eq 1 ]
then
    NNF1="0${NNF1}"
fi
if [ ${#NNF2} -eq 1 ]
then
    NNF2="0${NNF2}"
fi
if [ ${#NNF3} -eq 1 ]
then
    NNF3="0${NNF3}"
fi
if [ ${#NNF4} -eq 1 ]
then
    NNF4="0${NNF4}"
fi
typeset -i16 NN
let NN=16#${NNF1}${NNF2}${NNF3}${NNF4}
unset NNF1 NNF2 NNF3 NNF4

typeset -i NNO1=${2%%.*}
tmp=${2#*.}
typeset -i NNO2=${tmp%%.*}
tmp=${tmp#*.}
typeset -i NNO3=${tmp%%.*}
tmp=${tmp#*.}
typeset -i NNO4=${tmp%%.*}
typeset -i16 NNF1
let NNF1=${NNO1}
typeset -i16 NNF2
let NNF2=${NNO2}
typeset -i16 NNF3
let NNF3=${NNO3}
```

**EXAMPLE 18–1** addclient.ksh Script With the pntadm Command *(Continued)*

```
    typeset -i16 NNF4
    let NNF4=${NNO4}
    typeset +i16 NNF1
    typeset +i16 NNF2
    typeset +i16 NNF3
    typeset +i16 NNF4
    NNF1=${NNF1#16\#}
    NNF2=${NNF2#16\#}
    NNF3=${NNF3#16\#}
    NNF4=${NNF4#16\#}
    if [ ${#NNF1} -eq 1 ]
    then
        NNF1="0${NNF1}"
    fi
    if [ ${#NNF2} -eq 1 ]
    then
        NNF2="0${NNF2}"
    fi
    if [ ${#NNF3} -eq 1 ]
    then
        NNF3="0${NNF3}"
    fi
    if [ ${#NNF4} -eq 1 ]
    then
        NNF4="0${NNF4}"
    fi
    typeset -i16 NC
    let NC=16#${NNF1}${NNF2}${NNF3}${NNF4}
    typeset -i10 ANS
    let ANS=${NC}-${NN}
    print - $ANS
}

#
# Check usage.
#
if [ "$#" != 3 ]
then
    print "This script is used to add client entries to a DHCP network"
    print "table by utilizing the pntadm batch facilty.\n"
    print "usage: $0 network start_ip entries\n"
    print "where: network is the IP address of the network"
        print "       start_ip is the starting IP address \n"
        print "       entries is the number of the entries to add\n"
    print "example: $0 10.148.174.0 10.148.174.1 254\n"
    return
```

**EXAMPLE 18–1** addclient.ksh Script With the pntadm Command *(Continued)*

```
fi

#
# Use input arguments to set script variables.
#
NETWORK=$1
START_IP=$2
typeset -i STRTNUM=‘client_index ${NETWORK} ${START_IP}‘
let ENDNUM=${STRTNUM}+$3
let ENTRYNUM=${STRTNUM}
BATCHFILE=/tmp/batchfile.$$
MACRO=‘uname -n‘

#
# Check if mask in netmasks table. First try
# for network address as given, in case VLSM
# is in use.
#
NETMASK=‘get_netmask ${NETWORK}‘
if [ -z "${NETMASK}" ]
then
    get_default_class ${NETWORK} | read DEFNET DEFMASK
    # use the default.
    if [ "${DEFNET}" != "${NETWORK}" ]
    then
        # likely subnetted/supernetted.
        print - "\n\n###\tWarning\t###\n"
        print - "Network ${NETWORK} is netmasked, but no entry was found  \n
              in the ’netmasks’ table; please update the ’netmasks’  \n
              table in the appropriate nameservice before continuing. \n
              (See /etc/nsswitch.conf.) \n" >&2
        return 1
    else
        # use the default.
        NETMASK="${DEFMASK}"
    fi
fi

#
# Create a batch file.
#
print -n "Creating batch file "
while [ ${ENTRYNUM} -lt ${ENDNUM} ]
do
    if [ $((${ENTRYNUM}-${STRTNUM}))%50 -eq 0 ]
    then
```

**EXAMPLE 18–1** addclient.ksh Script With the pntadm Command *(Continued)*

```
        print -n "."
    fi

    CLIENTIP=‘get_addr ${NETWORK} ${NETMASK} ${ENTRYNUM}‘
    print "pntadm -A ${CLIENTIP} -m ${MACRO} ${NETWORK}" >> ${BATCHFILE}
    let ENTRYNUM=${ENTRYNUM}+1
done
print " done.\n"

#
# Run pntadm in batch mode and redirect output to a temporary file.
# Progress can be monitored by using the output file.
#
print "Batch processing output redirected to ${BATCHFILE}"
print "Batch processing started."

pntadm -B ${BATCHFILE} -v > /tmp/batch.out 2 >&1

print "Batch processing completed."
```

# Files Used by the DHCP Service

The following table lists files associated with Solaris DHCP.

**TABLE 18–2** Files and Tables Used by DHCP Daemons and Commands

| File or Table Name | Description | Man Page |
|---|---|---|
| dhcptab | A generic term for the table of DHCP configuration information that is recorded as options with assigned values, which are then grouped into macros. The name of the dhcptab table and its location is determined by the data store you use for DHCP information. | dhcptab(4) |
| DHCP network table | Maps IP addresses to client IDs and configuration options. DHCP network tables are named according to the IP address of the network, such as 10.21.32.0. There is no file that is called dhcp_network. The name and location of DHCP network tables is determined by the data store you use for DHCP information. | dhcp_network(4) |
| dhcpsvc.conf | Stores startup options for the DHCP daemon and data store information. This file must not be edited manually. Use the dhcpconfig command to change startup options. | dhcpsvc.conf(4) |

**TABLE 18–2** Files and Tables Used by DHCP Daemons and Commands *(Continued)*

| File or Table Name | Description | Man Page |
|---|---|---|
| `nsswitch.conf` | Specifies the location of name service databases and the order in which to search name services for various kinds of information. The `nsswitch.conf` file is read to obtain accurate configuration information when you configure a DHCP server. The file is located in the `/etc` directory. | `nsswitch.conf`(4) |
| `resolv.conf` | Contains information used to resolve DNS queries. During DHCP server configuration, this file is consulted for information about the DNS domain and DNS server. The file is located in the `/etc` directory. | `resolv.conf`(4) |
| `dhcp.`*interface* | Indicates that DHCP is to be used on the client's network interface that is specified in the `dhcp.`*interface* file name. For example, the existence of a file named `dhcp.qe0` indicates that DHCP is to be used on the `qe0` interface. The `dhcp.`*interface* file might contain commands that are passed as options to the `ifconfig` command, which is used to start DHCP on the client. The file is located in the `/etc` directory on Solaris DHCP client systems. | No specific man page, see `dhcp`(5) |
| *interface*`.dhc` | Contains the configuration parameters that are obtained from DHCP for the given network interface. The client caches the current configuration information in `/etc/dhcp/`*interface*`.dhc` when the interface's IP address lease is dropped. For example, if DHCP is used on the `qe0` interface, the `dhcpagent` caches the configuration information in `/etc/dhcp/qe0.dhc`. The next time DHCP starts on the interface, the client requests to use the cached configuration if the lease has not expired. If the DHCP server denies the request, the client begins the standard process for DHCP lease negotiation. | No specific man page, see `dhcpagent`(1M) |
| `dhcpagent` | Sets parameter values for the `dhcpagent` client daemon. The path to the file is `/etc/default/dhcpagent`. See the `/etc/default/dhcpagent` file or the `dhcpagent`(1M) man page for information about the parameters. | `dhcpagent`(1M) |

**TABLE 18–2** Files and Tables Used by DHCP Daemons and Commands *(Continued)*

| File or Table Name | Description | Man Page |
|---|---|---|
| DHCP `inittab` | Defines aspects of DHCP option codes, such as the data type, and assigns mnemonic labels. See the `dhcp_inittab`(4) man page for more information about the file syntax.<br><br>On the client, the information in the `/etc/dhcp/inittab` file is used by `dhcpinfo` to provide more meaningful information to human readers of the information. On the DHCP server system, this file is used by the DHCP daemon and management tools to obtain DHCP option information.<br><br>The `/etc/dhcp/inittab` file replaces the `/etc/dhcp/dhcptags` file that was used in previous releases. "DHCP Option Information" on page 471 provides more information about this replacement. | `dhcp_inittab`(4) |

# DHCP Option Information

Historically, DHCP option information has been stored in several places, including the server's `dhcptab` table, the client's `dhcptags` file, and internal tables of various programs. In the Solaris 8 release and later releases, the option information is consolidated in the `/etc/dhcp/inittab` file. See the `dhcp_inittab`(4) man page for detailed information about the file.

The Solaris DHCP client uses the DHCP `inittab` file as a replacement for the `dhcptags` file. The client uses the file to obtain information about option codes that were received in a DHCP packet. The `in.dhcpd`, `snoop`, and `dhcpmgr` programs on the DHCP server use the `inittab` file as well.

## Determining if Your Site Is Affected

Most sites that use Solaris DHCP are *not* affected by the switch to the `/etc/dhcp/inittab` file. Your site is affected if you meet all of the following criteria:

- You plan to upgrade from a Solaris release that is older than that the Solaris 8 release.
- You previously created new DHCP options.
- You modified the `/etc/dhcp/dhcptags` file, and you want to retain the changes.

When you upgrade, the upgrade log notifies you that your `dhcptags` file had been modified and that you should make changes to the DHCP `inittab` file.

## Differences Between `dhcptags` and `inittab` Files

The `inittab` file contains more information than the `dhcptags` file. The `inittab` file also uses a different syntax.

A sample `dhcptags` entry is as follows:

```
33 StaticRt - IPList Static_Routes
```

`33` is the numeric code that is passed in the DHCP packet. `StaticRt` is the option name. `IPList` indicates that the data type for `StaticRt` must be a list of IP addresses. `Static_Routes` is a more descriptive name.

The `inittab` file consists of one-line records that describe each option. The format is similar to the format that defines symbols in `dhcptab`. The following table describes the syntax of the `inittab` file.

| Option | Description |
|---|---|
| *option-name* | Name of the option. The option name must be unique within its option category, and not overlap with other option names in the Standard, Site, and Vendor categories. For example, you cannot have two Site options with the same name, and you should not create a Site option with the same name as a Standard option. |
| *category* | Identifies the namespace in which the option belongs. Must be one of the following: Standard, Site, Vendor, Field, or Internal. |
| *code* | Identifies the option when sent over the network. In most cases, the code uniquely identifies the option, without a category. However, in the case of internal categories such as Field or Internal, a code might be used for other purposes. The code might not be globally unique. The code should be unique within the option's category, and not overlap with codes in the Standard and Site fields. |
| *type* | Describes the data that is associated with this option. Valid types are IP, ASCII, Octet, Boolean, Unumber8, Unumber16, Unumber32, Unumber64, Snumber8, Snumber16, Snumber32, and Snumber64. For numbers, an initial U or S indicates that the number is unsigned or signed. The digits at the end indicate how many bits are in the number. For example, Unumber8 is an unsigned 8-bit number. The type is not case sensitive. |
| *granularity* | Describes how many units of data make up a whole value for this option. |
| *maximum* | Describes how many whole values are allowed for this option. 0 indicates an infinite number. |
| *consumers* | Describes which programs can use this information. Consumers should be set to `sdmi`, where: |

```
s    snoop
d    in.dhcpd
m    dhcpmgr
i    dhcpinfo
```

A sample `inittab` entry is as follows:

```
StaticRt - Standard, 33, IP, 2, 0, sdmi
```

This entry describes an option that is named `StaticRt`. The option is in the Standard category, and is option code 33. The expected data is a potentially infinite number of pairs of IP addresses because the type is `IP`, the granularity is `2`, and the maximum is infinite (`0`). The consumers of this option are `sdmi`: `snoop`, `in.dhcpd`, `dhcpmgr`, and `dhcpinfo`.

## Converting `dhcptags` Entries to `inittab` Entries

If you previously added entries to your `dhcptags` file, you must add corresponding entries to the new `inittab` file if you want to continue using the options you added to your site. The following example shows how a sample `dhcptags` entry might be expressed in `inittab` format.

Suppose you had added the following `dhcptags` entry for fax machines that are connected to the network:

```
128 FaxMchn - IP Fax_Machine
```

The code `128` means that the option must be in the Site category. The option name is `FaxMchn`, and the data type is `IP`.

The corresponding `inittab` entry might be:

```
FaxMchn SITE, 128, IP, 1, 1, sdmi
```

The granularity of 1 and the maximum of 1 indicate that one IP address is expected for this option.

PART IV

# IP Security

This section focuses on network security. IP security architecture (IPsec) protects the network at the packet level. Internet key management (IKE) manages the keys for IPsec. Solaris IP filter provides a firewall.

◆ ◆ ◆ CHAPTER 19

# IP Security Architecture (Overview)

The IP Security Architecture (IPsec) provides cryptographic protection for IP datagrams in IPv4 and IPv6 network packets.

This chapter contains the following information:


- "What's New in IPsec?" on page 477
- "Introduction to IPsec" on page 478
- "IPsec Packet Flow" on page 480
- "IPsec Security Associations" on page 482
- "IPsec Protection Mechanisms" on page 484
- "IPsec Protection Policies" on page 486
- "Transport and Tunnel Modes in IPsec" on page 487
- "Virtual Private Networks and IPsec" on page 489
- "IPsec and NAT Traversal" on page 490
- "IPsec and SCTP" on page 491
- "IPsec and Solaris Zones" on page 491
- "IPsec Utilities and Files" on page 491
- "Changes to IPsec for the Solaris 10 Release" on page 492


To implement IPsec on your network, see Chapter 20, "Configuring IPsec (Tasks)." For reference information, see Chapter 21, "IP Security Architecture (Reference)."

## What's New in IPsec?

**Solaris 10 8/07**: Starting in this release, IPsec fully implements tunnels in tunnel mode, and the utilities that support tunnels are modified.

- IPsec implements tunnels in tunnel mode for virtual private networks (VPNs). In tunnel mode, IPsec supports multiple clients behind a single NAT. In tunnel mode, IPsec is interoperable with implementations of IP-in-IP tunnels by other vendors. IPsec continues to support tunnels in transport mode, so it is compatible with earlier Solaris releases.

- The syntax to create a tunnel is simplified. To manage IPsec policy, the `ipsecconf` command has been expanded. The `ifconfig` command is deprecated for managing IPsec policy.
- Starting in this release, the `/etc/ipnodes` file is removed. Use the `/etc/hosts` file to configure network IPv6 addresses.

**Solaris 10 1/06**: Starting in this release, IKE is fully compliant with NAT-Traversal support as described in RFC 3947 and RFC 3948. IKE operations use the PKCS #11 library from the cryptographic framework, which improves performance.

The cryptographic framework provides a softtoken keystore for applications that use the metaslot. When IKE uses the metaslot, you have the option of storing the keys on disk, on an attached board, or in the softtoken keystore.

- To use the softtoken keystore, see the `cryptoadm`(1M) man page.
- For a complete listing of new Solaris features and a description of Solaris releases, see *Solaris Express Developer Edition What's New*.

# Introduction to IPsec

IPsec protects IP packets by authenticating the packets, by encrypting the packets, or by doing both. IPsec is performed inside the IP module, well below the application layer. Therefore, an Internet application can take advantage of IPsec while not having to configure itself to use IPsec. When used properly, IPsec is an effective tool in securing network traffic.

IPsec protection involves five main components:

- **Security protocols –** The IP datagram protection mechanisms. The authentication header (AH) signs IP packets and ensures integrity. The content of the datagram is not encrypted, but the receiver is assured that the packet contents have not been altered. The receiver is also assured that the packets were sent by the sender. The encapsulating security payload (ESP) encrypts IP data, thus obscuring the content during packet transmission. ESP also can ensure data integrity through an authentication algorithm option.
- **Security associations database (SADB) –** The database that associates a security protocol with an IP destination address and an indexing number. The indexing number is called the security parameter index (SPI). These three elements (the security protocol, the destination address, and the SPI) uniquely identify a legitimate IPsec packet. The database ensures that a protected packet that arrives to the packet destination is recognized by the receiver. The receiver also uses information from the database to decrypt the communication, verify that the packets are unchanged, reassemble the packets, and deliver the packets to their ultimate destination.
- **Key management –** The generation and distribution of keys for the cryptographic algorithms and for the SPI.

- **Security mechanisms –** The authentication and encryption algorithms that protect the data in the IP datagrams.
- **Security policy database (SPD) –** The database that specifies the level of protection to apply to a packet. The SPD filters IP traffic to determine how the packets should be processed. A packet can be discarded. A packet can be passed in the clear. Or, a packet can be protected with IPsec. For outbound packets, the SPD and the SADB determine what level of protection to apply. For inbound packets, the SPD helps to determine if the level of protection on the packet is acceptable. If the packet is protected by IPsec, the SPD is consulted after the packet has been decrypted and has been verified.

When you invoke IPsec, IPsec applies the security mechanisms to IP datagrams that travel to the IP destination address. The receiver uses information in its SADB to verify that the arriving packets are legitimate and to decrypt them. Applications can invoke IPsec to apply security mechanisms to IP datagrams on a per-socket level as well.

Note that sockets behave differently from ports:

- Per-socket SAs override their corresponding port entry in the SPD.
- Also, if a socket on a port is connected, and IPsec policy is later applied to that port, then traffic that uses that socket is not protected by IPsec.

  Of course, a socket that is opened on a port *after* IPsec policy is applied to the port is protected by IPsec policy.

# IPsec RFCs

The Internet Engineering Task Force (IETF) has published a number of Requests for Comment (RFCs) that describe the security architecture for the IP layer. All RFCs are copyrighted by the Internet Society. For a link to the RFCs, see `http://ietf.org/`. The following list of RFCs covers the more general IP security references:

- RFC 2411, "IP Security Document Roadmap," November 1998
- RFC 2401, "Security Architecture for the Internet Protocol," November 1998
- RFC 2402, "IP Authentication Header," November 1998
- RFC 2406, "IP Encapsulating Security Payload (ESP)," November 1998
- RFC 2408, "Internet Security Association and Key Management Protocol (ISAKMP)," November 1998
- RFC 2407, "The Internet IP Security Domain of Interpretation for ISAKMP," November 1998
- RFC 2409, "The Internet Key Exchange (IKE)," November 1998
- RFC 3554, "On the Use of Stream Control Transmission Protocol (SCTP) with IPsec," July 2003 [ not implemented in the Solaris 10 release ]

## IPsec Terminology

The IPsec RFCs define a number of terms that are useful to recognize when implementing IPsec on your systems. The following table lists IPsec terms, provides their commonly used acronyms, and defines each term. For a list of terminology used in key negotiation, see Table 22–1.

**TABLE 19–1** IPsec Terms, Acronyms, and Uses

| IPsec Term | Acronym | Definition |
|---|---|---|
| Security association | SA | A unique connection between two nodes on a network. The connection is defined by a triplet: a security protocol, a security parameter index, and an IP destination. The IP destination can be an IP address or a socket. |
| Security associations database | SADB | Database that contains all active security associations. |
| Security parameter index | SPI | The indexing value for a security association. An SPI is a 32-bit value that distinguishes among SAs that have the same IP destination and security protocol. |
| Security policy database | SPD | Database that determines if outbound packets and inbound packets have the specified level of protection. |
| Key exchange | | The process of generating keys for asymmetric cryptographic algorithms. The two main methods are RSA protocols and the Diffie-Hellman protocol. |
| Diffie-Hellman protocol | DH | A key exchange protocol that involves key generation and key authentication. Often called *authenticated key exchange*. |
| RSA protocol | RSA | A key exchange protocol that involves key generation and key distribution. The protocol is named for its three creators, Rivest, Shamir, and Adleman. |
| Internet Security Association and Key Management Protocol | ISAKMP | The common framework for establishing the format of SA attributes, and for negotiating, modifying, and deleting SAs. ISAKMP is the IETF standard for handling IPsec SAs. |

# IPsec Packet Flow

Figure 19–1 shows how an IP addressed packet, as part of an IP datagram, proceeds when IPsec has been invoked on an outbound packet. The flow diagram illustrates where authentication header (AH) and encapsulating security payload (ESP) entities can be applied to the packet. How to apply these entities, as well as how to choose the algorithms, are described in subsequent sections.

Figure 19–2 shows the IPsec inbound process.

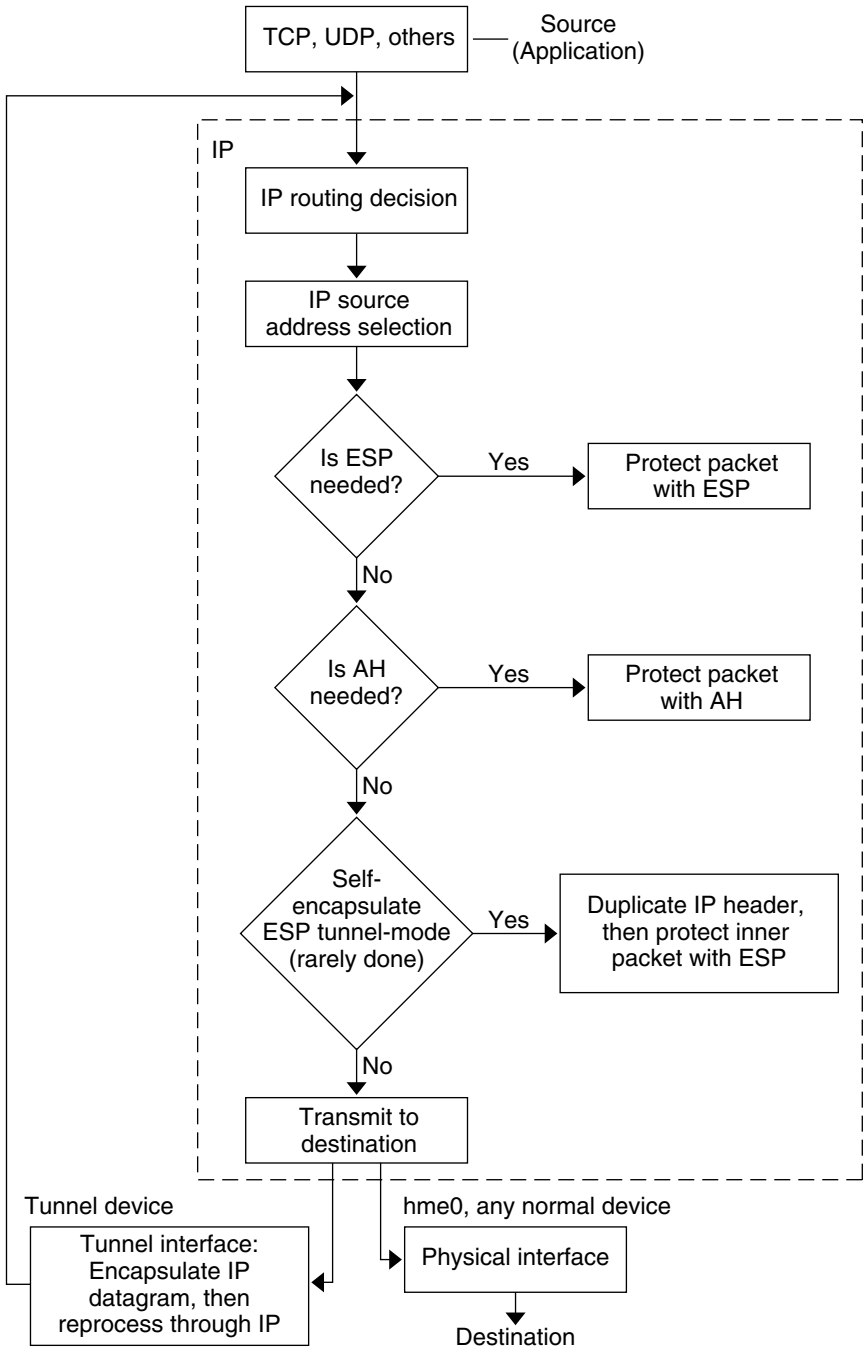



**FIGURE 19–1** IPsec Applied to Outbound Packet Process

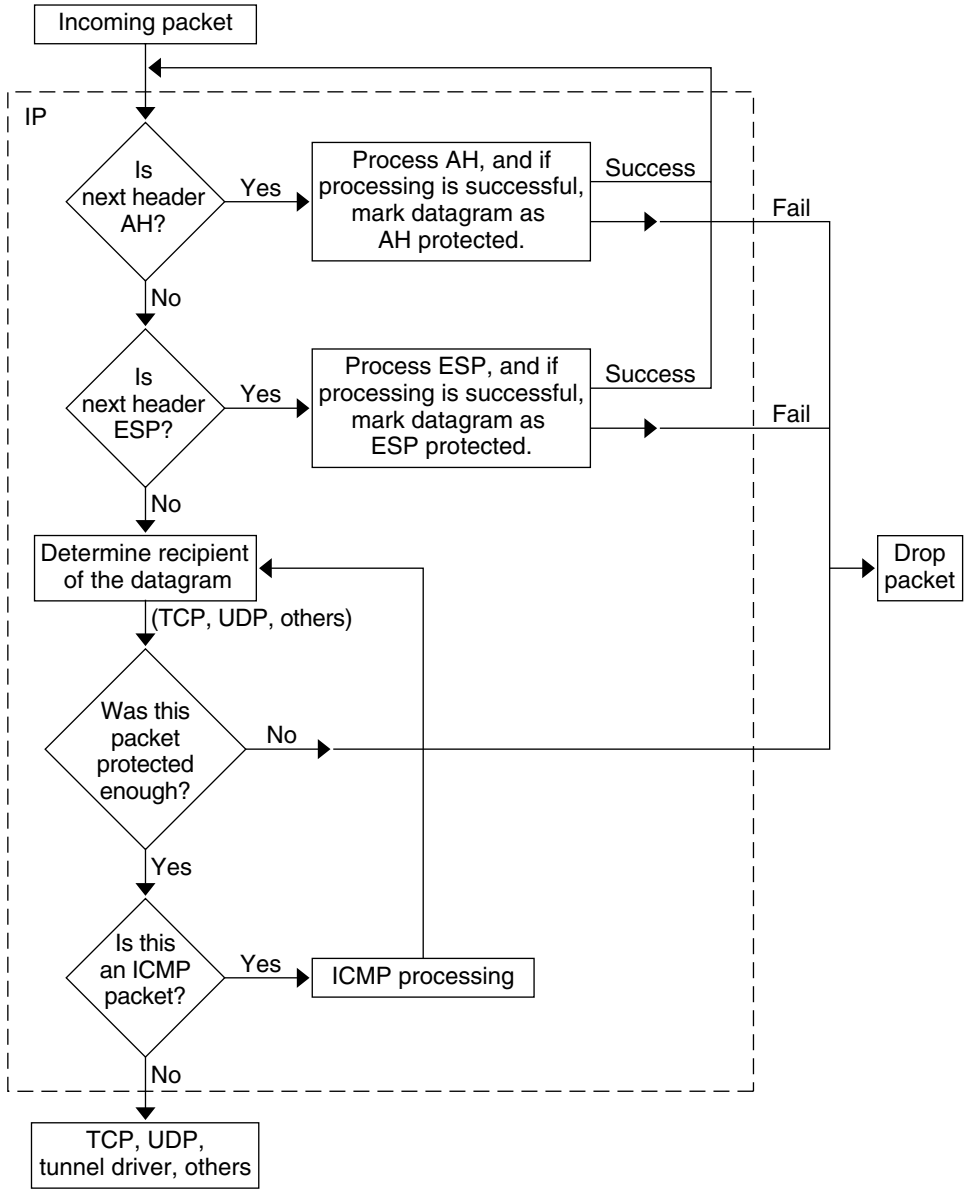



**FIGURE 19–2** IPsec Applied to Inbound Packet Process

# IPsec Security Associations

An IPsec *security association* (SA) specifies security properties that are recognized by communicating hosts. A single SA protects data in one direction. The protection is either to a

single host or to a group (multicast) address. Because most communication is either peer-to-peer or client-server, two SAs must be present to secure traffic in both directions.

The following three elements uniquely identify an IPsec SA:

- The security protocol (AH or ESP)
- The destination IP address
- The security parameter index (SPI)

The SPI, an arbitrary 32-bit value, is transmitted with an AH or ESP packet. The `ipsecah`(7P) and `ipsecesp`(7P) man pages explain the extent of protection that is provided by AH and ESP. An integrity checksum value is used to authenticate a packet. If the authentication fails, the packet is dropped.

Security associations are stored in a *security associations database* (SADB). A socket-based administration engine, the `pf_key` interface, enables privileged applications to manage the database.

- For a more complete description of the IPsec SADB, see "Security Associations Database for IPsec" on page 540.
- For more information about how to manage the SADB, see the `pf_key`(7P) man page.

# Key Management in IPsec

Security associations (SAs) require material to create the keys for authentication and for encryption. The managing of this *keying material* is called *key management*. The Internet Key Exchange (IKE) protocol handles key management automatically. You can also manage keys manually with the `ipseckey` command.

SAs on IPv4 and IPv6 packets can use either method of key management. Unless you have an overriding reason to use manual key management, automatic key management is preferred. For example, to interoperate with systems other than Solaris systems might require manual key management.

- The `in.iked` daemon provides automatic key management. For a description of IKE, see Chapter 22, "Internet Key Exchange (Overview)." For more information on the `in.iked` daemon, see the `in.iked`(1M) man page.
- The `ipseckey` command provides manual key management. For a description of the command, see "Utilities for Key Generation in IPsec" on page 541. For a detailed description of the `ipseckey` command options, see the `ipseckey`(1M) man page.

# IPsec Protection Mechanisms

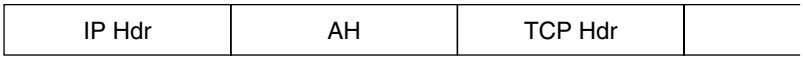
IPsec provides two security protocols for protecting data:

- Authentication Header (AH)
- Encapsulating Security Payload (ESP)

An AH protects data with an authentication algorithm. An ESP protects data with an encryption algorithm. Optionally, an ESP protects data with an authentication algorithm. Each implementation of an algorithm is called a *mechanism*.

## Authentication Header

The authentication header provides data authentication, strong integrity, and replay protection to IP datagrams. AH protects the greater part of the IP datagram. As the following illustration shows, AH is inserted between the IP header and the transport header.




The transport header can be TCP, UDP, SCTP, or ICMP. If a tunnel is being used, the transport header can be another IP header.

## Encapsulating Security Payload

The encapsulating security payload (ESP) module provides confidentiality over what the ESP encapsulates. ESP also provides the services that AH provides. However, ESP only provides its protections over the part of the datagram that ESP encapsulates. The authentication services of ESP are optional. These services enable you to use ESP and AH together on the same datagram without redundancy. Because ESP uses encryption-enabling technology, ESP must conform to U.S.export control laws.

ESP encapsulates its data, so ESP only protects the data that follows its beginning in the datagram, as shown in the following illustration.


In a TCP packet, ESP encapsulates only the TCP header and its data. If the packet is an IP-in-IP datagram, ESP protects the inner IP datagram. Per-socket policy allows *self-encapsulation*, so ESP can encapsulate IP options when ESP needs to.

If self-encapsulation is set, a copy of the IP header is made to construct an IP-in-IP datagram. For example, when self-encapsulation is not set on a TCP socket, the datagram is sent in the following format:

```
[ IP(a -> b) options + TCP + data ]
```

When self-encapsulation is set on that TCP socket, the datagram is sent in the following format:

```
[ IP(a -> b) + ESP [ IP(a -> b) options + TCP + data ] ]
```

For further discussion, see "Transport and Tunnel Modes in IPsec" on page 487.

### Security Considerations When Using AH and ESP

The following table compares the protections that are provided by AH and ESP.

**TABLE 19–2** Protections Provided by AH and ESP in IPsec

| Protocol | Packet Coverage | Protection | Against Attacks |
|---|---|---|---|
| AH | Protects packet from the IP header to the transport header | Provides strong integrity, data authentication:<br>■ Ensures that the receiver receives exactly what the sender sent<br>■ Is susceptible to replay attacks when an AH does not enable replay protection | Replay, cut-and-paste |
| ESP | Protects packet following the beginning of ESP in the datagram. | With encryption option, encrypts the IP datagram. Ensures confidentiality | Eavesdropping |
| | | With authentication option, provides the same protection as AH | Replay, cut-and-paste |
| | | With both options, provides strong integrity, data authentication, and confidentiality | Replay, cut-and-paste, eavesdropping |

## Authentication and Encryption Algorithms in IPsec

IPsec security protocols use two types of algorithms, authentication and encryption. The AH module uses authentication algorithms. The ESP module can use encryption as well as authentication algorithms. You can obtain a list of the algorithms on your system and their properties by using the `ipsecalgs` command. For more information, see the `ipsecalgs`(1M) man page. You can also use the functions that are described in the `getipsecalgbyname`(3NSL) man page to retrieve the properties of algorithms.

IPsec on a Solaris system uses the Solaris cryptographic framework to access the algorithms. The framework provides a central repository for algorithms, in addition to other services. The framework enables IPsec to take advantage of high performance cryptographic hardware

accelerators. The framework also provides resource control features. For example, the framework enables you to limit the amount of CPU time spent in cryptographic operations in the kernel.

For more information, see the following:

- Chapter 13, "Solaris Cryptographic Framework (Overview)," in *System Administration Guide: Security Services*
- Chapter 8, "Introduction to the Solaris Cryptographic Framework," in *Solaris Security for Developers Guide*

### Authentication Algorithms in IPsec

Authentication algorithms produce an integrity checksum value or *digest* that is based on the data and a key. The AH module uses authentication algorithms. The ESP module can use authentication algorithms as well.

### Encryption Algorithms in IPsec

Encryption algorithms encrypt data with a key. The ESP module in IPsec uses encryption algorithms. The algorithms operate on data in units of a *block size*. By default, the DES-CBC, 3DES-CBC, AES-CBC, and Blowfish-CBC algorithms are installed. The key sizes that are supported by the AES-CBC and Blowfish-CBC algorithms are limited to 128 bits.

AES-CBC and Blowfish-CBC algorithms that support key sizes that are greater than 128 bits are available to IPsec when you install the Solaris Encryption Kit. However, not all encryption algorithms are available outside of the United States. The kit is available on a separate CD that is *not* part of the Solaris 10 installation box. The *Solaris 10 Encryption Kit Installation Guide* describes how to install the kit. For more information, see the Sun Downloads web site (`http://www.sun.com/download`). To download the kit, click the Downloads A-Z tab, then click the letter S. The Solaris 10 Encryption Kit is among the first 20 entries.

## IPsec Protection Policies

IPsec protection policies can use any of the security mechanisms. IPsec policies can be applied at the following levels:

- On a system-wide level
- On a per-socket level

IPsec applies the system-wide policy to outbound datagrams and inbound datagrams. Outbound datagrams are either sent with protection or without protection. If protection is applied, the algorithms are either specific or non-specific. You can apply some additional rules to outbound datagrams, because of the additional data that is known by the system. Inbound datagrams can be either accepted or dropped. The decision to drop or accept an inbound

datagram is based on several criteria, which sometimes overlap or conflict. Conflicts are resolved by determining which rule is parsed first. The traffic is automatically accepted, except when a policy entry states that traffic should bypass all other policies.

The policy that normally protects a datagram can be bypassed. You can either specify an exception in the system-wide policy, or you can request a bypass in the per-socket policy. For traffic within a system, policies are enforced, but actual security mechanisms are not applied. Instead, the outbound policy on an intra-system packet translates into an inbound packet that has had those mechanisms applied.

You use the `ipsecinit.conf` file and the `ipsecconf` command to configure IPsec policies. For details and examples, see the `ipsecconf`(1M) man page.

# Transport and Tunnel Modes in IPsec

The IPsec standards define two distinct modes of IPsec operation, *transport mode* and *tunnel mode*. The modes do not affect the encoding of packets. The packets are protected by AH, ESP, or both in each mode. The modes differ in policy application when the inner packet is an IP packet, as follows:

- In transport mode, the outer header determines the IPsec policy that protects the inner IP packet.
- In tunnel mode, the inner IP packet determines the IPsec policy that protects its contents.

In transport mode, the outer header, the next header, and any ports that the next header supports, can be used to determine IPsec policy. In effect, IPsec can enforce different transport mode policies between two IP addresses to the granularity of a single port. For example, if the next header is TCP, which supports ports, then IPsec policy can be set for a TCP port of the outer IP address. Similarly, if the next header is an IP header, the outer header and the inner IP header can be used to determine IPsec policy.

Tunnel mode works only for IP-in-IP datagrams. Tunneling in tunnel mode can be useful when computer workers at home are connecting to a central computer location. In tunnel mode, IPsec policy is enforced on the contents of the inner IP datagram. Different IPsec policies can be enforced for different inner IP addresses. That is, the inner IP header, its next header, and the ports that the next header supports, can enforce a policy. Unlike transport mode, in tunnel mode the outer IP header does not dictate the policy of its inner IP datagram.

Therefore, in tunnel mode, IPsec policy can be specified for subnets of a LAN behind a router and for ports on those subnets. IPsec policy can also be specified for particular IP addresses, that is, hosts, on those subnets. The ports of those hosts can also have a specific IPsec policy. However, if a dynamic routing protocol is run over a tunnel, do not use subnet selection or address selection because the view of the network topology on the peer network could change. Changes would invalidate the static IPsec policy. For examples of tunneling procedures that include configuring static routes, see "Protecting a VPN With IPsec" on page 509.

In the Solaris OS, tunnel mode can be enforced only on an IP tunneling network interface. The `ipsecconf` command provides a `tunnel` keyword to select an IP tunneling network interface. When the `tunnel` keyword is present in a rule, all selectors that are specified in that rule apply to the inner packet.

In transport mode, ESP, AH, or both, can protect the datagram.

The following figure shows an IP header with an unprotected TCP packet.

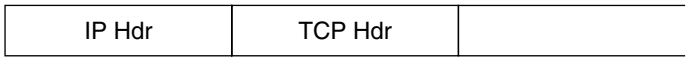



**FIGURE 19–3** Unprotected IP Packet Carrying TCP Information

In transport mode, ESP protects the data as shown in the following figure. The shaded area shows the encrypted part of the packet.


Encrypted

**FIGURE 19–4** Protected IP Packet Carrying TCP Information

In transport mode, AH protects the data as shown in the following figure.

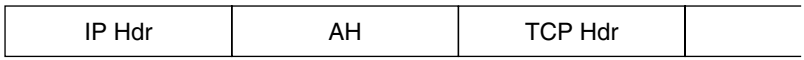



**FIGURE 19–5** Packet Protected by an Authentication Header

AH actually covers the data before the data appears in the datagram. Consequently, the protection that is provided by AH, even in transport mode, covers some of the IP header.

In tunnel mode, the entire datagram is *inside* the protection of an IPsec header. The datagram in Figure 19–3 is protected in tunnel mode by an outer IPsec header, and in this case ESP, as is shown in the following figure.


Encrypted

**FIGURE 19–6** IPsec Packet Protected in Tunnel Mode

The `ipsecconf` command includes keywords to set tunnels in tunnel mode or transport mode.

- For details on per-socket policy, see the `ipsec`(7P) man page.

- For an example of per-socket policy, see "How to Secure a Web Server With IPsec" on page 499.
- For more information about tunnels, see the `ipsecconf`(1M) man page.
- For an example of tunnel configuration, see "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode Over IPv4" on page 514.

# Virtual Private Networks and IPsec

A configured tunnel is a point-to-point interface. The tunnel enables one IP packet to be encapsulated within another IP packet. A correctly configured tunnel requires both a tunnel source and a tunnel destination. For more information, see the `tun`(7M)man page and "Configuring Tunnels for IPv6 Support" on page 189.

A tunnel creates an apparent physical interface to IP. The physical link's integrity depends on the underlying security protocols. If you set up the security associations (SAs) securely, then you can trust the tunnel. Packets that exit the tunnel must have originated from the peer that was specified in the tunnel destination. If this trust exists, you can use per-interface IP forwarding to create a virtual private network (VPN).

You can use IPsec to construct a VPN. IPsec secures the connection. For example, an organization that uses VPN technology to connect offices with separate networks can deploy IPsec to secure traffic between the two offices.

The following figure illustrates how two offices use the Internet to form their VPN with IPsec deployed on their network systems.

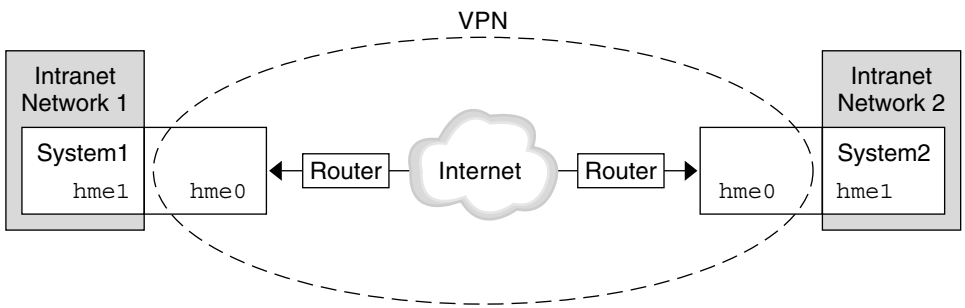



**FIGURE 19–7** Virtual Private Network

For a detailed example of the setup procedure, see "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode Over IPv4" on page 514. For IPv6 networks, see "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode Over IPv6" on page 520.

# IPsec and NAT Traversal

IKE can negotiate IPsec SAs across a NAT box. This ability enables systems to securely connect from a remote network, even when the systems are behind a NAT device. For example, employees who work from home, or who log on from a conference site can protect their traffic with IPsec.

NAT stands for network address translation. A NAT box is used to translate a private internal address into a unique Internet address. NATs are very common at public access points to the Internet, such as hotels. For a fuller discussion, see "Using Solaris IP Filter's NAT Feature" on page 612.

The ability to use IKE when a NAT box is between communicating systems is called NAT traversal, or NAT-T. In the Solaris 10 release, NAT-T has the following limitations:

- NAT-T works on IPv4 networks only.
- NAT-T cannot take advantage of the IPsec ESP acceleration provided by the Sun Crypto Accelerator 4000 board. However, IKE acceleration with the Sun Crypto Accelerator 4000 board works.
- The AH protocol depends on an unchanging IP header, therefore AH cannot work with NAT-T. The ESP protocol is used with NAT-T.
- The NAT box does not use special processing rules. A NAT box with special IPsec processing rules might interfere with the implementation of NAT-T.
- NAT-T works only when the IKE initiator is the system behind the NAT box. An IKE responder cannot be behind a NAT box unless the box has been programmed to forward IKE packets to the appropriate individual system behind the box.

The following RFCs describe NAT functionality and the limits of NAT-T. Copies of the RFCs can be retrieved from `http://www.rfc-editor.org`.

- RFC 3022, "Traditional IP Network Address Translator (Traditional NAT)," January 2001
- RFC 3715, "IPsec-Network Address Translation (NAT) Compatibility Requirements," March 2004
- RFC 3947, "Negotiation of NAT-Traversal in the IKE," January 2005
- RFC 3948, "UDP Encapsulation of IPsec Packets," January 2005

To use IPsec across a NAT, see "Configuring IKE for Mobile Systems (Task Map)" on page 583.

# IPsec and SCTP

The Solaris 10 release supports the Streams Control Transmission Protocol (SCTP). The use of the SCTP protocol and SCTP port number to specify IPsec policy is supported, but is not robust. The IPsec extensions for SCTP as specified in RFC 3554 are not yet implemented. These limitations can create complications in creating IPsec policy for SCTP.

SCTP can make use of multiple source and destination addresses in the context of a single SCTP association. When IPsec policy is applied to a single source or a single destination address, communication can fail when SCTP switches the source or the destination address of that association. IPsec policy only recognizes the original address. For information about SCTP, read the RFCs and "SCTP Protocol" on page 42.

# IPsec and Solaris Zones

IPsec is configured from the global zone. The IPsec policy configuration file, `ipsecinit.conf`, exists in the global zone only. The file can have entries that apply to non-global zones, as well as entries that apply to the global zone. For information about how to use IPsec with zones, see "Protecting Traffic With IPsec" on page 496. For information about zones, see Chapter 16, "Introduction to Solaris Zones," in *System Administration Guide: Virtualization Using the Solaris Operating System*.

# IPsec Utilities and Files

Table 19–3 describes the files and commandsthat are used to configure and manage IPsec. For completeness, the table includes key management files and commands.

- For instructions on implementing IPsec on your network, see "Protecting Traffic With IPsec (Task Map)" on page 495.
- For more details about IPsec utilities and files, see Chapter 21, "IP Security Architecture (Reference)."

**TABLE 19–3** List of Selected IPsec Files and Commands

| IPsec Utility or File | Description | Man Page |
|---|---|---|
| `/etc/inet/ipsecinit.conf` file | IPsec policy file. If this file exists, IPsec is activated at boot time. | `ipsecconf`(1M) |
| `ipsecconf` command | IPsec policy command. The boot scripts use`ipsecconf` to read the `/etc/inet/ipsecinit.conf` file and activate IPsec. Useful for viewing and modifying the current IPsec policy, and for testing. | `ipsecconf`(1M) |

**TABLE 19–3** List of Selected IPsec Files and Commands *(Continued)*

| IPsec Utility or File | Description | Man Page |
|---|---|---|
| PF_KEY socket interface | Interface for security associations database (SADB). Handles manual key management and automatic key management. | pf_key(7P) |
| ipseckey command | IPsec security associations (SAs)keying command. ipseckey is a command-line front end to the PF_KEY interface. ipseckey can create, destroy, or modify SAs. | ipseckey(1M) |
| /etc/inet/secret/ipseckeys file | Keys for IPsec SAs. If the ipsecinit.conf file exists, the ipseckeys file is automatically read at boot time. | |
| ipsecalgs command | IPsec algorithms command. Useful for viewing and modifying the list of IPsec algorithms and their properties. | ipsecalgs(1M) |
| /etc/inet/ipsecalgs file | Contains the configured IPsec protocols and algorithm definitions. This file is managed by the ipsecalgs utility and must never be edited manually. | |
| /etc/inet/ike/config file | IKE configuration and policy file. If this file exists, the IKE daemon, in.iked, provides automatic key management. The management is based on rules and global parameters in the /etc/inet/ike/config file. See "IKE Utilities and Files" on page 550. | ike.config(4) |

# Changes to IPsec for the Solaris 10 Release

For a complete listing of new Solaris features and a description of Solaris releases, see *Solaris Express Developer Edition What's New*. Since the Solaris 9 release, IPsec includes the following functionality:

- When a Sun Crypto Accelerator 4000 board is attached, the board automatically caches IPsec SAs for packets that use the board's Ethernet interface. The board also accelerates the processing of the IPsec SAs.
- IPsec can take advantage of automatic key management with IKE over IPv6 networks. For more information, see Chapter 22, "Internet Key Exchange (Overview)."

  For new IKE features, see "Changes to IKE for the Solaris 10 Release" on page 551.
- The parser for theipseckey command provides clearer help. The ipseckey monitor command timestamps each event. For details, see the ipseckey(1M) man page.
- IPsec algorithms now come from a central storage location, the Solaris cryptographic framework. The ipsecalgs(1M) man page describes the characteristics of the algorithms that are available. The algorithms are optimized for the architecture that they run on. For a description of the framework, see Chapter 13, "Solaris Cryptographic Framework (Overview)," in *System Administration Guide: Security Services*.

- IPsec works in the global zone. IPsec policy is managed in the global zone for a non-global zone. Keying material is created and is managed manually in the global zone for a non-global zone. IKE cannot be used to generate keys for a non-global zone. For more information on zones, see Chapter 16, "Introduction to Solaris Zones," in *System Administration Guide: Virtualization Using the Solaris Operating System*.
- IPsec policy can work with the Streams Control Transmission Protocol (SCTP) and SCTP port number. However, the implementation is not complete. The IPsec extensions for SCTP that are specified in RFC 3554 are not yet implemented. These limitations can cause complications when creating IPsec policy for SCTP. For details, consult the RFCs. Also, read "IPsec and SCTP" on page 491 and "SCTP Protocol" on page 42.
- IPsec and IKE can protect traffic that originates behind a NAT box. For details and limitations, see "IPsec and NAT Traversal" on page 490. For procedures, see "Configuring IKE for Mobile Systems (Task Map)" on page 583.

◆ ◆ ◆ CHAPTER 20

# Configuring IPsec (Tasks)

This chapter provides procedures for implementing IPsec on your network. The procedures are described in the following task maps:

- "Protecting Traffic With IPsec (Task Map)" on page 495
- "Protecting a VPN With IPsec (Task Map)" on page 511

For overview information about IPsec, see Chapter 19, "IP Security Architecture (Overview)." For reference information about IPsec, see Chapter 21, "IP Security Architecture (Reference)."

## Protecting Traffic With IPsec (Task Map)

The following task map points to procedures that set up IPsec between one or more systems. The `ipsecconf`(1M), `ipseckey`(1M), and `ifconfig`(1M) man pages also describe useful procedures in their respective Examples sections.

| Task | Description | For Instructions |
|---|---|---|
| Secure traffic between two systems. | Protects packets from one system to another system. | "How to Secure Traffic Between Two Systems With IPsec" on page 496 |
| Secure a web server by using IPsec policy. | Requires non-web traffic to use IPsec. Web clients are identified by particular ports, which bypass IPsec checks. | "How to Secure a Web Server With IPsec" on page 499 |
| Display IPsec policies. | Displays the IPsec policies that are currently being enforced, in the order in which the policies are enforced. | "How to Display IPsec Policies" on page 501 |
| Generate random numbers. | Generates random numbers for keying material for manually created security associations. | "How to Generate Random Numbers on a Solaris System" on page 502 |

| Task | Description | For Instructions |
|---|---|---|
| Create or replace security associations manually. | Provides the raw data for security associations:<br>■ IPsec algorithm name and keying material<br>■ Key for the security parameter index<br>■ IP source and destination addresses | "How to Manually Create IPsec Security Associations" on page 503 |
| Check that IPsec is protecting the packets. | Examines `snoop` output for specific headers that indicate how the IP datagrams are protected. | "How to Verify That Packets Are Protected With IPsec" on page 507 |
| (Optional) Create a Network Security role. | Creates a role that can set up a secure network, but has fewer powers than superuser. | "How to Create a Role for Configuring Network Security" on page 508 |
| Set up a secure virtual private network (VPN). | Sets up IPsec between two systems that are separated by the Internet. | "Protecting a VPN With IPsec (Task Map)" on page 511 |

# Protecting Traffic With IPsec

This section provides procedures that enable you to secure traffic between two systems and to secure a web server. To protect a VPN, see "Protecting a VPN With IPsec (Task Map)" on page 511. Additional procedures provide keying material and security associations, and verify that IPsec is working as configured.

The following information applies to all IPsec configuration tasks:

- **IPsec and zones –** To manage IPsec policy and keys for a non-global zone, create the IPsec policy file in the global zone, and run the IPsec configuration commands from the global zone. Use the source address that corresponds to the non-global zone that is being configured. You can also configure IPsec policy and keys in the global zone for the global zone. In the Solaris 10 8/07 you can use IKE to manage keys in a non-global zone.
- **IPsec and RBAC –** To use roles to administer IPsec, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *System Administration Guide: Security Services*. For an example, see "How to Create a Role for Configuring Network Security" on page 508.
- **IPsec and SCTP –** IPsec can be used to protect Streams Control Transmission Protocol (SCTP) associations, but caution must be used. For more information, see "IPsec and SCTP" on page 491.

## ▼ How to Secure Traffic Between Two Systems With IPsec

This procedure assumes the following setup:

- The two systems are named `enigma` and `partym`.
- Each system has two addresses, an IPv4 address and an IPv6 address.
- Each system invokes AH protection with the MD5 algorithm, which requires a key of 128 bits.

- Each system invokes ESP protections with the 3DES algorithm, which requires a key of 192 bits.
- Each system uses shared security associations.

  With shared SAs, only one pair of SAs is needed to protect the two systems.

**Before You Begin** You must be in the global zone to configure IPsec policy.

**1 On the system console, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

---

**2 On each system, add host entries.**

On a system that is running a release prior to the Solaris 10 8/07 release, add IPv4 and IPv6 entries to the `/etc/inet/ipnodes` file. The entries for one system must be contiguous in the file. For more information about system configuration files, see "TCP/IP Configuration Files" on page 235 and Chapter 11, "IPv6 in Depth (Reference)."

If you are connecting systems with IPv4 addresses only, you modify the `/etc/inet/hosts` file. In this example, the connecting systems are running an earlier Solaris release and are using IPv6 addresses.

**a. On a system that is named `enigma`, type the following in the `ipnodes` file:**

```
# Secure communication with partym
192.168.13.213 partym
2001::eeee:3333:3333 partym
```

**b. On a system that is named `partym`, type the following in the `ipnodes` file:**

```
# Secure communication with enigma
192.168.116.16 enigma
2001::aaaa:6666:6666 enigma
```

This step enables the boot scripts to use the system names without depending on nonexistent naming services.

**3 On each system, create the IPsec policy file.**

The file name is `/etc/inet/ipsecinit.conf`. For an example, see the `/etc/inet/ipsecinit.sample` file.

4 **Add an IPsec policy entry to the `ipsecinit.conf` file.**

a. **On the `enigma` system, add the following policy:**

```
{laddr enigma raddr partym} ipsec {auth_algs any encr_algs any sa shared}
```

b. **On the `partym` system, add the identical policy:**

```
{laddr partym raddr enigma} ipsec {auth_algs any encr_algs any sa shared}
```

For the syntax of IPsec policy entries, see the `ipsecconf`(1M) man page.

5 **On each system, add a pair of IPsec SAs between the two systems.**

You can configure Internet Key Exchange (IKE) to create the SAs automatically. You can also add the SAs manually.

---

**Note –** You should use IKE unless you have good reason to generate and maintain your keys manually. IKE key management is more secure than manual key management.

---

- Configure IKE by following one of the configuration procedures in "Configuring IKE (Task Map)" on page 553. For the syntax of the IKE configuration file, see the `ike.config`(4) man page.
- To add the SAs manually, see "How to Manually Create IPsec Security Associations" on page 503.

6 **Reboot each system.**

```
# init 6
```

7 **Verify that packets are being protected.**

For the procedure, see "How to Verify That Packets Are Protected With IPsec" on page 507.

**Example 20–1** Securing Traffic With IPsec Without Rebooting

The following example describes how to implement IPsec in a test environment. In a production environment, it is more secure to reboot than to run the `ipsecconf` command. For the security considerations, see the end of this example.

Instead of rebooting at Step 6, choose one of the following options:

- If you used IKE to create keying material, stop and then restart the `in.iked` daemon.

```
# pkill in.iked
# /usr/lib/inet/in.iked
```

- If you added keys manually, use the `ipseckey` command to add the SAs to the database.

```
# ipseckey -c -f /etc/inet/secret/ipseckeys
```

  Then, activate the IPsec policy with the `ipsecconf` command.

```
# ipsecconf -a /etc/inet/ipsecinit.conf
```

**Security Considerations** – Read the warning when you execute the `ipsecconf` command. A socket that is already latched, that is, a socket that is already in use, provides an unsecured back door into the system. For more extensive discussion, see "Security Considerations for `ipsecinit.conf` and `ipsecconf`" on page 539.

## How to Secure a Web Server With IPsec

A secure web server allows web clients to talk to the web service. On a secure web server, traffic that is not web traffic *must* pass security checks. The following procedure includes bypasses for web traffic. In addition, this web server can make unsecured DNS client requests. All other traffic requires ESP with AES and SHA-1 algorithms.

**Before You Begin** You must be in the global zone to configure IPsec policy.

**1 On the system console, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note** – Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

---

**2 Determine which services need to bypass security policy checks.**

For a web server, these services include TCP ports 80 (HTTP) and 443 (Secure HTTP). If the web server provides DNS name lookups, the server might also need to include port 53 for both TCP and UDP.

3 **Create a file in the `/etc/inet` directory for the web server policy.**

Give the file a name that indicates its purpose, for example `IPsecWebInitFile`. Type the following lines in this file:

```
# Web traffic that web server should bypass.
{lport  80 ulp tcp dir both} bypass {}
{lport 443 ulp tcp dir both} bypass {}

# Outbound DNS lookups should also be bypassed.
{rport 53 dir both} bypass {}

# Require all other traffic to use ESP with AES and SHA-1.
# Use a unique SA for outbound traffic from the port
{} ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

This configuration allows only secure traffic to access the system, with the bypass exceptions that are described in Step 2.

4 **Copy the contents of the file that you created in Step 3 into the `/etc/inet/ipsecinit.conf` file.**

5 **Protect the `IPsecWebInitFile` file with read-only permissions.**

```
# chmod 400 IPsecWebInitFile
```

6 **Secure the web server without rebooting.**

Choose one of the following options:

- If you are using IKE for key management, stop and restart the `in.iked` daemon.

  ```
  # pkill in.iked
  # /usr/lib/inet/in.iked
  ```

- If you are manually managing keys, use the `ipseckey` and `ipsecconf` commands.

  Use the `IPsecWebInitFile` as the argument to the `ipsecconf` command. If you use the `ipsecinit.conf` file as the argument, the `ipsecconf` command generates errors when policies in the file are already implemented on the system.

  ```
  # ipseckey -c -f /etc/inet/secret/ipseckeys
  # ipsecconf -a /etc/inet/IPsecWebInitFile
  ```

**Caution –** Read the warning when you execute the `ipsecconf` command. A socket that is already latched, that is, a socket that is already in use, provides an unsecured back door into the system. For more extensive discussion, see "Security Considerations for `ipsecinit.conf` and `ipsecconf`" on page 539. The same warning applies to restarting the `in.iked` daemon.

You can also reboot. Rebooting ensures that the IPsec policy is in effect on all TCP connections. At reboot, the TCP connections use the policy in the IPsec policy file.

7 **(Optional) Enable a remote system to communicate with the web server for nonweb traffic.**

Type the following policy in a remote system's `ipsecinit.conf` file:

```
# Communicate with web server about nonweb stuff
#
{laddr webserver} ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

A remote system can communicate securely with the web server for nonweb traffic only when the systems' IPsec policies match.

## How to Display IPsec Policies

You can see the policies that are configured in the system when you issue the `ipsecconf` command without any arguments.

**Before You Begin** You must run the `ipsecconf` command in the global zone.

1 **Assume a role that includes the Network Security profile, or become superuser.**

To create a role that includes the Network Security profile and assign that role to a user, see "How to Create a Role for Configuring Network Security" on page 508.

2 **Assume a role that includes the Network IPsec Management profile, or become superuser.**

To create a role that includes a network security profile and assign that role to a user, see "How to Create a Role for Configuring Network Security" on page 508.

3 **Display IPsec policies.**

a. **Display the global IPsec policy entries in the order that the entries were added.**

```
$ ipsecconf
```

The command displays each entry with an *index* followed by a number.

b. **Display the IPsec policy entries in the order in which a match occurs.**

```
$ ipsecconf -l
```

c. **Display the IPsec policy entries, including per-tunnel entries, in the order in which a match occurs.**

```
$ ipsecconf -L
```

## ▼ How to Generate Random Numbers on a Solaris System

If you are specifying keys manually, the keying material must be random. The format for keying material for a Solaris system is hexadecimal. Other operating systems can require ASCII keying material. To generate keying material for a Solaris system that is communicating with an operating system that requires ASCII, see Example 23–1.

If your site has a random number generator, use that generator. Otherwise, you can use the `od` command with the `/dev/random` Solaris device as input. For more information, see the `od`(1) man page.

**1 Generate random numbers in hexadecimal format.**

`% od -x|-X -A n` *file* `| head -`*n*

| | |
|---|---|
| `-x` | Displays the octal dump in hexadecimal format. Hexadecimal format is useful for keying material. The hexadecimal is printed in 4-character chunks. |
| `-X` | Displays the octal dump in hexadecimal format. The hexadecimal is printed in 8-character chunks. |
| `-A` n | Removes the input offset base from the display. |
| *file* | Serves as a source for random numbers. |
| `head -`*n* | Restricts the display to the first *n* lines of output. |

**2 Combine the output to create a key of the appropriate length.**

Remove the spaces between the numbers on one line to create a 32-character key. A 32-character key is 128 bits. For a security parameter index (SPI), you should use an 8-character key. The key should use the `0x` prefix.

**Example 20–2** Generating Key Material for IPsec

The following example displays two lines of keys in groups of eight hexadecimal characters each.

```
% od -X -A n /dev/random | head -2
         d54d1536 4a3e0352 0faf93bd 24fd6cad
         8ecc2670 f3447465 20db0b0c c83f5a4b
```

By combining the four numbers on the first line, you can create a 32-character key. An 8-character number that is preceded by `0x` provides a suitable SPI value, for example, `0xf3447465`.

The following example displays two lines of keys in groups of four hexadecimal characters each.

```
% od -x -A n /dev/random | head -2
         34ce 56b2 8b1b 3677 9231 42e9 80b0 c673
         2f74 2817 8026 df68 12f4 905a db3d ef27
```

By combining the eight numbers on the first line, you can create a 32-character key.

## How to Manually Create IPsec Security Associations

The following procedure provides the keying material for the procedure, "How to Secure Traffic Between Two Systems With IPsec" on page 496.

**Before You Begin** You must be in the global zone to manually manage keying material for a non-global zone.

**1 Generate the keying material for the SAs.**

You need three hexadecimal random numbers for outbound traffic and three hexadecimal random numbers for inbound traffic.

Therefore, one system needs to generate the following numbers:

- Two hexadecimal random numbers as the value for the `spi` keyword. One number is for outbound traffic. One number is for inbound traffic. Each number can be up to eight characters long.
- Two hexadecimal random numbers for the MD5 algorithm for AH. Each number must be 32 characters long. One number is for `dst enigma`. One number is for `dst partym`.
- Two hexadecimal random numbers for the 3DES algorithm for ESP. For a 192-bit key, each number must be 48 characters long. One number is for `dst enigma`. One number is for `dst partym`.

If you have a random number generator at your site, use the generator. You can also use the `od` command. See "How to Generate Random Numbers on a Solaris System" on page 502 for the procedure.

**2 On the system console on one of the systems, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

---

**3 Enable the `ipseckey` command mode.**

```
# ipseckey

>
```

The > prompt indicates that you are in `ipseckey` command mode.

**4 If you are replacing existing SAs, flush the current SAs.**

```
> flush
>
```

To prevent an adversary from having time to break your SAs, you need to replace the keying material.

---

**Note –** You must coordinate key replacement on communicating systems. When you replace the SAs on one system, the SAs must also be replaced on the remote system.

---

**5 To create SAs, type the following command.**

```
> add protocol spi random-hex-string \
src addr dst addr2 \
protocol-prefix_alg protocol-algorithm  \
protocol-prefixkey random-hex-string-of-algorithm-specified-length
```

You also use this syntax to replace SAs that you have just flushed.

*protocol*
Specifies either `esp` or `ah`.

*random-hex-string*
Specifies a random number of up to eight characters in hexadecimal format. Precede the characters with `0x`. If you enter more numbers than the security parameter index (SPI) accepts, the system ignores the extra numbers. If you enter fewer numbers than the SPI accepts, the system pads your entry.

*addr*
Specifies the IP address of one system.

*addr2*
Specifies the IP address of the peer system of *addr*.

*protocol-prefix*
Specifies one of `encr` or `auth`. The `encr` prefix is used with the `esp` protocol. The `auth` prefix is used with the `ah` protocol, and for authenticating the `esp` protocol.

*protocol-algorithm*
Specifies an algorithm for ESP or AH. Each algorithm requires a key of a specific length.

Authentication algorithms include MD5 and SHA. Encryption algorithms include 3DES and AES.

*random-hex-string-of-algorithm-specified-length*
Specifies a random hexadecimal number of the length that is required by the algorithm. For example, the MD5 algorithm requires a 32-character string for its 128-bit key. The 3DES algorithm requires a 48-character string for its 192-bit key.

**a. For example, on the `enigma` system, protect outbound packets.**

Use the random numbers that you generated in Step 1.

For Solaris 10 1/06:

```
> add esp spi 0x8bcd1407 \
src 192.168.116.16 dst 192.168.13.213 \
encr_alg 3des \
auth_alg md5 \
encrkey d41fb74470271826a8e7a80d343cc5aae9e2a7f05f13730d \
authkey e896f8df7f78d6cab36c94ccf293f031
>
```

**Note –** The peer system must use the same keying material and the same SPI.

**b. Still in `ipseckey` command mode on the `enigma` system, protect inbound packets.**

Type the following commands to protect the packets:

```
> add esp spi 0x122a43e4 \
src 192.168.13.213 dst 192.168.116.16 \
encr_alg 3des \
auth_alg md5 \
encrkey dd325c5c137fb4739a55c9b3a1747baa06359826a5e4358e \
authkey ad9ced7ad5f255c9a8605fba5eb4d2fd
>
```

**Note –** The keys and SPI can be different for each SA. You *should* assign different keys and a different SPI for each SA.

**6 To exit `ipseckey` command mode, press Control-D or type `quit`.**

**7 To ensure that the keying material is available to IPsec at reboot, add the keying material to the `/etc/inet/secret/ipseckeys` file.**

The lines of the `/etc/inet/secret/ipseckeys` file are identical to the command line language.

**a. For example, the `/etc/inet/secret/ipseckeys` file on the `enigma` system would appear similar to the following:**

```
# ipseckeys - This file takes the file format documented in
#   ipseckey(1m).
#   Note that naming services might not be available when this file
#   loads, just like ipsecinit.conf.
#
# for outbound packets on enigma
add esp spi 0x8bcd1407 \
   src 192.168.116.16 dst 192.168.13.213  \
   encr_alg 3des \
   auth_alg md5  \
   encrkey  d41fb74470271826a8e7a80d343cc5aae9e2a7f05f13730d \
   authkey  e896f8df7f78d6cab36c94ccf293f031
#
# for inbound packets
add esp spi 0x122a43e4 \
   src 192.168.13.213 dst 192.168.116.16 \
   encr_alg 3des \
   auth_alg md5  \
   encrkey dd325c5c137fb4739a55c9b3a1747baa06359826a5e4358e \
   authkey ad9ced7ad5f255c9a8605fba5eb4d2fd
```

**b. Protect the file with read-only permissions.**

```
# chmod 400 /etc/inet/secret/ipseckeys
```

**8 Repeat Step 2 through Step 7 on the `partym` system.**

Use the same keying material that was used on `enigma`.

The keying material on the two systems *must* be identical. As shown in the following example, only the comments in the `ipseckeys` file differ. The comments differ because `dst enigma` is inbound on the `enigma` system, and outbound on the `partym` system.

```
# partym ipseckeys file
#
# for inbound packets
add esp spi 0x8bcd1407 \
   src 192.168.116.16 dst 192.168.13.213  \
   encr_alg 3des \
   auth_alg md5  \
   encrkey  d41fb74470271826a8e7a80d343cc5aae9e2a7f05f13730d \
   authkey  e896f8df7f78d6cab36c94ccf293f031
#
```

```
# for outbound packets
add esp spi 0x122a43e4 \
   src 192.168.13.213 dst 192.168.116.16 \
   encr_alg 3des \
   auth_alg md5  \
   encrkey dd325c5c137fb4739a55c9b3a1747baa06359826a5e4358e \
   authkey ad9ced7ad5f255c9a8605fba5eb4d2fd
```

## ▼ How to Verify That Packets Are Protected With IPsec

To verify that packets are protected, test the connection with the `snoop` command. The following prefixes can appear in the `snoop` output:

- `AH:` Prefix indicates that AH is protecting the headers. You see `AH:` if you used `auth_alg` to protect the traffic.
- `ESP:` Prefix indicates that encrypted data is being sent. You see `ESP:` if you used `encr_auth_alg` or `encr_alg` to protect the traffic.

**Before You Begin** You must be superuser or have assumed an equivalent role to create the `snoop` output. You must have access to both systems to test the connection.

**1 On one system, such as `partym`, become superuser.**

```
% su -
Password:       Type root password
#
```

**2 From the `partym` system, prepare to snoop packets from a remote system.**

In a terminal window on `partym`, snoop the packets from the `enigma` system.

```
# snoop -v enigma
Using device /dev/hme (promiscuous mode)
```

**3 Send a packet from the remote system.**

In another terminal window, remotely log in to the `enigma` system. Provide your password. Then, become superuser and send a packet from the `enigma` system to the `partym` system. The packet should be captured by the `snoop -v enigma` command.

```
% ssh enigma
Password:       Type your password
% su -
Password:       Type root password
# ping partym
```

**4 Examine the `snoop` output.**

On the `partym` system, you should see output that includes `AH` and `ESP` information after the initial `IP` header information. `AH` and `ESP` information that resembles the following shows that packets are being protected:

```
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 51 (AH)
IP:   Header checksum = 4e0e
IP:   Source address = 192.168.116.16, enigma
IP:   Destination address = 192.168.13.213, partym
IP:   No options
IP:
AH:  ----- Authentication Header -----
AH:
AH:  Next header = 50 (ESP)
AH:  AH length = 4 (24 bytes)
AH:  <Reserved field = 0x0>
AH:  SPI = 0xb3a8d714
AH:  Replay = 52
AH:  ICV = c653901433ef5a7d77c76eaa
AH:
ESP:  ----- Encapsulating Security Payload -----
ESP:
ESP:  SPI = 0xd4f40a61
ESP:  Replay = 52
ESP:     ....ENCRYPTED DATA....

ETHER:  ----- Ether Header -----
...
```

# ▼ How to Create a Role for Configuring Network Security

If you are using role-based access control (RBAC) to administer your systems, you use this procedure to provide a network management or network security role.

**1 Find the Network rights profiles in the local `prof_attr` database.**

```
% cd /etc/security
% grep Network prof_attr
Network Management:::Manage the host and network configuration
Network Security:::Manage network and host security
System Administrator::: Network Management
```

The Network Management profile is a supplementary profile in the System Administrator profile. If you have included the System Administrator rights profile in a role, then that role can execute the commands in the Network Management profile.

2 **Determine which commands are in the Network Management rights profile.**

```
% grep "Network Management" /etc/security/exec_attr
Network Management:solaris:cmd:::/usr/sbin/ifconfig:privs=sys_net_config
...
Network Management:suser:cmd:::/usr/sbin/snoop:uid=0
```

The `solaris` policy commands run with privilege (`privs=sys_net_config`). The `suser` policy commands run as superuser (`uid=0`).

3 **Create a role that includes the Network Security and the Network Management rights profiles.**

A role with both profiles can execute the `ifconfig`, `snoop`, `ipsecconf`, and `ipseckey` commands, among others, with appropriate privilege.

To create the role, assign the role to a user, and register the changes with the name service, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

# Protecting a VPN With IPsec

IPsec tunnels can protect a VPN. In the Solaris 10 8/07 release, a tunnel can be in tunnel mode or transport mode. *Tunnel mode* is interoperable with the implementation of IPsec by other vendors. *Transport mode* is interoperable with earlier versions of the Solaris OS. For a discussion of tunnel modes, see "Transport and Tunnel Modes in IPsec" on page 487.

Tunnels in tunnel mode offer more fine-grained control of the traffic. In tunnel mode, for an inner IP address, you can specify the particular protection you want, down to a single port.

- For examples of IPsec policies for tunnels in tunnel mode, see "Examples of Protecting a VPN With IPsec by Using Tunnels in Tunnel Mode" on page 509.
- For the procedures that protect VPNs, see "Protecting a VPN With IPsec (Task Map)" on page 511.

## Examples of Protecting a VPN With IPsec by Using Tunnels in Tunnel Mode

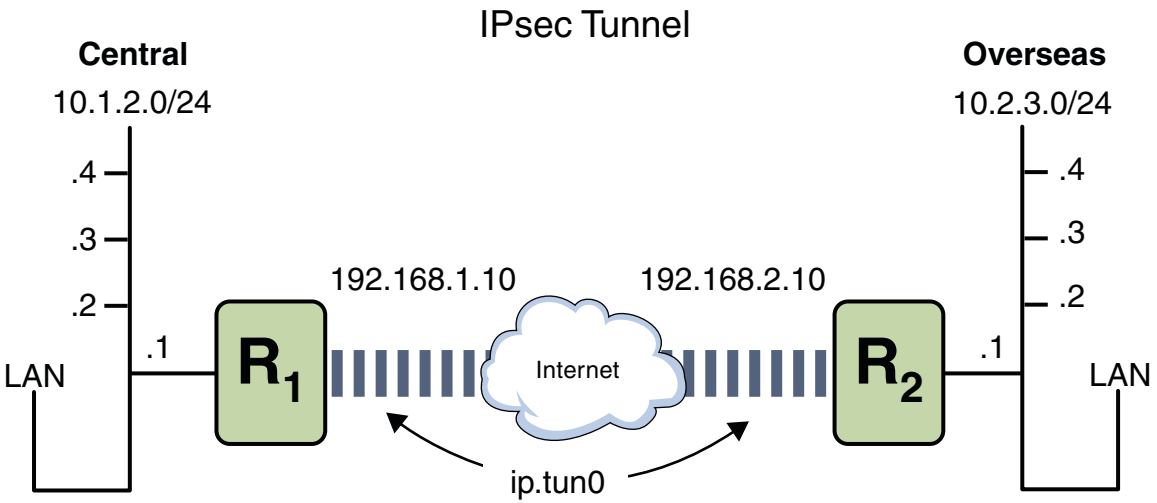



**FIGURE 20–1** IPsec Tunnel Diagram

The following examples assume that the tunnel is configured for all subnets of the LANs:

```
## Tunnel configuration ##
ifconfig ip.tun0 10.1.2.1 10.2.3.1 tsrc 192.168.1.10 tdst 192.168.2.10
```

**EXAMPLE 20–3** Creating a Tunnel That All Subnets Can Use

In this example, all traffic from the local LANs of the Central LAN in Figure 20–1 can be tunneled through Router 1 to Router 2, and then delivered to all local LANs of the Overseas LAN. The traffic is encrypted with AES.

```
## IPsec policy ##
{tunnel ip.tun0 negotiate tunnel}
 ipsec {encr_algs aes encr_auth_algs md5 sa shared}
```

**EXAMPLE 20–4** Creating a Tunnel That Connects Two Subnets Only

In this example, only traffic between subnet `10.1.2.0/24` of the Central LAN and subnet `10.2.3.0/24` of the Overseas LAN is tunneled and encrypted. In the absence of other IPsec policies for Central, if the Central LAN attempts to route any traffic for other LANs over this tunnel, the traffic is dropped at Router 1.

```
## IPsec policy ##
{tunnel ip.tun0 negotiate tunnel laddr 10.1.2.0/24 raddr 10.2.3.0/24}
 ipsec {encr_algs aes encr_auth_algs md5 sa shared}
```

**EXAMPLE 20–5** Creating a Tunnel for Email Traffic Only Between Two Subnets

In this example, a tunnel is created for email traffic only. The traffic is delivered from subnet `10.1.2.0/24` of the Central LAN to the email server on the `10.2.3.0/24` subnet of the Overseas LAN. The email is encrypted with Blowfish. The policies apply to the remote and local email ports. The `rport` policy protects email that Central sends to the remote email port of Overseas. The `lport` policy protects email that Central receives from Overseas on local port 25.

```
## IPsec policy for email from Central to Overseas ##
{tunnel ip.tun0 negotiate tunnel ulp tcp rport 25
 laddr 10.1.2.0/24 raddr 10.2.3.0/24}
 ipsec {encr_algs blowfish encr_auth_algs md5 sa shared}

## IPsec policy for email from Overseas to Central ##
{tunnel ip.tun0 negotiate tunnel ulp tcp lport 25
 laddr 10.1.2.0/24 raddr 10.2.3.0/24}
 ipsec {encr_algs blowfish encr_auth_algs md5 sa shared}
```

**EXAMPLE 20–6** Creating a Tunnel for FTP Traffic for All Subnets

In this example, IPsec policy protects the FTP ports in Figure 20–1 with 3DES for all subnets of the Central LAN to all subnets of the Overseas LAN. This configuration works for the active mode of FTP.

```
## IPsec policy for outbound FTP from Central to Overseas ##
{tunnel ip.tun0 negotiate tunnel ulp tcp rport 21}
  ipsec {encr_algs 3des encr_auth_algs md5 sa shared}
{tunnel ip.tun0 negotiate tunnel ulp tcp lport 20}
  ipsec {encr_algs 3des encr_auth_algs md5 sa shared}

## IPsec policy for inbound FTP from Central to Overseas ##
{tunnel ip.tun0 negotiate tunnel ulp tcp lport 21}
  ipsec {encr_algs 3des encr_auth_algs md5 sa shared}
{tunnel ip.tun0 negotiate tunnel ulp tcp rport 20}
  ipsec {encr_algs 3des encr_auth_algs md5 sa shared}
```

# Protecting a VPN With IPsec (Task Map)

The following task map points to procedures that configure IPsec to protect traffic across the Internet. These procedures set up a secure virtual private network (VPN) between two systems that are separated by the Internet. One common use of this technology is to protect traffic between home workers and their corporate office.

| Task | Description | For Instructions |
|---|---|---|
| Protect tunnel traffic in tunnel mode over IPv4. | Protects traffic in tunnel mode between two Solaris 10 8/07 systems, two Solaris Express systems, or between a Solaris 10 8/07 system and a Solaris Express system.<br>Also, protects traffic in tunnel mode between a Solaris 10 8/07 system or a Solaris Express system, and a system that is running on another platform. | "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode Over IPv4" on page 514 |
| Protect tunnel traffic in tunnel mode over IPv6. | Protects traffic in tunnel mode between two Solaris systems that are using the IPv6 protocol. | "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode Over IPv6" on page 520 |
| Protect tunnel traffic in transport mode over IPv4. | Protects traffic in transport mode between two Solaris 10 8/07 systems, two Solaris Express systems, or between a Solaris 10 8/07 system and a Solaris Express system.<br>Also, protects traffic in transport mode between a system that is running an earlier version of the Solaris OS and a Solaris 10 8/07 or a Solaris Express system. | "How to Protect a VPN With an IPsec Tunnel in Transport Mode Over IPv4" on page 524 |
| | Protects traffic by using an older, deprecated syntax. This method is useful when you are communicating with a system that is running an earlier version of the Solaris OS. This method simplifies comparing the configuration files on the two systems. | Example 20–11 |
| Protect tunnel traffic in transport mode over IPv6. | Protects traffic in transport mode between two Solaris systems that are using the IPv6 protocol. | "How to Protect a VPN With an IPsec Tunnel in Transport Mode Over IPv6" on page 530 |

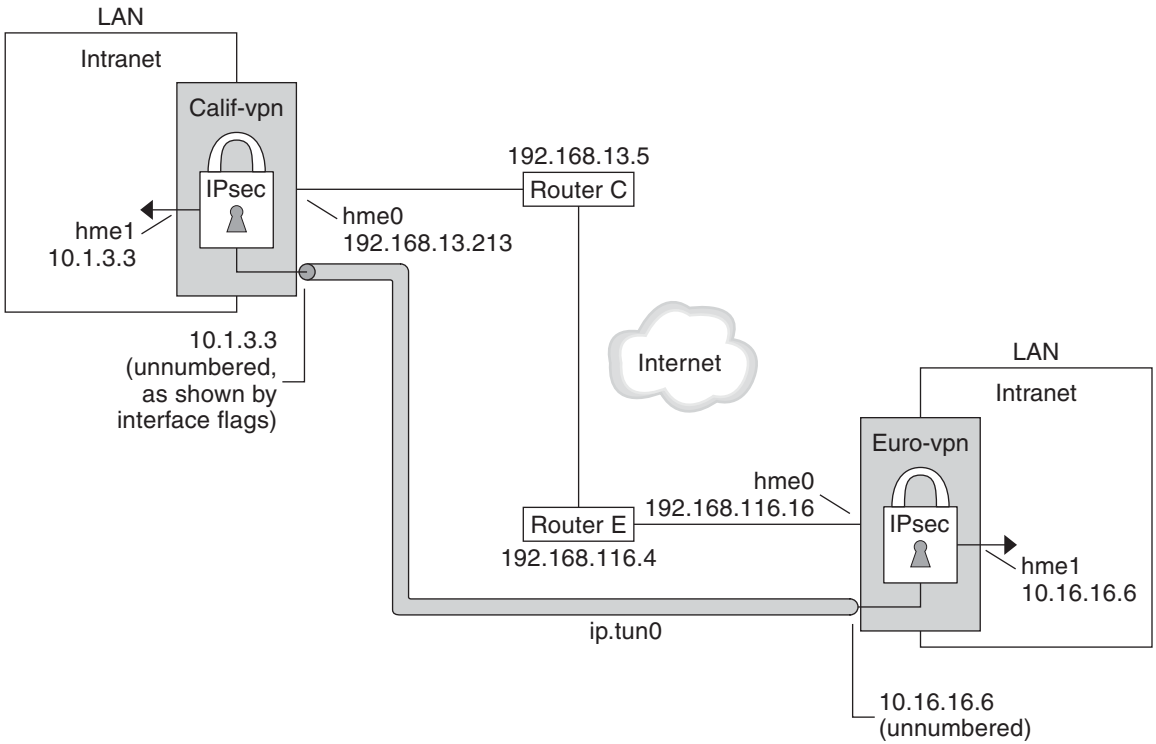
# Description of the Network Topology for the IPsec Tasks to Protect a VPN

The procedures that follow this section assume the following setup. For a depiction of the network, see Figure 20–2.

- Each system is using an IPv4 address space.

  For a similar example with IPv6 addresses, see "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode Over IPv6" on page 520.
- Each system has two interfaces. The `hme0` interface connects to the Internet. In this example, Internet IP addresses begin with `192.168`. The `hme1` interface connects to the company's LAN, its intranet. In this example, intranet IP addresses begin with the number `10`.
- Each system invokes AH protection with the MD5 algorithm. The MD5 algorithm requires a 128-bit key.
- Each system invokes ESP protection with the 3DES algorithm. The 3DES algorithm requires a 192-bit key.
- Each system can connect to a router that has direct access to the Internet.

- Each system uses shared security associations.




hme0 = Turn off IP forwarding

hme1 = Turn on IP forwarding

ip.tun0 = Turn on IP forwarding

Router C - /etc/defaultrouter for Calif-vpn

Router E - /etc/defaultrouter for Euro-vpn

**FIGURE 20–2** Sample VPN Between Offices Separated by the Internet

As the preceding illustration shows, the IPv4 procedures use these configuration parameters.

| Parameter | Europe | California |
|---|---|---|
| System name | `enigma` | `partym` |
| System intranet interface | `hme1` | `hme1` |
| System Internet interface | `hme0` | `hme0` |

| Parameter | Europe | California |
|---|---|---|
| System intranet address, also the *-point* address in Step 5 | `10.16.16.6` | `10.1.3.3` |
| System Internet address, also the *tsrc* address in Step 5 | `192.168.116.16` | `192.168.13.213` |
| Name of Internet router | `router-E` | `router-C` |
| Address of Internet router | `192.168.116.4` | `192.168.13.5` |
| Tunnel name | `ip.tun0` | `ip.tun0` |

# ▼ How to Protect a VPN With an IPsec Tunnel in Tunnel Mode Over IPv4

This procedure extends the procedure "How to Secure Traffic Between Two Systems With IPsec" on page 496. The setup is described in "Description of the Network Topology for the IPsec Tasks to Protect a VPN" on page 512. In addition to connecting two systems, you are connecting two intranets that connect to these two systems. The systems in this procedure function as gateways.

**Before You Begin** You must be in the global zone to configure IPsec policy.

**1 On the system console on one of the systems, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

---

**2 Control the flow of packets before configuring IPsec.**

**a. Ensure that IP forwarding and IP dynamic routing are disabled.**

```
# routeadm
Configuration       Current         Current
       Option       Configuration  System State
---------------------------------------------------
IPv4 forwarding     disabled           disabled
   IPv4 routing     default (enabled)   enabled
...
```

If IP forwarding and IP dynamic routing are enabled, you can disable them by typing:

```
# routeadm -d ipv4-routing -d ipv4-forwarding
# routeadm -u
```

Turning off IP forwarding prevents packets from being forwarded from one network to another network through this system. For a description of the `routeadm` command, see the `routeadm`(1M) man page.

**b. Turn on IP strict destination multihoming.**

```
# ndd -set /dev/ip ip_strict_dst_multihoming 1
```

Turning on IP strict destination multihoming ensures that packets for one of the system's destination addresses arrive at the correct destination address.

When strict destination multihoming is enabled, packets that arrive on a particular interface must be addressed to one of the local IP addresses of that interface. All other packets, even packets that are addressed to other local addresses of the system, are dropped.

**c. Disable most network services, and possibly all network services.**

---

**Note –** If your system was installed with the "limited" SMF profile, then you can skip this step. Network services, with the exception of Solaris Secure Shell, are disabled.

---

The disabling of network services prevents IP packets from doing any harm to the system. For example, an SNMP daemon, a `telnet` connection, or an `rlogin` connection could be exploited.

Choose one of the following options:

- If you are running the Solaris 10 11/06 release or a later release, run the "limited" SMF profile.

```
# netservices limited
```

- Otherwise, individually disable network services.

```
# svcadm disable network/ftp:default
# svcadm disable network/finger:default
# svcadm disable network/login:rlogin
# svcadm disable network/nfs/server:default
# svcadm disable network/rpc/rstat:default
# svcadm disable network/smtp:sendmail
# svcadm disable network/telnet:default
# svcs | grep network
online         Aug_02   svc:/network/loopback:default

online         Aug_09   svc:/network/ssh:default
```

**3 On each system, add a pair of SAs between the two systems.**

Choose one of the following options:

- Configure IKE to manage the keys for the SAs. Use one of the procedures in "Configuring IKE (Task Map)" on page 553 to configure IKE for the VPN.
- If you have an overriding reason to manually manage the keys, see "How to Manually Create IPsec Security Associations" on page 503.

**4 On each system, add IPsec policy.**

Edit the `/etc/inet/ipsecinit.conf` file to add the IPsec policy for the VPN. To strengthen the policy, see Example 20–7. For additional examples, see "Examples of Protecting a VPN With IPsec by Using Tunnels in Tunnel Mode" on page 509.

In this policy, IPsec protection is not required between systems on the local LAN and the internal IP address of the gateway, so a `bypass` statement is added.

**a. For example, on the `enigma` system, type the following entry into the `ipsecinit.conf` file:**

```
# LAN traffic to and from this host can bypass IPsec.
{laddr 10.16.16.6 dir both} bypass {}

# WAN traffic uses ESP with 3DES and MD5.
{tunnel ip.tun0 negotiate tunnel}
 ipsec {encr_algs 3des encr_auth_algs md5 sa shared}
```

**b. On the `partym` system, type the following entry into the `ipsecinit.conf` file:**

```
# LAN traffic to and from this host can bypass IPsec.
{laddr 10.1.3.3 dir both} bypass {}

# WAN traffic uses ESP with 3DES and MD5.
{tunnel ip.tun0 negotiate tunnel}
 ipsec {encr_algs 3des encr_auth_algs md5 sa shared}
```

**5 On each system, configure the tunnel, `ip.tun0`.**

Use the following `ifconfig` commands to create the point-to-point interface:

```
# ifconfig ip.tun0 plumb

# ifconfig ip.tun0 system1-point system2-point \
tsrc system1-taddr tdst system2-taddr
```

**a. For example, on the `enigma` system, type the following commands:**

```
# ifconfig ip.tun0 plumb

# ifconfig ip.tun0 10.16.16.6 10.1.3.3 \
tsrc 192.168.116.16 tdst 192.168.13.213
```

**b. On the `partym` system, type the following commands:**

```
# ifconfig ip.tun0 plumb

# ifconfig ip.tun0 10.1.3.3 10.16.16.6  \
tsrc 192.168.13.213 tdst 192.168.116.16
```

**6 Protect the tunnel with the IPsec policy that you created.**

```
# ipsecconf
```

**7 Bring up the router for the tunnel.**

```
# ifconfig ip.tun0 router up
```

**8 On each system, turn on IP forwarding for the `hme1` interface.**

```
# ifconfig hme1 router
```

IP forwarding means that packets that arrive from somewhere else can be forwarded. IP forwarding also means that packets that leave this interface might have originated somewhere else. To successfully forward a packet, both the receiving interface and the transmitting interface must have IP forwarding turned on.

Because the `hme1` interface is *inside* the intranet, IP forwarding must be turned on for `hme1`. Because `ip.tun0` connects the two systems through the Internet, IP forwarding must be turned on for `ip.tun0`.

The `hme0` interface has its IP forwarding turned off to prevent an *outside* adversary from injecting packets into the protected intranet. The *outside* refers to the Internet.

**9 On each system, ensure that routing protocols do not advertise the default route within the intranet.**

```
# ifconfig hme0 private
```

Even if `hme0` has IP forwarding turned off, a routing protocol implementation might still advertise the interface. For example, the `in.routed` protocol might still advertise that `hme0` is available to forward packets to its peers inside the intranet. By setting the interface's *private* flag, these advertisements are prevented.

**10 On each system, manually add a default route over `hme0`.**

The default route must be a router with direct access to the Internet.

**a. For example, on the `enigma` system, add the following route:**

```
# route add default 192.168.116.4
```

**b. On the `partym` system, add the following route:**

```
# route add default 192.168.13.5
```

Even though the `hme0` interface is not part of the intranet, `hme0` does need to reach across the Internet to its peer system. To find its peer, `hme0` needs information about Internet routing. The VPN system appears to be a host, rather than a router, to the rest of the Internet. Therefore, you can use a default router or run the router discovery protocol to find a peer system. For more information, see the `route`(1M) and `in.routed`(1M) man pages.

**11 On each system, ensure that the VPN starts after a reboot by adding an entry to the `/etc/hostname.ip.tun0` file.**

*system1-point system2-point* `tsrc` *system1-taddr* `tdst` *system2-taddr* `router up`

**a. For example, on the `enigma` system, add the following entry to the `hostname.ip.tun0` file:**

```
10.16.16.6 10.1.3.3 tsrc 192.168.116.16 tdst 192.168.13.213 router up
```

**b. On the `partym` system, add the following entry to the `hostname.ip.tun0` file:**

```
10.1.3.3 10.16.16.6 tsrc 192.168.13.213 tdst 192.168.116.16 router up
```

**12 On each system, configure the interface files to pass the correct parameters to the routing daemon.**

**a. On the `enigma` system, modify the `/etc/hostname.`*interface* files.**

```
# cat enigma hostname.hme0
10.16.16.6 private

# cat enigma hostname.hme1
192.168.116.16 router
```

**b. On the `partym` system, modify the `/etc/hostname.`*interface* files.**

```
# cat partym hostname.hme0
10.1.3.3 private

# cat partym hostname.hme1
192.168.13.213 router
```

**13 On each system, run a routing protocol.**

```
# routeadm -e ipv4-routing
# routeadm -u
```

You might need to configure the routing protocol before running the routing protocol. For more information, see "Routing Protocols in the Solaris OS" on page 253. For a procedure, see "How to Configure an IPv4 Router" on page 116.

### Example 20–7 Requiring IPsec Policy on All Systems on a LAN

In this example, the administrator comments out the `bypass` policy that was configured in Step 4, thereby strengthening the protection. With this policy configuration, each system on the LAN must activate IPsec to communicate with the router.

```
# LAN traffic must implement IPsec.
# {laddr 10.1.3.3 dir both} bypass {}

# WAN traffic uses ESP with 3DES and MD5.
{tunnel ip.tun0 negotiate tunnel} ipsec {encr_algs 3des encr_auth_algs md5}
```

### Example 20–8 Using IPsec to Protect Telnet Traffic Differently From SMTP Traffic

In this example, the first rule protects `telnet` traffic on port 23 with Blowfish and Sha-1. The second rule protects SMTP traffic on port 25 with AES and MD5.

```
{laddr 10.1.3.3 ulp tcp dport 23 dir both}
  ipsec {encr_algs blowfish encr_auth_algs sha1 sa unique}
{laddr 10.1.3.3 ulp tcp dport 25 dir both}
 ipsec {encr_algs aes encr_auth_algs md5 sa unique}
```

**Example 20–9** Using an IPsec Tunnel in Tunnel Mode to Protect a Subnet Differently From Other Network Traffic

The following tunnel configuration protects all traffic from subnet `10.1.3.0/24` across the tunnel:

```
{tunnel ip.tun0 negotiate tunnel laddr 10.1.3.0/24}
  ipsec {encr_algs aes encr_auth_algs md5 sa shared}
```

The following tunnel configurations protect traffic from subnet `10.1.3.0/24` to different subnets across the tunnel. Subnets that begin with `10.2.`*x.x* are across the tunnel.

```
{tunnel ip.tun0 negotiate tunnel laddr 10.1.3.0/24 raddr 10.2.1.0/24}
  ipsec {encr_algs blowfish encr_auth_algs md5 sa shared}

{tunnel ip.tun0 negotiate tunnel laddr 10.1.3.0/24 raddr 10.2.2.0/24}
  ipsec {encr_algs blowfish encr_auth_algs sha1 sa shared}

{tunnel ip.tun0 negotiate tunnel laddr 10.1.3.0/24 raddr 10.2.3.0/24}
  ipsec {encr_algs aes encr_auth_algs sha1 sa shared}
```

## ▼ How to Protect a VPN With an IPsec Tunnel in Tunnel Mode Over IPv6

To set up a VPN on an IPv6 network, you follow the same steps as for an IPv4 network. However, the syntax of the commands is slightly different. For a fuller description of the reasons for running particular commands, see "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode Over IPv4" on page 514.

This procedure uses the following configuration parameters.

| Parameter | Europe | California |
|---|---|---|
| System name | `enigma` | `partym` |
| System intranet interface | `hme1` | `hme1` |
| System Internet interface | `hme0` | `hme0` |
| System intranet address | `6000:6666::aaaa:1116` | `6000:3333::eeee:1113` |
| System Internet address | `2001::aaaa:6666:6666` | 2001::eeee:3333:3333 |
| Name of Internet router | `router-E` | `router-C` |
| Address of Internet router | `2001::aaaa:0:4` | `2001::eeee:0:1` |

| Parameter | Europe | California |
|---|---|---|
| Tunnel name | `ip6.tun0` | `ip6.tun0` |

**1 On the system console on one of the systems, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

**2 Control the flow of packets before configuring IPsec.**

For the effects of these commands, see Step 2 in "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode Over IPv4" on page 514.

**a. Ensure that IP forwarding and IP dynamic routing are disabled.**

```
# routeadm
Configuration       Current         Current
       Option       Configuration  System State
---------------------------------------------------
...
IPv6 forwarding     disabled          disabled
   IPv6 routing     disabled          disabled
```

If IP forwarding and IP dynamic routing are enabled, you can disable them by typing:

```
# routeadm -d ipv6-forwarding -d ipv6-routing
# routeadm -u
```

**b. Turn on IP strict destination multihoming.**

```
# ndd -set /dev/ip ip6_strict_dst_multihoming 1
```

**c. Disable most network services, and possibly all network services.**

**Note –** If your system was installed with the "limited" SMF profile, then you can skip this step. Network services, with the exception of Solaris Secure Shell, are disabled.

The disabling of network services prevents IP packets from doing any harm to the system. For example, an SNMP daemon, a `telnet` connection, or an `rlogin` connection could be exploited.

Choose one of the following options:

- If you are running the Solaris 10 11/06 release or a later release, run the "limited" SMF profile.

```
# netservices limited
```

- Otherwise, individually disable network services.

```
# svcadm disable network/ftp:default
# svcadm disable network/finger:default
# svcadm disable network/login:rlogin
# svcadm disable network/nfs/server:default
# svcadm disable network/rpc/rstat:default
# svcadm disable network/smtp:sendmail
# svcadm disable network/telnet:default
# svcs | grep network
online         Aug_02   svc:/network/loopback:default

online         Aug_09   svc:/network/ssh:default
```

**3 On each system, add a pair of SAs between the two systems.**

Choose one of the following options:

- Configure IKE to manage the keys for the SAs. Use one of the procedures in "Configuring IKE (Task Map)" on page 553 to configure IKE for the VPN.
- If you have an overriding reason to manually manage the keys, see "How to Manually Create IPsec Security Associations" on page 503.

**4 On each system, add IPsec policy.**

Edit the `/etc/inet/ipsecinit.conf` file to add the IPsec policy for the VPN.

**a. For example, onthe `enigma` system, type the following entry into the `ipsecinit.conf` file:**

```
# IPv6 Neighbor Discovery messages bypass IPsec.
{ulp ipv6-icmp type 133-137 dir both} pass {}

# LAN traffic to and from this host can bypass IPsec.
{laddr 6000:6666::aaaa:1116 dir both} bypass {}

# WAN traffic uses ESP with 3DES and MD5.
{tunnel ip6.tun0 negotiate tunnel}
  ipsec {encr_algs 3des encr_auth_algs md5 sa shared}
```

**b. On the `partym` system, type the following entry into the `ipsecinit.conf` file:**

```
# IPv6 Neighbor Discovery messages bypass IPsec.
{ulp ipv6-icmp type 133-137 dir both} pass {}
```

```
# LAN traffic to and from this host can bypass IPsec.
{laddr 6000:3333::eeee:1113 dir both} bypass {}

# WAN traffic uses ESP with 3DES and MD5.
{tunnel ip6.tun0 negotiate tunnel}
  ipsec {encr_algs 3des encr_auth_algs md5 sa shared}
```

5 **On each system, configure a secure tunnel,** `ip6.tun0`**.**

a. **For example, on the** `enigma` **system, type the following commands:**

```
# ifconfig ip6.tun0 inet6 plumb

# ifconfig ip6.tun0 inet6 6000:6666::aaaa:1116 6000:3333::eeee:1113 \
tsrc 2001::aaaa:6666:6666   tdst 2001::eeee:3333:3333
```

b. **On the** `partym` **system, type the following commands:**

```
# ifconfig ip6.tun0 inet6 plumb

# ifconfig ip6.tun0 inet6 6000:3333::eeee:1113 6000:6666::aaaa:1116 \
tsrc 2001::eeee:3333:3333   tdst 2001::aaaa:6666:6666
```

6 **Protect the tunnel with the IPsec policy that you created.**

```
# ipsecconf
```

7 **Bring up the router for the tunnel.**

```
# ifconfig ip6.tun0 router up
```

8 **On each system, turn on IP forwarding for the** `hme1` **interface.**

```
# ifconfig hme1 router
```

9 **On each system, ensure that routing protocols do not advertise the default route within the intranet.**

```
# ifconfig hme0 private
```

10 **On each system, manually add a default route over** `hme0`**.**

The default route must be a router with direct access to the Internet.

a. **For example, on the** `enigma` **system, add the following route:**

```
# route add -inet6 default 2001::aaaa:0:4
```

b. **On the** `partym` **system, add the following route:**

```
# route add -inet6 default 2001::eeee:0:1
```

**11 On each system, ensure that the VPN starts after a reboot by adding an entry to the `/etc/hostname6.ip6.tun0` file.**

The entry replicates the parameters that were passed to the `ifconfig` command in Step 5.

**a. For example, on the `enigma` system, add the following entry to the `hostname6.ip6.tun0` file:**

```
6000:6666::aaaa:1116 6000:3333::eeee:1113 \
tsrc 2001::aaaa:6666:6666  tdst 2001::eeee:3333:3333 router up
```

**b. On the `partym` system, add the following entry to the `hostname6.ip6.tun0` file:**

```
6000:3333::eeee:1113 6000:6666::aaaa:1116 \
tsrc 2001::eeee:3333:3333   tdst 2001::aaaa:6666:6666  router up
```

**12 On each system, configure the interface files to pass the correct parameters to the routing daemon.**

**a. On the `enigma` system, modify the `/etc/hostname6.`*interface* files.**

```
# cat enigma hostname6.hme0
6000:6666::aaaa:1116 inet6 private

#  cat enigma hostname6.hme1
2001::aaaa:6666:6666 inet6 router
```

**b. On the `partym` system, modify the `/etc/hostname6.`*interface* files.**

```
# cat partym hostname6.hme0
6000:3333::eeee:1113 inet6 private

# cat partym hostname6.hme1
2001::eeee:3333:3333 inet6 router
```

**13 On each system, run a routing protocol.**

```
# routeadm -e ipv6-routing
# routeadm -u
```

## ▼ How to Protect a VPN With an IPsec Tunnel in Transport Mode Over IPv4

This procedure extends the procedure "How to Secure Traffic Between Two Systems With IPsec" on page 496. In addition to connecting two systems, you are connecting two intranets that connect to these two systems. The systems in this procedure function as gateways.

This procedure uses the setup that is described in "Description of the Network Topology for the IPsec Tasks to Protect a VPN" on page 512.

**1 On the system console on one of the systems, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

---

**2 Control the flow of packets before configuring IPsec.**

**a. Ensure that IP forwarding and IP dynamic routing are disabled.**

```
# routeadm
Configuration       Current         Current
       Option       Configuration  System State
---------------------------------------------------
IPv4 forwarding     disabled           disabled
   IPv4 routing     default (enabled)   enabled
...
```

If IP forwarding and IP dynamic routing are enabled, you can disable them by typing:

```
# routeadm -d ipv4-routing -d ipv4-forwarding
# routeadm -u
```

Turning off IP forwarding prevents packets from being forwarded from one network to another network through this system. For a description of the `routeadm` command, see the `routeadm`(1M) man page.

**b. Turn on IP strict destination multihoming.**

```
# ndd -set /dev/ip ip_strict_dst_multihoming 1
```

Turning on IP strict destination multihoming ensures that packets for one of the system's destination addresses arrive at the correct destination address.

When you turn off IP forwarding and turn on IP strict destination multihoming, fewer packets flow all the way through the system. When strict destination multihoming is enabled, packets that arrive on a particular interface must be addressed to one of the local IP addresses of that interface. All other packets, even packets that are addressed to other local addresses of the system, are dropped.

**c. Disable most network services, and possibly all network services.**

**Note –** If your system was installed with the "limited" SMF profile, then you can skip this step. Network services, with the exception of Solaris Secure Shell, are disabled.

The disabling of network services prevents IP packets from doing any harm to the system. For example, an SNMP daemon, a `telnet` connection, or an `rlogin` connection could be exploited.

Choose one of the following options:

- If you are running the Solaris 10 11/06 release or a later release, run the "limited" SMF profile.

```
# netservices limited
```

- Otherwise, individually disable network services.

```
# svcadm disable network/ftp:default
# svcadm disable network/finger:default
# svcadm disable network/login:rlogin
# svcadm disable network/nfs/server:default
# svcadm disable network/rpc/rstat:default
# svcadm disable network/smtp:sendmail
# svcadm disable network/telnet:default
# svcs | grep network
online         Aug_02   svc:/network/loopback:default

online         Aug_09   svc:/network/ssh:default
```

**3 On each system, add a pair of SAs between the two systems.**

Choose one of the following options:

- Configure IKE to manage the keys for the SAs. Use one of the procedures in "Configuring IKE (Task Map)" on page 553 to configure IKE for the VPN.
- If you have an overriding reason to manually manage the keys, see "How to Manually Create IPsec Security Associations" on page 503.

**4 On each system, add IPsec policy.**

Edit the `/etc/inet/ipsecinit.conf` file to add the IPsec policy for the VPN. To strengthen the policy, see Example 20–10.

**a. For example, on the `enigma` system, type the following entry into the `ipsecinit.conf` file:**

```
# LAN traffic to and from this host can bypass IPsec.
{laddr 10.16.16.6 dir both} bypass {}
```

```
# WAN traffic uses ESP with 3DES and MD5.
{tunnel ip.tun0 negotiate transport}
 ipsec {encr_algs 3des encr_auth_algs md5 sa shared}
```

**b. On the `partym` system, type the following entry into the `ipsecinit.conf` file:**

```
# LAN traffic to and from this host can bypass IPsec.
{laddr 10.1.3.3 dir both} bypass {}

# WAN traffic uses ESP with 3DES and MD5.
{tunnel ip.tun0 negotiate transport}
 ipsec {encr_algs 3des encr_auth_algs md5 sa shared}
```

**5 On each system, configure the tunnel, `ip.tun0`.**

Use the following `ifconfig` commands to create the point-to-point interface:

```
# ifconfig ip.tun0 plumb

# ifconfig ip.tun0 system1-point system2-point \
tsrc system1-taddr tdst system2-taddr
```

**a. For example, on the `enigma` system, type the following commands:**

```
# ifconfig ip.tun0 plumb

# ifconfig ip.tun0 10.16.16.6 10.1.3.3 \
tsrc 192.168.116.16 tdst 192.168.13.213
```

**b. On the `partym` system, type the following commands:**

```
# ifconfig ip.tun0 plumb

# ifconfig ip.tun0 10.1.3.3 10.16.16.6  \
tsrc 192.168.13.213 tdst 192.168.116.16
```

**6 Protect the tunnel with the IPsec policy that you created.**

```
# ipsecconf
```

**7 Bring up the router for the tunnel.**

```
# ifconfig ip.tun0 router up
```

**8 On each system, turn on IP forwarding for the `hme1` interface.**

```
# ifconfig hme1 router
```

IP forwarding means that packets that arrive from somewhere else can be forwarded. IP forwarding also means that packets that leave this interface might have originated somewhere else. To successfully forward a packet, both the receiving interface and the transmitting interface must have IP forwarding turned on.

Because the hme1 interface is *inside* the intranet, IP forwarding must be turned on for hme1. Because ip.tun0 connects the two systems through the Internet, IP forwarding must be turned on for ip.tun0.

The hme0 interface has its IP forwarding turned off to prevent an *outside* adversary from injecting packets into the protected intranet. The *outside* refers to the Internet.

**9 On each system, ensure that routing protocols do not advertise the default route within the intranet.**

```
# ifconfig hme0 private
```

Even if hme0 has IP forwarding turned off, a routing protocol implementation might still advertise the interface. For example, the in.routed protocol might still advertise that hme0 is available to forward packets to its peers inside the intranet. By setting the interface's *private* flag, these advertisements are prevented.

**10 On each system, manually add a default route over hme0.**

The default route must be a router with direct access to the Internet.

```
# route add default router-on-hme0-subnet
```

**a. For example, on the enigma system, add the following route:**

```
# route add default 192.168.116.4
```

**b. On the partym system, add the following route:**

```
# route add default 192.168.13.5
```

Even though the hme0 interface is not part of the intranet, hme0 does need to reach across the Internet to its peer system. To find its peer, hme0 needs information about Internet routing. The VPN system appears to be a host, rather than a router, to the rest of the Internet. Therefore, you can use a default router or run the router discovery protocol to find a peer system. For more information, see the route(1M) and in.routed(1M) man pages.

**11 On each system, ensure that the VPN starts after a reboot by adding an entry to the /etc/hostname.ip.tun0 file.**

```
system1-point system2-point tsrc system1-taddr \
tdst system2-taddr encr_algs 3des encr_auth_algs md5 router up
```

**a. For example, on the enigma system, add the following entry to the hostname.ip.tun0 file:**

```
10.16.16.6 10.1.3.3 tsrc 192.168.116.16 \
tdst 192.168.13.213 router up
```

**b. On the partym system, add the following entry to the hostname.ip.tun0 file:**

```
10.1.3.3 10.16.16.6 tsrc 192.168.13.213 \
tdst 192.168.116.16 router up
```

**12 On each system, configure the interface files to pass the correct parameters to the routing daemon.**

**a. On the `enigma` system, modify the `/etc/hostname.`*interface* files.**

```
# cat enigma hostname.hme0
10.16.16.6 private

# cat enigma hostname.hme1
192.168.116.16 router
```

**b. On the `partym` system, modify the `/etc/hostname.`*interface* files.**

```
# cat partym hostname.hme0
10.1.3.3 private

# cat partym hostname.hme1
192.168.13.213 router
```

**13 On each system, run a routing protocol.**

You might need to configure the routing protocol before enabling routing. For more information, see "Routing Protocols in the Solaris OS" on page 253. For a procedure, see "How to Configure an IPv4 Router" on page 116.

```
# routeadm -e ipv4-routing
# routeadm -u
```

**Example 20–10** Requiring IPsec Policy on All Systems in Transport Mode

In this example, the administrator comments out the `bypass` policy that was configured in Step 4, thereby strengthening the protection. With this policy configuration, each system on the LAN must activate IPsec to communicate with the router.

```
# LAN traffic must implement IPsec.
# {laddr 10.1.3.3 dir both} bypass {}

# WAN traffic uses ESP with 3DES and MD5.
{tunnel ip.tun0 negotiate transport} ipsec {encr_algs 3des encr_auth_algs md5}
```

**Example 20–11** Using Deprecated Syntax to Configure an IPsec Tunnel in Transport Mode

In this example, the administrator is connecting a Solaris 10 8/07 system with a system that is running the Solaris 10 release. Therefore, the administrator uses Solaris 10 syntax in the configuration file and includes the IPsec algorithms in the `ifconfig` command.

The administrator follows the procedure "How to Protect a VPN With an IPsec Tunnel in Transport Mode Over IPv4" on page 524 with the following changes in syntax.

- For Step 4, the syntax is the following:

```
# LAN traffic to and from this address can bypass IPsec.
{laddr 10.1.3.3 dir both} bypass {}

# WAN traffic uses ESP with 3DES and MD5.
{} ipsec {encr_algs 3des encr_auth_algs md5}
```

- For Step 5 to Step 7, the syntax is the following:

```
# ifconfig ip.tun0 plumb

# ifconfig ip.tun0 10.16.16.6 10.1.3.3 \
tsrc 192.168.116.16 tdst 192.168.13.213 \
encr_algs 3des encr_auth_algs md5

# ifconfig ip.tun0 router up

# ifconfig ip.tun0 plumb

# ifconfig ip.tun0 10.16.16.6 10.1.3.3 \
tsrc 192.168.116.16 tdst 192.168.13.213 \
encr_algs 3des encr_auth_algs md5
```

  The IPsec policy that is passed to the `ifconfig` commands must be the same as the IPsec policy in the `ipsecinit.conf` file. Upon reboot, each system reads the `ipsecinit.conf` file for its policy.

- For Step 11, the syntax is the following:

```
10.16.16.6 10.1.3.3 tsrc 192.168.116.16 \
tdst 192.168.13.213 encr_algs 3des encr_auth_algs md5 router up
```

## How to Protect a VPN With an IPsec Tunnel in Transport Mode Over IPv6

To set up a VPN on an IPv6 network, you follow the same steps as for an IPv4 network. However, the syntax of the commands is slightly different. For a fuller description of the reasons for running particular commands, see "How to Protect a VPN With an IPsec Tunnel in Tunnel Mode Over IPv4" on page 514.

This procedure uses the following configuration parameters.

| Parameter | Europe | California |
|---|---|---|
| System name | `enigma` | `partym` |
| System intranet interface | `hme1` | `hme1` |
| System Internet interface | `hme0` | `hme0` |
| System intranet address | `6000:6666::aaaa:1116` | `6000:3333::eeee:1113` |
| System Internet address | `2001::aaaa:6666:6666` | `2001::eeee:3333:3333` |
| Name of Internet router | `router-E` | `router-C` |
| Address of Internet router | `2001::aaaa:0:4` | `2001::eeee:0:1` |
| Tunnel name | `ip6.tun0` | `ip6.tun0` |

**1 On the system console on one of the systems, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

**2 Control the flow of packets before configuring IPsec.**

For the effects of these commands, see Step 2 in "How to Protect a VPN With an IPsec Tunnel in Transport Mode Over IPv4" on page 524.

**a. Ensure that IP forwarding and IP dynamic routing are disabled.**

```
# routeadm
Configuration       Current         Current
       Option       Configuration  System State
-------------------------------------------------
...
IPv6 forwarding     disabled          disabled
   IPv6 routing     disabled          disabled
```

If IP forwarding and IP dynamic routing are enabled, you can disable them by typing:

```
# routeadm -d ipv6-forwarding -d ipv6-routing
# routeadm -u
```

**b. Turn on IP strict destination multihoming.**

```
# ndd -set /dev/ip ip6_strict_dst_multihoming 1
```

**c. Disable most network services, and possibly all network services.**

---

**Note –** If your system was installed with the "limited" SMF profile, then you can skip this step. Network services, with the exception of Solaris Secure Shell, are disabled.

---

The disabling of network services prevents IP packets from doing any harm to the system. For example, an SNMP daemon, a `telnet` connection, or an `rlogin` connection could be exploited.

Choose one of the following options:

- If you are running the Solaris 10 11/06 release or a later release, run the "limited" SMF profile.

  ```
  # netservices limited
  ```

- Otherwise, individually disable network services.

  ```
  # svcadm disable network/ftp:default
  # svcadm disable network/finger:default
  # svcadm disable network/login:rlogin
  # svcadm disable network/nfs/server:default
  # svcadm disable network/rpc/rstat:default
  # svcadm disable network/smtp:sendmail
  # svcadm disable network/telnet:default
  # svcs | grep network
  online         Aug_02   svc:/network/loopback:default

  online         Aug_09   svc:/network/ssh:default
  ```

**3 On each system, add a pair of SAs between the two systems.**

Choose one of the following options:

- Configure IKE to manage the keys for the SAs. Use one of the procedures in "Configuring IKE (Task Map)" on page 553 to configure IKE for the VPN.
- If you have an overriding reason to manually manage the keys, see "How to Manually Create IPsec Security Associations" on page 503.

4 **On each system, add IPsec policy.**

Edit the `/etc/inet/ipsecinit.conf` file to add the IPsec policy for the VPN.

a. **For example, on the `enigma` system, type the following entry into the `ipsecinit.conf` file:**

```
# IPv6 Neighbor Discovery messages bypass IPsec.
{ulp ipv6-icmp type 133-137 dir both} pass {}

# LAN traffic can bypass IPsec.
{laddr 6000:6666::aaaa:1116 dir both} bypass {}

# WAN traffic uses ESP with 3DES and MD5.
{tunnel ip6.tun0 negotiate transport}
 ipsec {encr_algs 3des encr_auth_algs md5}
```

b. **On the `partym` system, type the following entry into the `ipsecinit.conf` file:**

```
# IPv6 Neighbor Discovery messages bypass IPsec.
{ulp ipv6-icmp type 133-137 dir both} pass {}

# LAN traffic can bypass IPsec.
{laddr 6000:3333::eeee:1113 dir both} bypass {}

# WAN traffic uses ESP with 3DES and MD5.
{tunnel ip6.tun0 negotiate transport}
 ipsec {encr_algs 3des encr_auth_algs md5}
```

5 **On each system, configure a secure tunnel, `ip6.tun0`.**

a. **For example, on the `enigma` system, type the following commands:**

```
# ifconfig ip6.tun0 inet6 plumb

# ifconfig ip6.tun0 inet6 6000:6666::aaaa:1116 6000:3333::eeee:1113 \
tsrc 2001::aaaa:6666:6666   tdst 2001::eeee:3333:3333
```

b. **On the `partym` system, type the following commands:**

```
# ifconfig ip6.tun0 inet6 plumb

# ifconfig ip6.tun0 inet6  6000:3333::eeee:1113  6000:6666::aaaa:1116 \
tsrc 2001::eeee:3333:3333   tdst 2001::aaaa:6666:6666
```

6 **Protect the tunnel with the IPsec policy that you created.**

```
# ipsecconf
```

7 **Bring up the router for the tunnel.**

```
# ifconfig ip6.tun0 router up
```

8 **On each system, turn on IP forwarding for the `hme1` interface.**

```
# ifconfig hme1 router
```

9 **On each system, ensure that routing protocols do not advertise the default route within the intranet.**

```
# ifconfig hme0 private
```

10 **On each system, manually add a default route over `hme0`.**

The default route must be a router with direct access to the Internet.

a. **For example, on the `enigma` system, add the following route:**

```
# route add -inet6 default 2001::aaaa:0:4
```

b. **On the `partym` system, add the following route:**

```
# route add -inet6 default 2001::eeee:0:1
```

11 **On each system, ensure that the VPN starts after a reboot by adding an entry to the `/etc/hostname6.ip6.tun0` file.**

The entry replicates the parameters that were passed to the `ifconfig` command in Step 5.

a. **For example, on the `enigma` system, add the following entry to the `hostname6.ip6.tun0` file:**

```
6000:6666::aaaa:1116  6000:3333::eeee:1113 \
tsrc 2001::aaaa:6666:6666   tdst 2001::eeee:3333:3333  router up
```

b. **On the `partym` system, add the following entry to the `hostname6.ip6.tun0` file:**

```
6000:3333::eeee:1113  6000:6666::aaaa:1116 \
tsrc 2001::eeee:3333:3333   tdst 2001::aaaa:6666:6666  router up
```

12 **On each system, configure the interface files to pass the correct parameters to the routing daemon.**

a. **On the `enigma` system, modify the `/etc/hostname6.`*interface* files.**

```
# cat enigma hostname6.hme0
6000:6666::aaaa:1116 inet6 private

#  cat enigma hostname6.hme1
2001::aaaa:6666:6666 inet6 router
```

**b. On the `partym` system, modify the `/etc/hostname6.`*interface* files.**

```
# cat partym hostname6.hme0
6000:3333::eeee:1113 inet6 private

# cat partym hostname6.hme1
2001::eeee:3333:3333 inet6 router
```

**13 On each system, run a routing protocol.**

```
# routeadm -e ipv6-routing
# routeadm -u
```

**Example 20–12** Using Deprecated Syntax to Configure IPsec in Transport Mode Over IPv6

In this example, the administrator is connecting a Solaris 10 8/07 system with a system that is running the Solaris 10 release. Therefore, the administrator uses Solaris 10 syntax in the configuration file and includes the IPsec algorithms in the `ifconfig` command.

The administrator follows the procedure "How to Protect a VPN With an IPsec Tunnel in Transport Mode Over IPv6" on page 530 with the following changes in syntax.

- For Step 4, the syntax is the following:

```
# IPv6 Neighbor Discovery messages bypass IPsec.
{ulp ipv6-icmp type 133-137 dir both} pass {}

# LAN traffic can bypass IPsec.
{laddr 6000:3333::eeee:1113 dir both} bypass {}

# WAN traffic uses ESP with 3DES and MD5.
{} ipsec {encr_algs 3des encr_auth_algs md5}
```

- For Step 5 to Step 7, the syntax is the following:

```
# ifconfig ip6.tun0 inet6 plumb

# ifconfig ip6.tun0 inet6 6000:6666::aaaa:1116 6000:3333::eeee:1113 \
tsrc 2001::aaaa:6666:6666 tdst 2001::eeee:3333:3333 \
encr_algs 3des encr_auth_algs md5

# ifconfig ip6.tun0 inet6 router up
```

  The IPsec policy that is passed to the `ifconfig` commands must be the same as the IPsec policy in the `ipsecinit.conf` file. Upon reboot, each system reads the `ipsecinit.conf` file for its policy.

- For Step 11, the syntax is the following:

```
6000:6666::aaaa:1116  6000:3333::eeee:1113 \
tsrc 2001::aaaa:6666:6666   tdst 2001::eeee:3333:3333 \
encr_algs 3des encr_auth_algs md5 router up
```

◆ ◆ ◆ CHAPTER 21

# IP Security Architecture (Reference)

This chapter contains the following reference information:


- "ipsecconf Command" on page 537
- "ipsecinit.conf File" on page 538
- "ipsecalgs Command" on page 540
- "Security Associations Database for IPsec" on page 540
- "Utilities for Key Generation in IPsec" on page 541
- "IPsec Extensions to Other Utilities" on page 542


For instructions on how to implement IPsec on your network, see Chapter 20, "Configuring IPsec (Tasks)." For an overview of IPsec, see Chapter 19, "IP Security Architecture (Overview)."

## ipsecconf Command

You use the ipsecconf command to configure the IPsec policy for a host. When you run the command to configure the policy, the system creates the IPsec policy entries in the kernel. The system uses these entries to check the policy on all inbound and outbound IP datagrams. Forwarded datagrams are not subjected to policy checks that are added by using this command. The ipsecconf command also configures the security policy database (SPD).

- For information about how to protect forwarded packets, see the ifconfig(1M) and tun(7M) man pages.
- For IPsec policy options, see the ipsecconf(1M) man page.
- For instructions about how to use the ipsecconf command to protect traffic between systems, see "Configuring IKE (Task Map)" on page 553.

You must become superuser or assume an equivalent role to invoke the ipsecconf command. The command accepts entries that protect traffic in both directions. The command also accepts entries that protect traffic in only one direction.

Policy entries with a format of local address and remote address can protect traffic in both directions with a single policy entry. For example, entries that contain the patterns `laddr host1` and `raddr host2` protect traffic in both directions, if no direction is specified for the named host. Thus, you need only one policy entry for each host.

Policy entries with a format of source address to destination address protect traffic in only one direction. For example, a policy entry of the pattern `saddr host1 daddr host2` protects inbound traffic or outbound traffic, not both directions. Thus, to protect traffic in both directions, you need to pass the `ipsecconf` command another entry, as in `saddr host2 daddr host1`.

To ensure that the IPsec policy is active when the machine boots, you can create an IPsec policy file, `/etc/inet/ipsecinit.conf`. This file is read when the network services are started. For instructions on how to create an IPsec policy file, see "Protecting Traffic With IPsec (Task Map)" on page 495.

# `ipsecinit.conf` File

To invoke IPsec security policies when you start the Solaris Operating System (Solaris OS), you create a configuration file to initialize IPsec with your specific IPsec policy entries. You should name the file `/etc/inet/ipsecinit.conf`. See the `ipsecconf`(1M) man page for details about policy entries and their format. After policies are configured, you can use the `ipsecconf` command to view or modify the existing configuration.

## Sample `ipsecinit.conf` File

The Solaris software includes a sample IPsec policy file, `ipsecinit.sample`. You can use the file as a template to create your own `ipsecinit.conf` file. The `ipsecinit.sample` file contains the following examples:

```
#
# For example,
#
#     {rport 23} ipsec {encr_algs des encr_auth_algs md5}
#
# will protect the telnet traffic originating from the host with ESP using
# DES and MD5. Also:
#
#     {raddr 10.5.5.0/24} ipsec {auth_algs any}
#
# will protect traffic to or from the 10.5.5.0 subnet with AH
# using any available algorithm.
#
#
```

```
# To do basic filtering, a drop rule may be used. For example:
#
#    {lport 23 dir both} drop {}
# will disallow any remote system from telnetting in.
#
# If you are using IPv6, it may be useful to bypass neighbor discovery
# to allow in.iked to work properly with on-link neighbors. To do that,
# add the following lines:
#
#        {ulp ipv6-icmp type 133-137 dir both } pass { }
#
# This will allow neighbor discovery to work normally.
```

## Security Considerations for `ipsecinit.conf` and `ipsecconf`

Use extreme caution if transmitting a copy of the `ipsecinit.conf` file over a network. An adversary can read a network-mounted file as the file is being read. If, for example, the `/etc/inet/ipsecinit.conf` file is accessed or is copied from an NFS-mounted file system, an adversary can change the policy that is contained in the file.

Ensure that you set up IPsec policies before starting any communications, because existing connections might be affected by the addition of new policy entries. Similarly, do not change policies in the middle of a communication.

Specifically, IPsec policy cannot be changed for SCTP, TCP, or UDP sockets on which a `connect()` or `accept()` function call has been issued. A socket whose policy cannot be changed is called a *latched socket*. New policy entries do not protect sockets that are already latched. For more information, see the `connect`(3SOCKET) and `accept`(3SOCKET) man pages.

Protect your naming system. If the following two conditions are met, then your host names are no longer trustworthy:

- Your source address is a host that can be looked up over the network.
- Your naming system is compromised.

Security weaknesses often arise from the misapplication of tools, not from the actual tools. You should be cautious when using the `ipsecconf` command. Use a console or other hard-connected TTY for the safest mode of operation.

# ipsecalgs Command

The Solaris cryptographic framework provides authentication and encryption algorithms to IPsec. You use the ipsecalgs command to query and modify the list of protocols and the list of algorithms that IPsec supports. The ipsecalgs command stores this information in tabular format in the IPsec protocols and algorithms file, /etc/inet/ipsecalgs. This file must never be edited manually.

The valid IPsec protocols and algorithms are described by the ISAKMP domain of interpretation (DOI), which is covered by RFC 2407. In a general sense, a DOI defines data formats, network traffic exchange types, and conventions for naming security-relevant information. Security policies, cryptographic algorithms, and cryptographic modes are examples of security-relevant information.

Specifically, the ISAKMP DOI defines the naming and numbering conventions for the valid IPsec algorithms and for their protocols, PROTO_IPSEC_AH and PROTO_IPSEC_ESP. Each algorithm is associated with exactly one protocol. These ISAKMP DOI definitions are in the /etc/inet/ipsecalgs file. The algorithm and protocol numbers are defined by the Internet Assigned Numbers Authority (IANA). The ipsecalgs command makes the list of algorithms for IPsec extensible.

For more information about the algorithms, refer to the ipsecalgs(1M) man page. For more information on the Solaris cryptographic framework, see Chapter 13, "Solaris Cryptographic Framework (Overview)," in *System Administration Guide: Security Services*.

# Security Associations Database for IPsec

Information on key material for IPsec security services is maintained in a security associations database (SADB). Security associations (SAs) protect inbound packets and outbound packets. The SADBs are maintained by a user process, or possibly multiple cooperating processes, that send messages over a special kind of socket. This method of maintaining SADBs is analogous to the method that is described in the route(7P) man page. Only superuser or a user who has assumed an equivalent role can access the database.

The in.iked daemon and the ipseckey command use the PF_KEY socket interface to maintain SADBs. For more information on how SADBs handle requests and messages, see the pf_key(7P) man page.

# Utilities for Key Generation in IPsec

The IKE protocol provides automatic key management for IPv4 and IPv6 addresses. See Chapter 23, "Configuring IKE (Tasks)," for instructions on how to set up IKE. The manual keying utility is the `ipseckey` command, which is described in the `ipseckey`(1M) man page.

You use the `ipseckey` command to manually populate the security association databases (SADBs) when automated key management is not used. When you invoke the `ipseckey` command with no arguments, the command enters an interactive mode and displays a prompt that enables you to make entries. Some commands require an explicit security association (SA) type, while others permit you to specify the SA type and act on all SA types.

While the `ipseckey` command has only a limited number of general options, the command supports a rich command language. You can specify that requests be delivered by means of a programmatic interface specific for manual keying. For additional information, see the `pf_key`(7P) man page.

## Security Considerations for `ipseckey`

The `ipseckey` command enables superuser or a role with the Network Security profile to enter sensitive cryptographic keying information. If an adversary gains access to this information, the adversary can compromise the security of IPsec traffic.

You should consider the following issues when you handle keying material and use the `ipseckey` command:

- Have you refreshed the keying material? Periodic key refreshment is a fundamental security practice. Key refreshment guards against potential weaknesses of the algorithm and keys, and limits the damage of an exposed key.
- Is the TTY going over a network? Is the `ipseckey` command in interactive mode?
  - In interactive mode, the security of the keying material is the security of the network path for this TTY's traffic. You should avoid using the `ipseckey` command over a clear-text telnet or rlogin session.
  - Even local windows might be vulnerable to attacks by a concealed program that reads window events.
- Have you used the `-f` option? Is the file being accessed over the network? Can the file be read by the world?
  - An adversary can read a network-mounted file as the file is being read. You should avoid using a world-readable file that contains keying material.

- Protect your naming system. If the following two conditions are met, then your host names are no longer trustworthy:
  - Your source address is a host that can be looked up over the network.
  - Your naming system is compromised.

Security weaknesses often arise from the misapplication of tools, not from the actual tools. You should be cautious when using the `ipseckey` command. Use a console or other hard-connected TTY for the safest mode of operation.

# IPsec Extensions to Other Utilities

The `ifconfig` command has options to manage the IPsec policy on a tunnel interface. The `snoop` command can parse AH and ESP headers.

## `ifconfig` Command and IPsec

**In the Solaris 10, Solaris 10 7/05, Solaris 10 1/06, and Solaris 10 11/06 releases:** To support IPsec, the following security options are available from the `ifconfig` command. These security options are handled by the `ipsecconf` command in the Solaris 10 8/07 release.

- `auth_algs`
- `encr_auth_algs`
- `encr_algs`

You must specify all IPsec security options for a tunnel in one invocation. For example, if you are using only ESP to protect traffic, you would configure the tunnel, `ip.tun0`, once with both security options, as in:

```
# ifconfig ip.tun0   encr_algs 3des encr_auth_algs md5
```

Similarly, an `ipsecinit.conf` entry would configure the tunnel once with both security options, as in:

```
# WAN traffic uses ESP with 3DES and MD5.
   {} ipsec {encr_algs 3des encr_auth_algs md5}
```

### `auth_algs` Security Option

This option enables IPsec AH for a tunnel with a specified authentication algorithm. The `auth_algs` option has the following format:

`auth_algs` *authentication-algorithm*

For the algorithm, you can specify either a number or an algorithm name, including the parameter *any*, to express no specific algorithm preference. To disable tunnel security, specify the following option:

```
auth_algs none
```

For a list of available authentication algorithms, run the `ipsecalgs` command.

---

**Note –** The `auth_algs` option cannot work with NAT-Traversal. For more information, see "IPsec and NAT Traversal" on page 490.

---

## `encr_auth_algs` Security Option

This option enables IPsec ESP for a tunnel with a specified authentication algorithm. The `encr_auth_algs` option has the following format:

`encr_auth_algs` *authentication-algorithm*

For the algorithm, you can specify either a number or an algorithm name, including the parameter *any*, to express no specific algorithm preference. If you specify an ESP encryption algorithm, but you do not specify the authentication algorithm, the ESP authentication algorithm value defaults to the parameter *any*.

For a list of available authentication algorithms, run the `ipsecalgs` command.

## `encr_algs` Security Option

This option enables IPsec ESP for a tunnel with a specified encryption algorithm. The `encr_algs` option has the following format:

`encr_algs` *encryption-algorithm*

For the algorithm, you can specify either a number or an algorithm name. To disable tunnel security, specify the following option:

```
encr_algs none
```

If you specify an ESP authentication algorithm, but not an encryption algorithm, ESP's encryption value defaults to the parameter *null*.

For a list of available encryption algorithms, run the `ipsecalgs` command.

## snoop Command and IPsec

The snoop command can parse AH and ESP headers. Because ESP encrypts its data, the snoop command cannot see encrypted headers that are protected by ESP. AH does not encrypt data. Therefore, traffic that is protected by AH can be inspected with the snoop command. The -V option to the command shows when AH is in use on a packet. For more details, see the snoop(1M) man page.

For a sample of verbose snoop output on a protected packet, see "How to Verify That Packets Are Protected With IPsec" on page 507.

◆ ◆ ◆ CHAPTER 22

# Internet Key Exchange (Overview)

Internet Key Exchange (IKE) automates key management for IPsec. This chapter contains the following information about IKE:


- "What's New in IKE?" on page 545
- "Key Management With IKE" on page 546
- "IKE Key Negotiation" on page 546
- "IKE Configuration Choices" on page 548
- "IKE and Hardware Acceleration" on page 549
- "IKE and Hardware Storage" on page 549
- "IKE Utilities and Files" on page 550
- "Changes to IKE for the Solaris 10 Release" on page 551


For instructions on implementing IKE, see Chapter 23, "Configuring IKE (Tasks)." For reference information, see Chapter 24, "Internet Key Exchange (Reference)." For information about IPsec, see Chapter 19, "IP Security Architecture (Overview)."

## What's New in IKE?

**Solaris 10 8/07**: Starting in this release, IKE can use the AES algorithm and can be configured in the global zone for use in non-global zones.

- The `SO_ALLZONES` socket option enables IKE to handle traffic in non-global zones.
- For a complete listing of new Solaris features and a description of Solaris releases, see *Solaris Express Developer Edition What's New*.

# Key Management With IKE

The management of keying material for IPsec security associations (SAs) is called *key management*. Automatic key management requires a secure channel of communication for the creation, authentication, and exchange of keys. The Solaris Operating System uses Internet Key Exchange (IKE) to automate key management. IKE easily scales to provide a secure channel for a large volume of traffic. IPsec SAs on IPv4 and IPv6 packets can take advantage of IKE.

When IKE is used on a system with a Sun™ Crypto Accelerator 1000 board or a Sun Crypto Accelerator 4000 board, the public key operations can be offloaded to the accelerator. Operating system resources are not used for public key operations. When IKE is used on a system with a Sun Crypto Accelerator 4000 board, the certificates, public keys, and private keys can be stored on the board. Key storage that is off the system provides an additional layer of protection.

# IKE Key Negotiation

The IKE daemon, `in.iked`, negotiates and authenticates keying material for SAs in a protected manner. The daemon uses random seeds for keys from internal functions provided by the Solaris Operating System. IKE provides perfect forward secrecy (PFS). In PFS, the keys that protect data transmission are not used to derive additional keys. Also, seeds used to create data transmission keys are not reused. See the `in.iked`(1M) man page.

When the IKE daemon discovers a remote system's public encryption key, the local system can then use that key. The system encrypts messages by using the remote system's public key. The messages can be read only by that remote system. The IKE daemon performs its job in two phases. The phases are called *exchanges*.

## IKE Key Terminology

The following table lists terms that are used in key negotiation, provides their commonly used acronyms, and gives a definition and use for each term.

**TABLE 22–1** Key Negotiation Terms, Acronyms, and Uses

| Key Negotiation Term | Acronym | Definition and Use |
|---|---|---|
| Key exchange | | The process of generating keys for asymmetric cryptographic algorithms. The two main methods are RSA protocols and the Diffie-Hellman protocol. |
| Diffie-Hellman protocol | DH | A key exchange protocol that involves key generation and key authentication. Often called *authenticated key exchange*. |

**TABLE 22–1** Key Negotiation Terms, Acronyms, and Uses *(Continued)*

| Key Negotiation Term | Acronym | Definition and Use |
|---|---|---|
| RSA protocol | RSA | A key exchange protocol that involves key generation and key transport. The protocol is named for its three creators, Rivest, Shamir, and Adleman. |
| Perfect forward secrecy | PFS | Applies to authenticated key exchange only. PFS ensures that long-term secret material for keys does not compromise the secrecy of the exchanged keys from previous communications.<br><br>In PFS, the key that is used to protect transmission of data is not used to derive additional keys. Also, the source of the key that is used to protect data transmission is never used to derive additional keys. |
| Oakley method | | A method for establishing keys for Phase 2 in a secure manner. This protocol is analogous to the Diffie-Hellman method of key exchange. Similar to Diffie-Hellman, Oakley group key exchange involves key generation and key authentication. The Oakley method is used to negotiate PFS. |

# IKE Phase 1 Exchange

The Phase 1 exchange is known as *Main Mode*. In the Phase 1 exchange, IKE uses public key encryption methods to authenticate itself with peer IKE entities. The result is an Internet Security Association and Key Management Protocol (ISAKMP) security association (SA). An ISAKMP SA is a secure channel for IKE to negotiate keying material for the IP datagrams. Unlike IPsec SAs, the ISAKMP SAs are bidirectional, so only one security association is needed.

How IKE negotiates keying material in the Phase 1 exchange is configurable. IKE reads the configuration information from the `/etc/inet/ike/config` file. Configuration information includes the following:

- Global parameters, such as the names of public key certificates
- Whether perfect forward secrecy (PFS) is used
- The interfaces that are affected
- The security protocols and their algorithms
- The authentication method

The two authentication methods are preshared keys and public key certificates. The public key certificates can be self-signed. Or, the certificates can be issued by a certificate authority (CA) from a public key infrastructure (PKI) organization. Organizations include beTrusted, Entrust, GeoTrust, RSA Security, and Verisign.

## IKE Phase 2 Exchange

The Phase 2 exchange is known as *Quick Mode*. In the Phase 2 exchange, IKE creates and manages the IPsec SAs between systems that are running the IKE daemon. IKE uses the secure channel that was created in the Phase 1 exchange to protect the transmission of keying material. The IKE daemon creates the keys from a random number generator by using the `/dev/random` device. The daemon refreshes the keys at a configurable rate. The keying material is available to algorithms that are specified in the configuration file for IPsec policy, `ipsecinit.conf`.

# IKE Configuration Choices

The `/etc/inet/ike/config` configuration file contains IKE policy entries. For two IKE daemons to authenticate each other, the entries must be valid. Also, keying material must be available. The entries in the configuration file determine the method for using the keying material to authenticate the Phase 1 exchange. The choices are preshared keys or public key certificates.

The entry `auth_method preshared` indicates that preshared keys are used. Values for `auth_method` other than `preshared` indicate that public key certificates are to be used. Public key certificates can be self-signed, or the certificates can be installed from a PKI organization. For more information, see the `ike.config`(4) man page.

## IKE With Preshared Keys

Preshared keys are created by an administrator on one system. The keys are then shared out of band with administrators of remote systems. You should take care to create large random keys and to protect the file and the out-of-band transmission. The keys are placed in the `/etc/inet/secret/ike.preshared` file on each system. The `ike.preshared` file is for IKE as the `ipseckeys` file is for IPsec. Any compromise of the keys in the `ike.preshared` file compromises all keys that are derived from the keys in the file.

One system's preshared key must be identical to its remote system's key. The keys are tied to a particular IP address. Keys are most secure when one administrator controls the communicating systems. For more information, see the `ike.preshared`(4) man page.

## IKE With Public Key Certificates

Public key certificates eliminate the need for communicating systems to share secret keying material out of band. Public keys use the Diffie-Hellman protocol (DH) for authenticating and negotiating keys. Public key certificates come in two flavors. The certificates can be self-signed, or the certificates can be certified by a certificate authority (CA).

Self-signed public key certificates are created by you, the administrator. The `ikecert certlocal -ks` command creates the private part of the public-private key pair for the system. You then get the self-signed certificate output in X.509 format from the remote system. The remote system's certificate is input to the `ikecert certdb` command for the public part of the key pair. The self-signed certificates reside in the `/etc/inet/ike/publickeys` directory on the communicating systems. When you use the `-T` option, the certificates reside on attached hardware.

Self-signed certificates are a halfway point between preshared keys and CAs. Unlike preshared keys, a self-signed certificate can be used on a mobile machine or on a system that might be renumbered. To self-sign a certificate for a system without a fixed number, use a `DNS` (`www.example.org`) or `email` (`root@domain.org`) alternative name.

Public keys can be delivered by a PKI or a CA organization. You install the public keys and their accompanying CAs in the `/etc/inet/ike/publickeys` directory. When you use the `-T` option, the certificates reside on attached hardware. Vendors also issue certificate revocation lists (CRLs). Along with installing the keys and CAs, you are responsible for installing the CRL in the `/etc/inet/ike/crls` directory.

CAs have the advantage of being certified by an outside organization, rather than by the site administrator. In a sense, CAs are notarized certificates. As with self-signed certificates, CAs can be used on a mobile machine or on a system that might be renumbered. Unlike self-signed certificates, CAs can very easily scale to protect a large number of communicating systems.

# IKE and Hardware Acceleration

IKE algorithms are computationally expensive, particularly in the Phase 1 exchange. Systems that handle a large number of exchanges can use a Sun Crypto Accelerator 1000 board to handle the public key operations. The Sun Crypto Accelerator 4000 board can also be used to handle expensive Phase 1 computations.

For information on how to configure IKE to offload its computations to the accelerator board, see "How to Configure IKE to Find the Sun Crypto Accelerator 1000 Board" on page 592. For information on how to store keys, see "How to Configure IKE to Find the Sun Crypto Accelerator 4000 Board" on page 593, and the `cryptoadm`(1M) man page.

# IKE and Hardware Storage

Public key certificates, private keys, and public keys can be stored on a Sun Crypto Accelerator 4000 board. For RSA encryption, the board supports keys up to 2048 bits. For DSA encryption, the board supports keys up to 1024 bits.

For information on how to configure IKE to access the board, see "How to Configure IKE to Find the Sun Crypto Accelerator 1000 Board" on page 592. For information on how to add certificates and public keys to the board, see "How to Generate and Store Public Key Certificates on Hardware" on page 577.

# IKE Utilities and Files

The following table summarizes the configuration files for IKE policy, the storage locations for IKE keys, and the various commands that implement IKE.

**TABLE 22–2** IKE Configuration Files, Key Storage Locations, and Commands

| File, Command, or Location | Description | For More Information |
|---|---|---|
| `/usr/lib/inet/in.iked` daemon | Internet Key Exchange (IKE) daemon. Activates automated key management. | `in.iked`(1M) |
| `/usr/sbin/ikeadm` command | IKE administration command for viewing and modifying the IKE policy. | `ikeadm`(1M) |
| `/usr/sbin/ikecert` command | Certificate database management command for manipulating local databases that hold public key certificates. The databases can also be stored on an attached Sun Crypto Accelerator 4000 board. | `ikecert`(1M) |
| `/etc/inet/ike/config` file | Configuration file for the IKE policy in the `/etc/inet` directory. Contains the site's rules for matching inbound IKE requests and preparing outbound IKE requests. If this file exists, the `in.iked` daemon starts automatically at boot time. | `ike.config`(4) |
| `ike.preshared` file | Preshared keys file in the `/etc/inet/secret` directory. Contains secret keying material for authentication in the Phase 1 exchange. Used when configuring IKE with preshared keys. | `ike.preshared`(4) |
| `ike.privatekeys` directory | Private keys directory in the `/etc/inet/secret` directory. Contains the private keys that are part of a public-private key pair. | `ikecert`(1M) |
| `publickeys` directory | Directory in the `/etc/inet/ike` directory that holds public keys and certificate files. Contains the public key part of a public-private key pair. | `ikecert`(1M) |
| `crls` directory | Directory in the `/etc/inet/ike` directory that holds revocation lists for public keys and certificate files. | `ikecert`(1M) |
| Sun Crypto Accelerator 1000 board | Hardware that accelerates public key operations by offloading the operations from the operating system. | `ikecert`(1M) |

**TABLE 22–2** IKE Configuration Files, Key Storage Locations, and Commands *(Continued)*

| File, Command, or Location | Description | For More Information |
|---|---|---|
| Sun Crypto Accelerator 4000 board | Hardware that accelerates public key operations by offloading the operations from the operating system. The board also stores public keys, private keys, and public key certificates. | `ikecert`(1M) |

# Changes to IKE for the Solaris 10 Release

Since the Solaris 9 release, IKE includes the following functionality:

- IKE can be used to automate key exchange for IPsec over IPv6 networks. For more information, see "Key Management With IKE" on page 546.

  **Note –** IKE cannot be used to manage keys for IPsec in a non-global zone.

- Public key operations in IKE can be accelerated by a Sun Crypto Accelerator 1000 board or a Sun Crypto Accelerator 4000 board. The operations are offloaded to the board. The offloading accelerates encryption, thereby reducing demands on operating system resources. For more information, see "IKE and Hardware Acceleration" on page 549. For procedures, see "Configuring IKE to Find Attached Hardware (Task Map)" on page 591.
- Public key certificates, private keys, and public keys can be stored on a Sun Crypto Accelerator 4000 board. For more information on key storage, see "IKE and Hardware Storage" on page 549.
- IKE can be used to automate key exchange for IPsec from behind a NAT box. The traffic must use an IPv4 network. Also, the NAT-traversing IPsec ESP keys cannot be accelerated by hardware. For more information, see "IPsec and NAT Traversal" on page 490. For procedures, see "Configuring IKE for Mobile Systems (Task Map)" on page 583.
- Retransmission parameters and packet time out parameters have been added to the `/etc/inet/ike/config` file. These parameters tune the IKE Phase 1 (Main Mode) negotiation to handle network interference, heavy network traffic, and interoperation with platforms that have different implementations of the IKE protocol. For details about the parameters, see the `ike.config`(4) man page. For procedures, see "Changing IKE Transmission Parameters (Task Map)" on page 594.

◆ ◆ ◆ CHAPTER 23

# Configuring IKE (Tasks)

This chapter describes how to configure IKE for your systems. After IKE is configured, it automatically generates keying material for IPsec on your network. This chapter contains the following information:


- "Configuring IKE (Task Map)" on page 553
- "Configuring IKE With Preshared Keys (Task Map)" on page 554
- "Configuring IKE With Public Key Certificates (Task Map)" on page 564
- "Configuring IKE for Mobile Systems (Task Map)" on page 583
- "Configuring IKE to Find Attached Hardware (Task Map)" on page 591
- "Changing IKE Transmission Parameters (Task Map)" on page 594


For overview information about IKE, see Chapter 22, "Internet Key Exchange (Overview)." For reference information about IKE, see Chapter 24, "Internet Key Exchange (Reference)." For more procedures, see the Examples sections of the `ikeadm`(1M), `ikecert`(1M), and `ike.config`(4) man pages.

## Configuring IKE (Task Map)

You can use preshared keys, self-signed certificates, and certificates from a Certificate Authority (CA) to authenticate IKE. A rule links the particular IKE authentication method with the end points that are being protected. Therefore, you can use one or all IKE authentication methods on a system. A pointer to a PKCS #11 library enables certificates to use an attached hardware accelerator.

After configuring IKE, complete the IPsec task that uses the IKE configuration. The following table refers you to task maps that focus on a specific IKE configuration.

| Task | Description | For Instructions |
|---|---|---|
| Configure IKE with preshared keys | Protects communications between two systems by having the systems share a secret key. | "Configuring IKE With Preshared Keys (Task Map)" on page 554 |
| Configure IKE with public key certificates | Protects communications with public key certificates. The certificates can be self-signed, or they can be vouched for by a PKI organization. | "Configuring IKE With Public Key Certificates (Task Map)" on page 564 |
| Cross a NAT boundary | Configures IPsec and IKE to communicate with a mobile system | "Configuring IKE for Mobile Systems (Task Map)" on page 583 |
| Configure IKE to generate and store public key certificates on attached hardware | Enables a Sun Crypto Accelerator 1000 board or a Sun Crypto Accelerator 4000 board to accelerate IKE operations. Also enables the Sun Crypto Accelerator 4000 board to store public key certificates. | "Configuring IKE to Find Attached Hardware (Task Map)" on page 591 |
| Tune Phase 1 key negotiation parameters | Changes the timing of IKE key negotiations. | "Changing IKE Transmission Parameters (Task Map)" on page 594 |

# Configuring IKE With Preshared Keys (Task Map)

The following table points to procedures to configure and maintain IKE with preshared keys.

| Task | Description | For Instructions |
|---|---|---|
| Configure IKE with preshared keys | Creates an IKE policy file and one key to be shared. | "How to Configure IKE With Preshared Keys" on page 555 |
| Refresh preshared keys on a running IKE system | Adds fresh keying material for IKE on communicating systems. | "How to Refresh IKE Preshared Keys" on page 558 |
| Add preshared keys to a running IKE system | Adds a new IKE policy entry and new keying material to a system that is currently enforcing IKE policy. | "How to Add an IKE Preshared Key for a New Policy Entry in `ipsecinit.conf`" on page 559 |
| Check that preshared keys are identical | Displays the preshared keys on both systems to see that the keys are identical. | "How to Verify That IKE Preshared Keys Are Identical" on page 563 |

# Configuring IKE With Preshared Keys

Preshared keys is the simplest authentication method for IKE. If you are configuring two systems to use IKE, and you are the administrator for both of the systems, using preshared keys is a good choice. However, unlike public key certificates, preshared keys are tied to particular IP addresses. Preshared keys cannot be used with mobile systems or systems that might be renumbered. Also, when you use preshared keys, you cannot offload IKE computations to attached hardware.

# How to Configure IKE With Preshared Keys

The IKE implementation offers algorithms whose keys vary in length. The key length that you choose is determined by site security. In general, longer keys provide more security than shorter keys.

These procedures use the system names `enigma` and `partym`. Substitute the names of your systems for the names `enigma` and `partym`.

1 **On the system console, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

---

2 **On each system, copy the file `/etc/inet/ike/config.sample` to the file `/etc/inet/ike/config`.**

3 **Enter rules and global parameters in the `ike/config` file on each system.**

The rules and global parameters in this file should permit the IPsec policy in the system's `ipsecinit.conf` file to succeed. The following `ike/config` examples work with the `ipsecinit.conf` examples in "How to Secure Traffic Between Two Systems With IPsec" on page 496.

a. **For example, modify the `/etc/inet/ike/config` file on the `enigma` system:**

```
### ike/config file on enigma, 192.168.116.16

## Global parameters
#
## Phase 1 transform defaults
p1_lifetime_secs 14400
p1_nonce_len 40
#
## Defaults that individual rules can override.
p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha encr_alg des }
p2_pfs 2
#
## The rule to communicate with partym
#  Label must be unique
{ label "enigma-partym"
  local_addr 192.168.116.16
```

```
  remote_addr 192.168.13.213
  p1_xform
   { auth_method preshared oakley_group 5 auth_alg md5 encr_alg 3des }
  p2_pfs 5
}
```

**Note –** All arguments to the `auth_method` parameter must be on the same line.

**b. Modify the `/etc/inet/ike/config` file on the `partym` system:**

```
### ike/config file on partym, 192.168.13.213
## Global Parameters
#
p1_lifetime_secs 14400
p1_nonce_len 40
#
p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha encr_alg des }
p2_pfs 2

## The rule to communicate with enigma
#  Label must be unique
{ label "partym-enigma"
  local_addr 192.168.13.213
  remote_addr 192.168.116.16
p1_xform
   { auth_method preshared oakley_group 5 auth_alg md5 encr_alg 3des }
p2_pfs 5
}
```

**4 On each system, check the validity of the file.**

```
# /usr/lib/inet/in.iked -c -f /etc/inet/ike/config
```

**5 Generate random numbers for use as keying material.**

If your site has a random number generator, use that generator. On a Solaris system, you can use the `od` command. For example, the following command prints two lines of hexadecimal numbers:

```
% od -X -A n /dev/random | head -2
         f47cb0f4 32e14480 951095f8 2b735ba8
         0a9467d0 8f92c880 68b6a40e 0efe067d
```

For an explanation of the `od` command, see "How to Generate Random Numbers on a Solaris System" on page 502 and the `od`(1) man page.

**Note –** Other operating systems can require ASCII keying material. To generate the identical key in hexadecimal and ASCII formats, see Example 23–1.

**6 From the output of Step 5, construct one key.**

```
f47cb0f432e14480951095f82b735ba80a9467d08f92c88068b6a40e
```

The authentication algorithm in this procedure is MD5, as shown in Step 3. The size of the hash, that is, the size of the authentication algorithm's output, determines the minimum recommended size of a preshared key. The output of the MD5 algorithm is 128 bits, or 32 characters. The example key is 56 characters long, which provides additional keying material for IKE to use.

**7 Create the file `/etc/inet/secret/ike.preshared` on each system.**

Put the preshared key in each file.

**a. For example, on the `enigma` system, the `ike.preshared` file would appear similar to the following:**

```
# ike.preshared on enigma, 192.168.116.16
#...
{ localidtype IP
    localid 192.168.116.16
    remoteidtype IP
    remoteid 192.168.13.213
    # enigma and partym's shared key in hex (192 bits)
    key f47cb0f432e14480951095f82b735ba80a9467d08f92c88068b6a40e
    }
```

**b. On the `partym` system, the `ike.preshared` file would appear similar to the following:**

```
# ike.preshared on partym, 192.168.13.213
#...
{ localidtype IP
    localid 192.168.13.213
    remoteidtype IP
    remoteid 192.168.116.16
    # partym and enigma's shared key in hex (192 bits)
    key f47cb0f432e14480951095f82b735ba80a9467d08f92c88068b6a40e
    }
```

**Note –** The preshared keys on each system must be identical.

**Example 23–1** Generating Identical Keying Material for Two Systems With Different Operating Systems

Solaris IPsec interoperates with other operating systems. If your system is communicating with a system that requires ASCII preshared keys, you need to generate one key in two formats, hexadecimal and ASCII.

In this example, the Solaris system administrator wants 56 characters of keying material. The administrator uses the following command to generate a hexadecimal key from an ASCII passphrase. The option `-tx1` prints the bytes one at a time on all Solaris systems.

```
# /bin/echo "papiermache with cashews and\c" | od -tx1 | cut -c 8-55 | \
tr -d '\n' | tr -d ' ' | awk '{print}'
706170696572 6d6163686520776974682063617368657773 20616e64
```

By removing the offsets and concatenating the hexadecimal output, the hexadecimal key for the Solaris system is `706170696572 6d6163686520776974682063617368657773 20616e64`. The administrator places this value in the `ike.preshared` file on the Solaris system.

```
# Shared key in hex (192 bits)
key 706170696572 6d6163686520776974682063617368657773 20616e64
```

On the system that requires ASCII preshared keys, the passphrase is the preshared key. The Solaris system administrator telephones the other administrator with the passphrase, `papiermache with cashews and`.

## ▼ How to Refresh IKE Preshared Keys

This procedure assumes that you want to replace an existing preshared key at regular intervals without rebooting. If you use a strong encryption algorithm, such 3DES or Blowfish, you might want to refresh keys just before you reboot both systems.

1 **On the system console, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session. Use the `ssh` command for a secure remote login.

---

2. **Generate random numbers and construct a key of the appropriate length.**

   For details, see "How to Generate Random Numbers on a Solaris System" on page 502. If you are generating a preshared key for a Solaris system that is communicating with an operating system that requires ASCII, see Example 23–1.

3. **Replace the current key with a new key.**

   For example, on the hosts `enigma` and `partym`, you would replace the value of `key` in the `/etc/inet/secret/ike.preshared` file with a new number of the same length.

4. **Check the privilege level of the `in.iked` daemon.**

   ```
   # /usr/sbin/ikeadm get priv
   Current privilege level is 0x0, base privileges enabled
   ```

   You can change the keying material if the command returns a privilege level of `0x1` or `0x2`. Level `0x0` does not permit operations to modify or view keying material. By default, the `in.iked` daemon runs at the `0x0` level of privilege.

   - **If the privilege level is `0x0`, kill and restart the daemon.**

     When the daemon restarts, it reads the new version of the `ike.preshared` file.

     ```
     # pkill in.iked
     # /usr/lib/inet/in.iked
     ```

   - **If the privilege level is `0x1` or `0x2`, read in the new version of the `ike.preshared` file.**

     ```
     # ikeadm read preshared
     ```

## How to Add an IKE Preshared Key for a New Policy Entry in `ipsecinit.conf`

You must have one preshared key for every policy entry in the `ipsecinit.conf` file. If you add a new policy entry while IPsec and IKE are running, the `in.iked` daemon can read in new keys.

**Before You Begin** This procedure assumes the following:

- The `enigma` system is set up as described in "How to Configure IKE With Preshared Keys" on page 555.
- The `enigma` system is going to protect its traffic with a new system, `ada`.
- The `in.iked` daemon is running on both systems.
- The systems' interfaces are included as entries in the `/etc/hosts` file on both systems. The following entry is an example.

  ```
  192.168.15.7 ada
  192.168.116.16 enigma
  ```

This procedure also works with an IPv6 address in the `/etc/inet/ipnodes` file. In the Solaris 10 5/07 release, IPv6 entries are placed in the `/etc/hosts` file.

- You have added a new policy entry to the `/etc/inet/ipsecinit.conf` file on both systems. The entries appear similar to the following:

```
# ipsecinit.conf file for enigma
{laddr enigma raddr ada} ipsec {auth_algs any encr_algs any sa shared}

# ipsecinit.conf file for ada
{laddr ada raddr enigma} ipsec {auth_algs any encr_algs any sa shared}
```

**1 On the system console, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

---

**2 Create a rule for IKE to manage the keys for `enigma` and `ada`.**

**a. For example, the rule in the `/etc/inet/ike/config` file on the `enigma` system appears similar to the following:**

```
### ike/config file on enigma, 192.168.116.16

## The rule to communicate with ada

{label "enigma-to-ada"
 local_addr 192.168.116.16
 remote_addr 192.168.15.7
 p1_xform
 {auth_method preshared oakley_group 5 auth_alg md5 encr_alg blowfish}
 p2_pfs 5
    }
```

**b. The rule in the `/etc/inet/ike/config` file on the `ada` system appears similar to the following:**

```
### ike/config file on ada, 192.168.15.7

## The rule to communicate with enigma

{label "ada-to-enigma"
 local_addr 192.168.15.7
```

```
 remote_addr 192.168.116.16
 p1_xform
 {auth_method preshared oakley_group 5 auth_alg md5 encr_alg blowfish}
 p2_pfs 5
}
```

**3 Check the privilege level of the `in.iked` daemon.**

```
# /usr/sbin/ikeadm get priv
Current privilege level is 0x0, base privileges enabled
```

- **If the privilege level is `0x1` or `0x2`, continue with the next step.**
- **If the privilege level is `0x0`, stop the daemon. Then, restart the daemon with the appropriate privilege level.**

  ```
  # pkill in.iked
  # /usr/lib/inet/in.iked -p 2
  Setting privilege level to 2!
  ```

**4 Generate random numbers and construct a key of 64 to 448 bits.**

For details, see "How to Generate Random Numbers on a Solaris System" on page 502. If you are generating a preshared key for a Solaris system that is communicating with an operating system that requires ASCII, see Example 23–1.

**5 By some means, send the key to the administrator of the remote system.**

You both need to add the same preshared key at the same time. Your key is only as safe as the safety of your transmission mechanism. An out-of-band mechanism, such as registered mail or a protected fax machine, is best.

**6 Add the new keying material with the `add preshared` subcommand in `ikeadm` command mode.**

```
ikeadm> add preshared { localidtype id-type localid id
remoteidtype id-type remoteid id ike_mode mode key key }
```

*id-type* Specifies the type of the *id*.

*id* Specifies the IP address when *id-type* is IP.

*mode* Specifies the IKE mode. The only accepted value is `main`.

*key* Specifies the preshared key in hexadecimal format.

**a. For example, on host `enigma`, you would add the key for the remote system, `ada`, and exit the `ikeadm` interactive mode.**

```
# ikeadm
ikeadm> add preshared { localidtype ip localid 192.168.116.16
remoteidtype ip remoteid 192.168.15.7
```

```
key 8d1fb4ee500e2bea071deb2e781cb48374411af5a9671714672bb1749ad9364d }
ikeadm: Successfully created new preshared key.
ikeadm> exit
#
```

**b. On host `ada`, you would add the identical key for the remote system, `enigma`, and exit the interactive mode.**

```
# ikeadm
ikeadm> add preshared { localidtype ip localid 192.168.15.7
remoteidtype ip remoteid 192.168.116.16

key 8d1fb4ee500e2bea071deb2e781cb48374411af5a9671714672bb1749ad9364d }
ikeadm: Successfully created new preshared key.
ikeadm> exit
#
```

**7 On each system, lower the privilege level of the `in.iked` daemon.**

```
# ikeadm set priv base
```

**8 On each system, activate the `ipsecinit.conf` file to secure the added interface.**

```
# ipsecconf -a /etc/inet/ipsecinit.conf
```

**Caution –** Read the warning when you execute the `ipsecconf` command. The same warning applies to restarting the `in.iked` daemon. A socket that is already latched, that is, the socket is in use, provides an unsecured back door into the system. For more extensive discussion, see "Security Considerations for `ipsecinit.conf` and `ipsecconf`" on page 539.

**9 On each system, read in the new rules by using the `ikeadm` command.**

```
# ikeadm read rules
```

You created the rules in Step 2. Because the rules are in the `/etc/inet/ike/config` file, the name of the file does not have to be specified to the `ikeadm` command.

**10 Ensure that IKE preshared keys are available at reboot.**

Add the keys to the `/etc/inet/secret/ike.preshared` file.

**a. For example, on the `enigma` system, you would add the following keying information to the `ike.preshared` file:**

```
# ike.preshared on enigma for the ada interface
#
{ localidtype IP
  localid 192.168.116.16
  remoteidtype IP
```

```
  remoteid 192.168.15.7
  # enigma and ada's shared key in hex (32 - 448 bits required)
  key 8d1fb4ee500e2bea071deb2e781cb48374411af5a9671714672bb1749ad9364d
}
```

**b. On the `ada` system, you would add the following keying information to the `ike.preshared` file:**

```
# ike.preshared on ada for the enigma interface
#
{ localidtype IP
  localid 192.168.15.7
  remoteidtype IP
  remoteid 192.168.116.16
  # ada and enigma's shared key in hex (32 - 448 bits required)
  key 8d1fb4ee500e2bea071deb2e781cb48374411af5a9671714672bb1749ad9364d
}
```

**11 Verify that the systems can communicate.**

See "How to Verify That IKE Preshared Keys Are Identical" on page 563.

## ▼ How to Verify That IKE Preshared Keys Are Identical

If the preshared keys on the communicating systems are not identical, the systems cannot authenticate.

**Before You Begin** IPsec has been configured and is enabled between the two systems that you are testing.

**1 On the system console, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

---

**2 Check the privilege level of the `in.iked` daemon.**

```
# /usr/sbin/ikeadm get priv
Current privilege level is 0x0, base privileges enabled
```

- **If the privilege level is `0x1` or `0x2`, continue with the next step.**

- **If the privilege level is `0x0`, stop the daemon. Then, restart the daemon with the appropriate privilege level.**

```
# pkill in.iked
# /usr/lib/inet/in.iked -p 2
Setting privilege level to 2!
```

3. **On each system, view the preshared key information.**

```
# ikeadm dump preshared
PSKEY: Preshared key (24 bytes): f47cb.../192
LOCIP: AF_INET: port 0, 192.168.116.16 (enigma).
REMIP: AF_INET: port 0, 192.168.13.213 (partym).
```

4. **Compare the two dumps.**

   If the preshared keys are not identical, replace one key with the other key in the `/etc/inet/secret/ike.preshared` file.

5. **When the verification is complete, lower the privilege level of the `in.iked` daemon on each system.**

```
# ikeadm set priv base
```

# Configuring IKE With Public Key Certificates (Task Map)

The following table provides pointers to procedures for creating public key certificates for IKE. The procedures include how to accelerate and store the certificates on attached hardware.

| Task | Description | For Instructions |
|---|---|---|
| Configure IKE with self-signed public key certificates | Creates and places two certificates on each system:<br>■ A self-signed certificate<br>■ The public key certificate from the remote system | "How to Configure IKE With Self-Signed Public Key Certificates" on page 565 |
| Configure IKE with a PKI Certificate Authority | Creates a certificate request, and then places three certificates on each system:<br>■ The certificate that the Certificate Authority (CA) creates from your request<br>■ The public key certificate from the CA<br>■ The CRL from the CA | "How to Configure IKE With Certificates Signed by a CA" on page 571 |

| Task | Description | For Instructions |
|---|---|---|
| Configure public key certificates on local hardware | Involves one of:<br>■ Generating a self-signed certificate on the local hardware and then adding the public key from a remote system to the hardware.<br>■ Generating a certificate request on the local hardware and then adding the public key certificates from the CA to the hardware. | "How to Generate and Store Public Key Certificates on Hardware" on page 577 |
| Update the certificate revocation list (CRL) from a PKI | Accesses the CRL from a central distribution point. | "How to Handle a Certificate Revocation List" on page 581 |

# Configuring IKE With Public Key Certificates

Public key certificates eliminate the need for communicating systems to share secret keying material out of band. Unlike preshared keys, a public key certificate can be used on a mobile machine or on a system that might be renumbered.

Public key certificates can also be stored on attached hardware. For the procedure, see "Configuring IKE to Find Attached Hardware (Task Map)" on page 591.

## ▼ How to Configure IKE With Self-Signed Public Key Certificates

Self-signed certificates require less overhead than public certificates from a CA, but do not scale very easily.

**1 On the system console, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

---

**2 Add a self-signed certificate to the `ike.privatekeys` database.**

```
# ikecert certlocal -ks|-kc -m keysize -t keytype \
-D dname -A altname \
[-S validity-start-time] [-F validity-end-time] [-T token-ID]
```

| | |
|---|---|
| `-ks` | Creates a self-signed certificate. |
| `-kc` | Creates a certificate request. For the procedure, see "How to Configure IKE With Certificates Signed by a CA" on page 571. |
| `-m` *keysize* | Is the size of the key. The *keysize* can be 512, 1024, 2048, 3072, or 4096. |
| `-t` *keytype* | Specifies the type of algorithm to use. The *keytype* can be `rsa-sha1`, `rsa-md5`, or `dsa-sha1`. |
| `-D` *dname* | Is the X.509 distinguished name for the certificate subject. The *dname* typically has the form: C=country, O=organization, OU=organizational unit, CN=common name. Valid tags are C, O, OU, and CN. |
| `-A` *altname* | Is the alternate name for the certificate. The *altname* is in the form of `tag=value`. Valid tags are `IP`, `DNS`, `email`, and `DN`. |
| `-S` *validity-start-time* | Provides an absolute or relative valid start time for the certificate. |
| `-F` *validity-end-time* | Provides an absolute or relative valid end time for the certificate. |
| `-T` *token-ID* | Enables a PKCS #11 hardware token to generate the keys. The certificates are then stored in the hardware. |

**a. For example, the command on the `partym` system would appear similar to the following:**

```
# ikecert certlocal -ks -m 1024 -t rsa-md5 \
-D "C=US, O=PartyCompany, OU=US-Partym, CN=Partym" \
-A IP=192.168.13.213
Creating software private keys.
  Writing private key to file /etc/inet/secret/ike.privatekeys/0.
Enabling external key providers - done.
Acquiring private keys for signing - done.
Certificate:
 Proceeding with the signing operation.
 Certificate generated successfully (.../publickeys/0)
Finished successfully.
Certificate added to database.
-----BEGIN X509 CERTIFICATE-----
MIICLTCCAZagAwIBAgIBATANBgkqhkiG9w0BAQQFADBNMQswCQYDVQQGEwJVUzEX
...
6sKTxpg4GP3GkQGcd0r1rhW/3yaWBkDwOdFCqEUyffzU
-----END X509 CERTIFICATE-----
```

**b. The command on the `enigma` system would appear similar to the following:**

```
# ikecert certlocal -ks -m 1024 -t rsa-md5 \
-D "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax" \
-A IP=192.168.116.16
Creating software private keys.
```

```
  ...
Certificate added to database.
-----BEGIN X509 CERTIFICATE-----
MIICKDCCAZGgAwIBAgIBATANBgkqhkiG9w0BAQQFADBJMQswCQYDVQQGEwJVUzEV
...
jpxfLM98xyFVyLCbkr3dZ3Tvxvi732BXePKF2A==
-----END X509 CERTIFICATE-----
```

**3 Save the certificate and send it to the remote system.**

You can paste the certificate into an email.

**a. For example, you would send the following `partym` certificate to the `enigma` administrator:**

```
To: admin@ja.enigmaexample.com
From: admin@us.partyexample.com
Message: -----BEGIN X509 CERTIFICATE-----
MIICLTCCAZagAwIBAgIBATANBgkqhkiG9w0BAQQFADBNMQswCQYDVQQGEwJVUzEX
...
6sKTxpg4GP3GkQGcd0r1rhW/3yaWBkDwOdFCqEUyffzU
-----END X509 CERTIFICATE-----
```

**b. The `enigma` administrator would send you the following `enigma` certificate:**

```
To: admin@us.partyexample.com
From: admin@ja.enigmaexample.com
Message: -----BEGIN X509 CERTIFICATE-----
MIICKDCCAZGgAwIBAgIBATANBgkqhkiG9w0BAQQFADBJMQswCQYDVQQGEwJVUzEV
...
jpxfLM98xyFVyLCbkr3dZ3Tvxvi732BXePKF2A==
-----END X509 CERTIFICATE-----
```

**4 On each system, add the certificate that you received.**

**a. Copy the public key from the administrator's email.**

**b. Type the `ikecert certdb -a` command and press the Return key.**

No prompts display when you press the Return key.

```
# ikecert certdb -a       Press the Return key
```

**c. Paste the public key. Then press the Return key. To end the entry, press Control-D.**

```
-----BEGIN X509 CERTIFICATE-----
MIIC...
...
----END X509 CERTIFICATE-----       Press the Return key
<Control>-D
```

**5 Verify with the other administrator that the certificate is from that administrator.**

For example, you can telephone the other administrator to compare the values of the public key hash. The public key hash for the shared certificate must be identical on the two systems.

**a. List the stored certificate on your system.**

For example, on the `partym` system, the public certificate is in slot 1, and the private certificate is in slot 0.

```
partym # ikecert certdb -l
Certificate Slot Name: 0   Type: rsa-md5      Private Key
    Subject Name: <C=US, O=PartyCompany, OU=US-Partym, CN=Partym>
    Key Size: 1024
    Public key hash: B2BD13FCE95FD27ECE6D2DCD0DE760E2

Certificate Slot Name: 1   Type: rsa-md5      Public Certificate
    (Private key in certlocal slot 0)      Points to certificate's private key
    Subject Name: <C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax>
    Key Size: 1024
    Public key hash: 2239A6A127F88EE0CB40F7C24A65B818
```

**b. Compare this value with the public key hash on the `enigma` system.**

You can read the public key hash over the telephone.

```
enigma # ikecert certdb -l
Certificate Slot Name: 4   Type: rsa-md5      Private Key
    Subject Name: <C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax>
    Key Size: 1024
    Public key hash: DF3F108F6AC669C88C6BD026B0FCE3A0

Certificate Slot Name: 5   Type: rsa-md5      Public Certificate
    (Private key in certlocal slot 4)
    Subject Name: <C=US, O=PartyCompany, OU=US-Partym, CN=Partym>
    Key Size: 1024
    Public key hash: 2239A6A127F88EE0CB40F7C24A65B818
```

**6 On each system, trust both certificates.**

Edit the `/etc/inet/ike/config` file to recognize the certificates.

The administrator of the remote system provides the values for the `cert_trust`, `remote_addr`, and `remote_id` parameters.

**a. For example, on the `partym` system, the `ike/config` file would appear similar to the following:**

```
# Explicitly trust the following self-signed certs
# Use the Subject Alternate Name to identify the cert
```

```
# Verified remote address and remote ID
# Verified public key hash per telephone call from administrator
cert_trust "192.168.13.213"      Local system's certificate Subject Alt Name
cert_trust "192.168.116.16"      Remote system's certificate Subject Alt Name

## Parameters that may also show up in rules.

p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha encr_alg des }
p2_pfs 5

{
 label "US-partym to JA-enigmax"
 local_id_type dn
 local_id "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"
 remote_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"

 local_addr  192.168.13.213
 remote_addr 192.168.116.16

 p1_xform
  {auth_method rsa_sig oakley_group 2 auth_alg md5 encr_alg 3des}
}
```

**b. On the `enigma` system, add `enigma` values for local parameters in the `ike/config` file.**

For the remote parameters, use `partym` values. Ensure that the value for the `label` keyword is unique. This value must be different from the remote system's `label` value.

```
...
{
 label "JA-enigmax to US-partym"
 local_id_type dn
 local_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
 remote_id "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"

 local_addr  192.168.116.16
 remote_addr 192.168.13.213
...
```

**Example 23–2** Verifying That a Certificate From Another Administrator is Valid

In this example, the administrators use the Subject Name to verify that the certificates are identical.

The first administrator saves the output of generating and listing the certificate to a file. Because the output of the `ikecert` command prints to standard error, the administrator redirects standard error to the file.

```
sys1# cd /
sys1# ikecert certlocal -ks -m1024 -trsa-md5 \
-D"C=US, O=TestCo, CN=Co2Sys" 2>/tmp/for_co2sys
Certificate added to database.
sys1# ikecert certdb -l "C=US, O=TestCo, CN=Co2Sys" 2>>/tmp/for_co2sys
```

The administrator verifies the contents of the file.

```
sys1# cat /tmp/for_co2sys
Creating private key.
-----BEGIN X509 CERTIFICATE-----
MIIB7TCCAVagAwIBAgIEZkHfOTANBgkqhkiG9w0BAQQFADAxMQwwCgYDVQQGEwNV
U0ExEDAOBgNVBAoMB3Rlc3RfY28xDzANBgNVBAMTBkVuaWdtYTAeFw0wODAxMTUx
OTI1MjBaFw0xMjAxMTUxOTI1MjBaMDExDDAKBgNVBAYTA1VTQTEQMA4GA1UECgwH
dGVzdF9jbzEPMA0GA1UEAxMGRW5pZ21hMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCB
iQKBgQCPxGv0rUzHMnFtkx9uwYuPiWbftmWfa9iDt6ELOEuw3zlboy2qtuRUZohz
FIbCxAJevdCY6a+pktvYy3/2nJL0WATObO5T0FKn3F0bphajinLYbyCrYhEzD9E2
gkiT2D9/ttbSiMvi9usphprEDcLAFaWgCJiHnKPBEkjC0vhA3wIDAQABoxIwEDAO
BgNVHQ8BAf8EBAMCBaAwDQYJKoZIhvcNAQEEBQADgYEAL/q6xgweylGQylqLCwzN
5PIpjfzsNPf3saTyh3VplwEOW6WTHwRQT17IO/1Oc6Jnz9Mr0ZrbHWDXq+1sx180
F8+DMW1Qv1UR/lGMq3ufDG3qedmSN6txDF8qLlPCUML0YL8m4oGdewqGb+78aPyE
Y/cJRsK1hWbYyseqcIkjj5k=
-----END X509 CERTIFICATE-----
Certificate Slot Name: 2   Key Type: rsa
        (Private key in certlocal slot 2)
        Subject Name: <C=US, O=TestCo, CN=Co2Sys>
        Key Size: 1024
        Public key hash: C46DE77EF09084CE2B7D9C70479D77FF
```

Then, the administrator sends the file in an email to the second administrator.

The second administrator places the file in a secure directory, then imports the certificate from the file.

```
sys2# cd /
sys2# ikecert certdb -a < /sec/co2sys
```

The `ikecert` command imports only the text between the `-----BEGIN` and `-----END` lines. The administrator verifies that the local certificate has the same public key hash as the public key hash in the `co2sys` file.

```
sys2# ikecert certdb -l
Certificate Slot Name: 1   Key Type: rsa
        (Private key in certlocal slot 1)
        Subject Name: <C=US, O=TestCo, CN=Co2Sys>
        Key Size: 1024
        Public key hash: C46DE77EF09084CE2B7D9C70479D77FF
```

To ensure that the first administrator sent this email, the second administrator telephones the first administrator to verify the Subject Name of the certificate.

**Example 23–3** Specifying a Start Time and an End Time for a Certificate

In this example, the administrator on the `partym` system establishes dates within which the certificate is valid. The certificate is backdated by 2 1/2 days, and is valid for 4 years and 6 months from the date of creation.

```
# ikecert certlocal -ks -m 1024 -t rsa-md5 \
-D "C=US, O=PartyCompany, OU=US-Partym, CN=Partym" \
-A IP=192.168.13.213 \
-S -2d12h -F +4y6m
```

The administrator on the `enigma` system establishes dates within which the certificate is valid. The certificate is backdated by 2 days and is valid until midnight of December 31, 2010.

```
# ikecert certlocal -ks -m 1024 -t rsa-md5 \
-D "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax" \
-A IP=192.168.116.16 \
-S -2d -F "12/31/2010 12:00 AM"
```

## ▼ How to Configure IKE With Certificates Signed by a CA

Public certificates from a Certificate Authority (CA) require negotiation with an outside organization. The certificates very easily scale to protect a large number of communicating systems.

**1 On the system console, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

---

**2 Use the `ikecert certlocal -kc` command to create a certificate request.**

For a description of the arguments to the command, see Step 2 in "How to Configure IKE With Self-Signed Public Key Certificates" on page 565.

```
# ikecert certlocal -kc -m keysize -t keytype \
-D dname -A altname
```

**a. For example, the following command creates a certificate request on the `partym` system:**

```
# ikecert certlocal -kc -m 1024 -t rsa-md5 \
> -D "C=US, O=PartyCompany\, Inc., OU=US-Partym, CN=Partym" \
> -A "DN=C=US, O=PartyCompany\, Inc., OU=US-Partym"
Creating software private keys.
  Writing private key to file /etc/inet/secret/ike.privatekeys/2.
Enabling external key providers - done.
Certificate Request:
  Proceeding with the signing operation.
  Certificate request generated successfully (.../publickeys/0)
Finished successfully.
-----BEGIN CERTIFICATE REQUEST-----
MIIByjCCATMCAQAwUzELMAkGA1UEBhMCVVMxHTAbBgNVBAoTFEV4YW1wbGVDb21w
...
lcM+tw0ThRrfuJX9t/Qa1R/KxRlMA3zckO80mO9X
-----END CERTIFICATE REQUEST-----
```

**b. The following command creates a certificate request on the `enigma` system:**

```
# ikecert certlocal -kc -m 1024 -t rsa-md5 \
> -D "C=JA, O=EnigmaCo\, Inc., OU=JA-Enigmax, CN=Enigmax" \
> -A "DN=C=JA, O=EnigmaCo\, Inc., OU=JA-Enigmax"
Creating software private keys.
...
Finished successfully.
-----BEGIN CERTIFICATE REQUEST-----
MIIBuDCCASECAQAwSTELMAkGA1UEBhMCVVMxFTATBgNVBAoTDFBhcnR5Q29tcGFu
...
8qlqdjaStLGfhDOO
-----END CERTIFICATE REQUEST-----
```

**3 Submit the certificate request to a PKI organization.**

The PKI organization can tell you how to submit the certificate request. Most organizations have a web site with a submission form. The form requires proof that the submission is legitimate. Typically, you paste your certificate request into the form. When your request has been checked by the organization, the organization issues you the following two certificate objects and a list of revoked certificates:

- Your public key certificate – This certificate is based on the request that you submitted to the organization. The request that you submitted is part of this public key certificate. The certificate uniquely identifies you.
- A Certificate Authority – The organization's signature. The CA verifies that your public key certificate is legitimate.
- A Certificate Revocation List (CRL) – The latest list of certificates that the organization has revoked. The CRL is not sent separately as a certificate object if access to the CRL is embedded in the public key certificate.

  When a URI for the CRL is embedded in the public key certificate, IKE can automatically retrieve the CRL for you. Similarly, when a DN (directory name on an LDAP server) entry is embedded in the public key certificate, IKE can retrieve and cache the CRL from an LDAP server that you specify.

  See "How to Handle a Certificate Revocation List" on page 581 for an example of an embedded URI and an embedded DN entry in a public key certificate.

**4 Add each certificate to your system.**

The `-a` option to the `ikecert certdb -a` adds the pasted object to the appropriate certificate database on your system. For more information, see "IKE With Public Key Certificates" on page 548.

**a. On the system console, assume the Primary Administrator role or become superuser.**

**b. Add the public key certificate that you received from the PKI organization.**

```
# ikecert certdb -a
      Press the Return key
      Paste the certificate:
-----BEGIN X509 CERTIFICATE-----
...
-----END X509 CERTIFICATE----
      Press the Return key
<Control>-D
```

**c. Add the CA from the PKI organization.**

```
# ikecert certdb -a
      Press the Return key
      Paste the CA:
-----BEGIN X509 CERTIFICATE-----
```

```
...
-----END X509 CERTIFICATE----
```
*Press the Return key*
```
<Control>-D
```

**d. If the PKI organization has sent a list of revoked certificates, add the CRL to the `certrldb` database:**

```
# ikecert certrldb -a
```
*Press the Return key*
*Paste the CRL:*
```
-----BEGIN CRL-----
...
-----END CRL----
```
*Press the Return key*
```
<Control>-D
```

**5 Use the `cert_root` keyword to identify the PKI organization in the `/etc/inet/ike/config` file.**

Use the name that the PKI organization provides.

**a. For example, the `ike/config` file on the `partym` system might appear similar to the following:**

```
# Trusted root cert
# This certificate is from Example PKI
# This is the X.509 distinguished name for the CA that it issues.

cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example, CN=Example PKI"

## Parameters that may also show up in rules.

p1_xform
 { auth_method rsa_sig oakley_group 1 auth_alg sha1 encr_alg des }
p2_pfs 2

{
 label "US-partym to JA-enigmax - Example PKI"
 local_id_type dn
 local_id  "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"
 remote_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"

 local_addr  192.168.13.213
 remote_addr 192.168.116.16

 p1_xform
  {auth_method rsa_sig oakley_group 2 auth_alg md5 encr_alg 3des}
}
```

**Note –** All arguments to the `auth_method` parameter must be on the same line.

**b. On the `enigma` system, create a similar file.**

Specifically, the `enigma` `ike/config` file should do the following:

- Include the same `cert_root` value.
- Use `enigma` values for local parameters.
- Use `partym` values for remote parameters.
- Create a unique value for the `label` keyword. This value must be different from the remote system's `label` value.

```
...
cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example, CN=Example PKI"
...
{
 label "JA-enigmax to US-partym - Example PKI"
 local_id_type dn
 local_id   "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
 remote_id  "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"

 local_addr  192.168.116.16
 remote_addr 192.168.13.213
...
```

**6 Tell IKE how to handle CRLs.**

Choose the appropriate option:

- **No CRL available**

  If the PKI organization does not provide a CRL, add the keyword `ignore_crls` to the `ike/config` file.

  ```
  # Trusted root cert
  ...
  cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example,...
  ignore_crls
  ...
  ```

  The `ignore_crls` keyword tells IKE not to search for CRLs.

- **CRL available**

  If the PKI organization provides a central distribution point for CRLs, you can modify the `ike/config` file to point to that location.

  See "How to Handle a Certificate Revocation List" on page 581 for examples.

**Example 23–4** Using `rsa_encrypt` When Configuring IKE

When you use `auth_method rsa_encrypt` in the `ike/config` file, you must add the peer's certificate to the `publickeys` database.

1. Send the certificate to the remote system's administrator.

   You can paste the certificate into an email.

   For example, the `partym` administrator would send the following email:

   ```
   To: admin@ja.enigmaexample.com
   From: admin@us.partyexample.com
   Message: -----BEGIN X509 CERTIFICATE-----
   MII...
   ----END X509 CERTIFICATE-----
   ```

   The `enigma` administrator would send the following email:

   ```
   To: admin@us.partyexample.com
   From: admin@ja.enigmaexample.com
   Message: -----BEGIN X509 CERTIFICATE-----
   MII
   ...
   -----END X509 CERTIFICATE-----
   ```

2. On each system, add the emailed certificate to the local `publickeys` database.

   ```
   # ikecert certdb -a
        Press the Return key
   -----BEGIN X509 CERTIFICATE-----
   MII...
   -----END X509 CERTIFICATE-----
        Press the Return key
   <Control>-D
   ```

The authentication method for RSA encryption hides identities in IKE from eavesdroppers. Because the `rsa_encrypt` method hides the peer's identity, IKE cannot retrieve the peer's certificate. As a result, the `rsa_encrypt` method requires that the IKE peers know each other's public keys.

Therefore, when you use an `auth_method` of `rsa_encrypt` in the `/etc/inet/ike/config` file, you must add the peer's certificate to the `publickeys` database. The `publickeys` database then holds three certificates for each communicating pair of systems:

- Your public key certificate
- The CA certificate
- The peer's public key certificate

**Troubleshooting –** The IKE payload, which includes the three certificates, can become too large for `rsa_encrypt` to encrypt. Errors such as "authorization failed" and "malformed payload" can indicate that the `rsa_encrypt` method cannot encrypt the total payload. Reduce the size of the payload by using a method, such as `rsa_sig`, that requires only two certificates.

## How to Generate and Store Public Key Certificates on Hardware

Generating and storing public key certificates on hardware is similar to generating and storing public key certificates on your system. On hardware, the `ikecert certlocal` and `ikecert certdb` commands must identify the hardware. The `-T` option with the token ID identifies the hardware to the commands.

**Before You Begin**

- The hardware must be configured.
- The hardware uses the `/usr/lib/libpkcs11.so` library, unless the `pkcs11_path` keyword in the `/etc/inet/ike/config` file points to a different library. The library must be implemented according to the following standard: RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki), that is, a PKCS #11 library.

  See "How to Configure IKE to Find the Sun Crypto Accelerator 4000 Board" on page 593 for setup instructions.

**1 On the system console, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

---

**2 Generate a self-signed certificate or a certificate request, and specify the token ID.**

Choose one of the following options:

---

**Note –** The Sun Crypto Accelerator 4000 board supports keys up to 2048 bits for RSA. For DSA, this board supports keys up to 1024 bits.

---

- **For a self-signed certificate, use this syntax.**

```
# ikecert certlocal -ks -m 1024 -t rsa-md5 \
> -D "C=US, O=PartyCompany, OU=US-Partym, CN=Partym" \
> -a -T dca0-accel-stor IP=192.168.116.16
```

```
Creating hardware private keys.
Enter PIN for PKCS#11 token:      Type user:password
```

The argument to the `-T` option is the token ID from the attached Sun Crypto Accelerator 4000 board.

- **For a certificate request, use this syntax.**

```
# ikecert certlocal -kc -m 1024 -t rsa-md5 \
> -D "C=US, O=PartyCompany, OU=US-Partym, CN=Partym" \
> -a -T dca0-accel-stor IP=192.168.116.16
Creating hardware private keys.
Enter PIN for PKCS#11 token:      Type user:password
```

For a description of the arguments to the `ikecert` command, see the `ikecert`(1M) man page.

**3 At the prompt for a PIN, type the Sun Crypto Accelerator 4000 user, a colon, and the user's password.**

If the Sun Crypto Accelerator 4000 board has a user `ikemgr` whose password is `rgm4tigt`, you would type the following:

```
Enter PIN for PKCS#11 token: ikemgr:rgm4tigt
```

---

**Note –** The PIN response is stored on disk *as clear text*.

---

After you type the password, the certificate prints out:

```
Enter PIN for PKCS#11 token: ikemgr:rgm4tigt
-----BEGIN X509 CERTIFICATE-----
MIIBuDCCASECAQAwSTELMAkGA1UEBhMCVVMxFTATBgNVBAoTDFBhcnR5Q29tcGFu
...
oKUDBbZ9O/pLWYGr
-----END X509 CERTIFICATE-----
```

**4 Send your certificate for use by the other party.**

Choose one of the following options:

- **Send the self-signed certificate to the remote system.**

  You can paste the certificate into an email.

- **Send the certificate request to an organization that handles PKI.**

  Follow the instructions of the PKI organization to submit the certificate request. For a more detailed discussion, see Step 3 of "How to Configure IKE With Certificates Signed by a CA" on page 571.

5 **On your system, edit the `/etc/inet/ike/config` file to recognize the certificates.**

Choose one of the following options.

- **Self-signed certificate**

  Use the values that the administrator of the remote system provides for the `cert_trust`, `remote_id`, and `remote_addr` parameters. For example, on the `enigma` system, the `ike/config` file would appear similar to the following:

```
# Explicitly trust the following self-signed certs
# Use the Subject Alternate Name to identify the cert

cert_trust "192.168.116.16"       Local system's certificate Subject Alt Name
cert_trust "192.168.13.213"       Remote system's certificate Subject Alt name


# Solaris 10 1/06 release: default path does not have to be typed in
#pkcs11_path "/usr/lib/libpkcs11.so"      Hardware connection

# Solaris 10 release: use this path
#pkcs11_path "/opt/SUNWconn/cryptov2/lib/libvpkcs11.so"
...
{
 label "JA-enigmax to US-partym"
 local_id_type dn
 local_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
 remote_id "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"

 local_addr  192.168.116.16
 remote_addr 192.168.13.213

 p1_xform
  {auth_method rsa_sig oakley_group 2 auth_alg md5 encr_alg 3des}
}
```

- **Certificate request**

  Type the name that the PKI organization provides as the value for the `cert_root` keyword. For example, the `ike/config` file on the `enigma` system might appear similar to the following:

```
# Trusted root cert
# This certificate is from Example PKI
# This is the X.509 distinguished name for the CA that it issues.

cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example, CN=Example PKI"

# Solaris 10 1/06 release: default path does not have to be typed in
#pkcs11_path "/usr/lib/libpkcs11.so"      Hardware connection
```

```
# Solaris 10 release: use this path
#pkcs11_path "/opt/SUNWconn/cryptov2/lib/libvpkcs11.so"
...
{
 label "JA-enigmax to US-partym - Example PKI"
 local_id_type dn
 local_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
 remote_id  "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"

 local_addr  192.168.116.16
 remote_addr 192.168.13.213

 p1_xform
  {auth_method rsa_sig oakley_group 2 auth_alg md5 encr_alg 3des}
}
```

**6 Place the certificates from the other party in the hardware.**

Respond to the PIN request as you responded in Step 3.

---

**Note –** You *must* add the public key certificates to the same attached hardware that generated your private key.

---

- **Self-signed certificate.**

  Add the remote system's self-signed certificate. In this example, the certificate is stored in the file, `DCA.ACCEL.STOR.CERT`.

  ```
  # ikecert certdb -a -T dca0-accel-stor < DCA.ACCEL.STOR.CERT
  Enter PIN for PKCS#11 token:      Type user:password
  ```

  If the self-signed certificate used `rsa_encrypt` as the value for the `auth_method` parameter, add the peer's certificate to the hardware store.

- **Certificates from a PKI organization.**

  Add the certificate that the organization generated from your certificate request, and add the certificate authority (CA).

  ```
  # ikecert certdb -a -T dca0-accel-stor < DCA.ACCEL.STOR.CERT
  Enter PIN for PKCS#11 token:      Type user:password

  # ikecert certdb -a -T dca0-accel-stor < DCA.ACCEL.STOR.CA.CERT
  Enter PIN for PKCS#11 token:      Type user:password
  ```

  To add a certificate revocation list (CRL) from the PKI organization, see "How to Handle a Certificate Revocation List" on page 581.

## How to Handle a Certificate Revocation List

A certificate revocation list (CRL) contains outdated or compromised certificates from a Certificate Authority. You have four ways to handle CRLs.

- You must instruct IKE to ignore CRLs if your CA organization does not issue CRLs. This option is shown in Step 6 in "How to Configure IKE With Certificates Signed by a CA" on page 571.
- You can instruct IKE to access the CRLs from a URI (uniform resource indicator) whose address is embedded in the public key certificate from the CA.
- You can instruct IKE to access the CRLs from an LDAP server whose DN (directory name) entry is embedded in the public key certificate from the CA.
- You can provide the CRL as an argument to the `ikecert certrldb` command. For an example, see Example 23–5.

The following procedure describes how to instruct IKE to use CRLs from a central distribution point.

**1 Display the certificate that you received from the CA.**

```
# ikecert certdb -lv certspec
```

`-l` Lists certificates in the IKE certificate database.

`-v` Lists the certificates in verbose mode. Use this option with care.

*certspec* Is a pattern that matches a certificate in the IKE certificate database.

For example, the following certificate was issued by Sun Microsystems. Details have been altered.

```
# ikecert certdb -lv example-protect.sun.com
Certificate Slot Name: 0   Type: dsa-sha1
   (Private key in certlocal slot 0)
 Subject Name: <O=Sun Microsystems Inc, CN=example-protect.sun.com>
 Issuer Name: <CN=Sun Microsystems Inc CA (Cl B), O=Sun Microsystems Inc>
 SerialNumber: 14000D93
   Validity:
      Not Valid Before: 2002 Jul 19th, 21:11:11 GMT
      Not Valid After:  2005 Jul 18th, 21:11:11 GMT
   Public Key Info:
      Public Modulus  (n) (2048 bits): C575A...A5
      Public Exponent (e) (  24 bits): 010001
   Extensions:
      Subject Alternative Names:
              DNS = example-protect.sun.com
      Key Usage: DigitalSignature KeyEncipherment
```

```
   [CRITICAL]
CRL Distribution Points:
   Full Name:
      URI = #Ihttp://www.sun.com/pki/pkismica.crl#i
      DN = <CN=Sun Microsystems Inc CA (Cl B), O=Sun Microsystems Inc>
   CRL Issuer:
   Authority Key ID:
   Key ID:              4F ... 6B
   SubjectKeyID:        A5 ... FD
   Certificate Policies
   Authority Information Access
```

Notice the `CRL Distribution Points` entry. The URI entry indicates that this organization's CRL is available on the web. The DN entry indicates that the CRL is available on an LDAP server. Once accessed by IKE, the CRL is cached for further use.

To access the CRL, you need to reach a distribution point.

**2 Choose one of the following methods to access the CRL from a central distribution point.**

- **Use the URI.**

  Add the keyword `use_http` to the host's `/etc/inet/ike/config` file. For example, the `ike/config` file would appear similar to the following:

  ```
  # Use CRL from organization's URI
  use_http
  ...
  ```

- **Use a web proxy.**

  Add the keyword `proxy` to the `ike/config` file. The `proxy` keyword takes a URL as an argument, as in the following:

  ```
  # Use own web proxy
  proxy "http://proxy1:8080"
  ```

- **Use an LDAP server.**

  Name the LDAP server as an argument to the `ldap-list` keyword in the host's `/etc/inet/ike/config` file. Your organization provides the name of the LDAP server. The entry in the `ike/config` file would appear similar to the following:

  ```
  # Use CRL from organization's LDAP
  ldap-list "ldap1.sun.com:389,ldap2.sun.com"
  ...
  ```

IKE retrieves the CRL and caches the CRL until the certificate expires.

**Example 23–5** Pasting a CRL Into the Local `certrldb` Database

If the PKI organization's CRL is not available from a central distribution point, you can add the CRL manually to the local `certrldb` database. Follow the PKI organization's instructions for extracting the CRL into a file, then add the CRL to the database with the `ikecert certrldb -a` command.

```
# ikecert certrldb -a < Sun.Cert.CRL
```

# Configuring IKE for Mobile Systems (Task Map)

The following table points to procedures to configure IKE to handle systems that log in remotely to a central site.

| Task | Description | For Instructions |
|---|---|---|
| Communicate with a central site from off-site | Enables off-site systems to communicate with a central site. The off-site systems might be mobile. | "How to Configure IKE for Off-Site Systems" on page 584 |
| Use a root certificate and IKE on a central system that accepts traffic from mobile systems | Configures a gateway system to accept IPsec traffic from a system that does not have a fixed IP address. | Example 23–6 |
| Use a root certificate and IKE on a system that does not have a fixed IP address | Configures a mobile system to protect its traffic to a central site, such as company headquarters. | Example 23–7 |
| Use self-signed certificates and IKE on a central system that accepts traffic from mobile systems | Configures a gateway system with self-signed certificates to accept IPsec traffic from a mobile system. | Example 23–8 |
| Use self-signed certificates and IKE on a system that does not have a fixed IP address | Configures a mobile system with self-signed certificates to protect its traffic to a central site. | Example 23–9 |

# Configuring IKE for Mobile Systems

When configured properly, home offices and mobile laptops can use IPsec and IKE to communicate with their company's central computers. A blanket IPsec policy that is combined with a public key authentication method enables off-site systems to protect their traffic to a central system.

## ▼ How to Configure IKE for Off-Site Systems

IPsec and IKE require a unique ID to identify source and destination. For off-site or mobile systems that do not have a unique IP address, you must use another ID type. ID types such as `DNS`, `DN`, or `email` can be used to uniquely identify a system.

Off-site or mobile systems that have unique IP addresses are still best configured with a different ID type. For example, if the systems attempt to connect to a central site from behind a NAT box, their unique addresses are not used. A NAT box assigns an arbitrary IP address, which the central system would not recognize.

Preshared keys also do not work well as an authentication mechanism for mobile systems, because preshared keys require fixed IP addresses. Self-signed certificates, or certificates from a PKI enable mobile systems to communicate with the central site.

**1 On the system console of the central system, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

---

**2 Configure the central system to recognize mobile systems.**

**a. Set up the `/etc/hosts` file.**

The central system does not have to recognize specific addresses for the mobile systems.

```
# /etc/hosts on central
central 192.xxx.xxx.x
```

**b. Set up the `ipsecinit.conf` file.**

The central system needs a policy that allows a wide range of IP addresses. Later, certificates in the IKE policy ensure that the connecting systems are legitimate.

```
# /etc/inet/ipsecinit.conf on central
# Keep everyone out unless they use this IPsec policy:
{} ipsec {encr_algs aes encr_auth_algs md5 sa shared}
```

**c. Set up the `ike.config` file.**

DNS identifies the central system. Certificates are used to authenticate the system.

```
## /etc/inet/ike/ike.config on central
# Global parameters
```

```
#
# Find CRLs by URI, URL, or LDAP
# Use CRL from organization's URI
use_http
#
# Use web proxy
proxy "http://somecache.domain:port/"
#
# Use LDAP server
ldap_server    "ldap-server1.domain.org,ldap2.domain.org:port"
#
# List CA-signed certificates
cert_root     "C=US, O=Domain  Org, CN=Domain STATE"
#
# List self-signed certificates - trust server and enumerated others
#cert_trust    "DNS=central.domain.org"
#cert_trust    "DNS=mobile.domain.org"
#cert_trust    "DN=CN=Domain Org STATE  (CLASS), O=Domain Org
#cert_trust    "email=root@central.domain.org"
#cert_trust    "email=user1@mobile.domain.org"
#

# Rule for mobile systems with certificate
{
  label "Mobile systems with certificate"
  local_id_type DNS

# Any mobile system who knows my DNS or IP can find me.

  local_id "central.domain.org"
  local_addr 192.xxx.xxx.x

# Root certificate ensures trust,
# so allow any remote_id and any remote IP address.
  remote_id ""
  remote_addr 0.0.0.0/0

p2_pfs 5

p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg blowfish auth_alg md5 }
}
```

**3 Log in to each mobile system, and configure the system to find the central system.**

**a. Set up the `/etc/hosts` file.**

The `/etc/hosts` file does not need an address for the mobile system, but can provide one. The file must contain a public IP address for the central system.

```
# /etc/hosts on mobile
mobile 10.x.x.xx
central 192.xxx.xxx.x
```

**b. Set up the `ipsecinit.conf` file.**

The mobile system needs to find the central system by its public IP address. The systems must configure the same IPsec policy.

```
# /etc/inet/ipsecinit.conf on mobile
# Find central
{raddr 192.xxx.xxx.x} ipsec {encr_algs aes encr_auth_algs md5 sa shared}
```

**c. Set up the `ike.config` file.**

The identifier cannot be an IP address. The following identifiers are valid for mobile systems:

- `DN=`*ldap-directory-name*
- `DNS=`*domain-name-server-address*
- `email=`*email-address*

Certificates are used to authenticate the mobile system.

```
## /etc/inet/ike/ike.config on mobile
# Global parameters
#
# Find CRLs by URI, URL, or LDAP
# Use CRL from organization's URI
use_http
#
# Use web proxy
proxy "http://somecache.domain:port/"
#
# Use LDAP server
ldap_server   "ldap-server1.domain.org,ldap2.domain.org:port"
#
# List CA-signed certificates
cert_root    "C=US, O=Domain Org, CN=Domain STATE"
#
# Self-signed certificates - trust me and enumerated others
#cert_trust    "DNS=mobile.domain.org"
#cert_trust    "DNS=central.domain.org"
#cert_trust    "DN=CN=Domain Org STATE (CLASS), O=Domain Org
```

```
#cert_trust    "email=user1@domain.org"
#cert_trust    "email=root@central.domain.org"
#
# Rule for off-site systems with root certificate
{
    label "Off-site mobile with certificate"
    local_id_type DNS

# NAT-T can translate local_addr into any public IP address
# central knows me by my DNS

    local_id "mobile.domain.org"
    local_addr 0.0.0.0/0

# Find central and trust the root certificate
    remote_id "central.domain.org"
    remote_addr 192.xxx.xxx.x

p2_pfs 5

p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg blowfish auth_alg md5 }
}
```

4 **Reboot each system when it is configured.**

Or, stop and start the `in.iked` daemon.

### Example 23–6 Configuring a Central Computer to Accept IPsec Traffic From a Mobile System

IKE can initiate negotiations from behind a NAT box. However, the ideal setup for IKE is without an intervening NAT box. In the following example, root certificates have been issued by a CA. The CA certificates have been placed on the mobile system and the central system. A central system accepts IPsec negotiations from a system behind a NAT box. `main1` is the company system that can accept connections from off-site systems. To set up the off-site systems, see Example 23–7.

```
## /etc/hosts on main1
main1 192.168.0.100

## /etc/inet/ipsecinit.conf on main1
# Keep everyone out unless they use this IPsec policy:
{} ipsec {encr_algs aes encr_auth_algs md5 sa shared}

## /etc/inet/ike/ike.config on main1
# Global parameters
#
# Find CRLs by URI, URL, or LDAP
```

```
# Use CRL from organization's URI
use_http
#
# Use web proxy
proxy "http://cache1.domain.org:8080/"
#
# Use LDAP server
ldap_server   "ldap1.domain.org,ldap2.domain.org:389"
#
# List CA-signed certificate
cert_root "C=US, O=ExamplePKI Inc, OU=PKI-Example, CN=Example PKI"
#
# Rule for off-site systems with root certificate
{
  label "Off-site system with root certificate"
  local_id_type DNS
  local_id "main1.domain.org"
  local_addr 192.168.0.100

# Root certificate ensures trust,
# so allow any remote_id and any remote IP address.
  remote_id ""
  remote_addr 0.0.0.0/0

p2_pfs 5

p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg blowfish auth_alg md5}
p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg 3des auth_alg md5}
p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg blowfish auth_alg sha}
p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg 3des auth_alg sha}
}
```

### Example 23–7 Configuring a System Behind a NAT With IPsec

In the following example, root certificates have been issued by a CA and placed on the mobile system and the central system. `mobile1` is connecting to the company headquarters from home. The Internet service provider (ISP) network uses a NAT box to enable the ISP to assign `mobile1` a private address. The NAT box then translates the private address into a public IP address that is shared with other ISP network nodes. Company headquarters is not behind a NAT. For setting up the computer at company headquarters, see Example 23–6.

```
## /etc/hosts on mobile1
mobile1 10.1.3.3
main1 192.168.0.100
```

```
## /etc/inet/ipsecinit.conf on mobile1
# Find main1
{raddr 192.168.0.100} ipsec {encr_algs aes encr_auth_algs md5 sa shared}

## /etc/inet/ike/ike.config on mobile1
# Global parameters
#
# Find CRLs by URI, URL, or LDAP
# Use CRL from organization's URI
use_http
#
# Use web proxy
proxy "http://cache1.domain.org:8080/"
#
# Use LDAP server
ldap_server   "ldap1.domain.org,ldap2.domain.org:389"
#
# List CA-signed certificate
cert_root "C=US, O=ExamplePKI Inc, OU=PKI-Example, CN=Example PKI"
#
# Rule for off-site systems with root certificate
{
  label "Off-site mobile1 with root certificate"
  local_id_type DNS
  local_id "mobile1.domain.org"
  local_addr 0.0.0.0/0

# Find main1 and trust the root certificate
  remote_id "main1.domain.org"
  remote_addr 192.168.0.100

p2_pfs 5

p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg blowfish auth_alg md5 }
}
```

### Example 23–8 Accepting Self-Signed Certificates From a Mobile System

In the following example, self-signed certificates have been issued and are on the mobile and the central system. `main1` is the company system that can accept connections from off-site systems. To set up the off-site systems, see Example 23–9.

```
## /etc/hosts on main1
main1 192.168.0.100

## /etc/inet/ipsecinit.conf on main1
# Keep everyone out unless they use this IPsec policy:
```

```
{} ipsec {encr_algs aes encr_auth_algs md5 sa shared}

## /etc/inet/ike/ike.config on main1
# Global parameters
#
# Self-signed certificates - trust me and enumerated others
cert_trust    "DNS=main1.domain.org"
cert_trust    "jdoe@domain.org"
cert_trust    "user2@domain.org"
cert_trust    "user3@domain.org"
#
# Rule for off-site systems with trusted certificate
{
  label "Off-site systems with trusted certificates"
  local_id_type DNS
  local_id "main1.domain.org"
  local_addr 192.168.0.100

# Trust the self-signed certificates
# so allow any remote_id and any remote IP address.
  remote_id ""
  remote_addr 0.0.0.0/0

p2_pfs 5

p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg blowfish auth_alg md5 }
}
```

### Example 23–9 Using Self-Signed Certificates to Contact a Central System

In the following example, `mobile1` is connecting to the company headquarters from home. The certificates have been issued and placed on the mobile and the central system. The ISP network uses a NAT box to enable the ISP to assign `mobile1` a private address. The NAT box then translates the private address into a public IP address that is shared with other ISP network nodes. Company headquarters is not behind a NAT. To set up the computer at company headquarters, see Example 23–8.

```
## /etc/hosts on mobile1
mobile1 10.1.3.3
main1 192.168.0.100

## /etc/inet/ipsecinit.conf on mobile1
# Find main1
{raddr 192.168.0.100} ipsec {encr_algs aes encr_auth_algs md5 sa shared}
```

```
## /etc/inet/ike/ike.config on mobile1
# Global parameters

# Self-signed certificates - trust me and the central system
cert_trust    "jdoe@domain.org"
cert_trust    "DNS=main1.domain.org"
#
# Rule for off-site systems with trusted certificate
{
  label "Off-site mobile1 with trusted certificate"
  local_id_type email
  local_id "jdoe@domain.org"
  local_addr 0.0.0.0/0

# Find main1 and trust the certificate
  remote_id "main1.domain.org"
  remote_addr 192.168.0.100

p2_pfs 5

p1_xform
{auth_method rsa_sig oakley_group 5 encr_alg blowfish auth_alg md5 }
}
```

# Configuring IKE to Find Attached Hardware (Task Map)

The following table points to procedures that inform IKE about attached hardware. You must inform IKE about attached hardware before IKE can use the hardware. To use the hardware, follow the hardware procedures in "Configuring IKE With Public Key Certificates" on page 565.

| Task | Description | For Instructions |
|---|---|---|
| Offload IKE key operations to the Sun Crypto Accelerator 1000 board | Links IKE to the PKCS #11 library. | "How to Configure IKE to Find the Sun Crypto Accelerator 1000 Board" on page 592 |
| Offload IKE key operations and store the keys on the Sun Crypto Accelerator 4000 board | Links IKE to the PKCS #11 library and lists the name of the attached hardware. | "How to Configure IKE to Find the Sun Crypto Accelerator 4000 Board" on page 593 |

# Configuring IKE to Find Attached Hardware

Public key certificates can also be stored on attached hardware, the Sun Crypto Accelerator 1000 board and the Sun Crypto Accelerator 4000 board. With the Sun Crypto Accelerator 4000 board, public key operations can also be offloaded from the system to the board.

## ▼ How to Configure IKE to Find the Sun Crypto Accelerator 1000 Board

**Before You Begin** The following procedure assumes that a Sun Crypto Accelerator 1000 board is attached to the system. The procedure also assumes that the software for the board has been installed and that the software has been configured. For instructions, see the *Sun Crypto Accelerator 1000 Board Version 1.1 Installation and User's Guide*.

**1 On the system console, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

---

**2 Check that the PKCS #11 library is linked.**

Type the following command to determine whether a PKCS #11 library is linked:

```
# ikeadm get stats
Phase 1 SA counts:
Current:   initiator:          0   responder:          0
Total:     initiator:          0   responder:          0
Attempted: initiator:          0   responder:          0
Failed:    initiator:          0   responder:          0
           initiator fails include 0 time-out(s)
PKCS#11 library linked in from /usr/lib/libpkcs11.so
#
```

**3 Solaris 10 1/06: Starting in this release, you can store keys in the softtoken keystore.**

For information on the keystore that is provided by the Solaris cryptographic framework, see the `cryptoadm`(1M) man page. For an example of using the keystore, see Example 23–10.

## ▼ How to Configure IKE to Find the Sun Crypto Accelerator 4000 Board

**Before You Begin** The following procedure assumes that a Sun Crypto Accelerator 4000 board is attached to the system. The procedure also assumes that the software for the board has been installed and that the software has been configured. For instructions, see the *Sun Crypto Accelerator 4000 Board Installation and User's Guide*. The guide is available from the `http://www.sun.com/products-n-solutions/hardware/docs` web site, under Network and Security Products.

**1 On the system console, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

---

**2 Check that the PKCS #11 library is linked.**

IKE uses the library's routines to handle key generation and key storage on the Sun Crypto Accelerator 4000 board. Type the following command to determine whether a PKCS #11 library has been linked:

```
$ ikeadm get stats
...
PKCS#11 library linked in from /usr/lib/libpkcs11.so
$
```

---

**Note –** The Sun Crypto Accelerator 4000 board supports keys up to 2048 bits for RSA. For DSA, this board supports keys up to 1024 bits.

---

**3 Find the token ID for the attached Sun Crypto Accelerator 4000 board.**

```
$ ikecert tokens
Available tokens with library "/usr/lib/libpkcs11.so":

"Sun Metaslot                        "
```

The library returns a token ID, also called a keystore name, of 32 characters. In this example, you could use the `Sun Metaslot` token with the `ikecert` commands to store and accelerate IKE keys.

For instructions on how to use the token, see "How to Generate and Store Public Key Certificates on Hardware" on page 577.

The trailing spaces are automatically padded by the `ikecert` command.

**Example 23–10** Finding and Using Metaslot Tokens

Tokens can be stored on disk, on an attached board, or in the softtoken keystore that the Solaris encryption framework provides. The softtoken keystore token ID might resemble the following.

```
$ ikecert tokens
Available tokens with library "/usr/lib/libpkcs11.so":

"Sun Metaslot                    "
```

To create a passphrase for the softtoken keystore, see the `pktool`(1) man page.

A command that resembles the following would add a certificate to the softtoken keystore. `Sun.Metaslot.cert` is a file that contains the CA certificate.

```
# ikecert certdb -a -T "Sun Metaslot" < Sun.Metaslot.cert
Enter PIN for PKCS#11 token:      Type user:passphrase
```

# Changing IKE Transmission Parameters (Task Map)

The following table points to procedures to configure transmission parameters for IKE.

| Task | Description | For Instructions |
|---|---|---|
| Make key negotiation more efficient | Changes the key negotiation parameters. | "How to Change the Duration of Phase 1 IKE Key Negotiation" on page 595 |
| Configure key negotiation to allow for delays in transmission | Lengthens the key negotiation parameters. | Example 23–11 |
| Configure key negotiation to succeed quickly, or to show failures quickly | Shortens the key negotiation parameters. | Example 23–12 |

# Changing IKE Transmission Parameters

When IKE negotiates keys, the speed of transmission can affect the success of the negotiation. Normally, you would not need to change the default values for IKE transmission parameters. However, when optimizing key negotiation over very dirty lines, or when reproducing a problem, you might want to change the transmission values.

Longer duration times enable IKE to negotiate keys over unreliable transmission lines. You can lengthen certain parameters so that initial attempts succeed. If the initial attempt does not succeed, you can space subsequent attempts to offer more time for success.

Shorter duration times enable you to take advantage of reliable transmission lines. You can more quickly retry a failed negotiation to speed up the negotiation. When diagnosing a problem, you might also want to speed up the negotiation for a quick failure. Shorter durations also enable the Phase 1 SAs to be used for their lifetime.

## How to Change the Duration of Phase 1 IKE Key Negotiation

**1 On the system console, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

---

**Note –** Logging in remotely exposes security-critical traffic to eavesdropping. Even if you somehow protect the remote login, the security of the system is reduced to the security of the remote login session.

---

**2 Change the default values of the global transmission parameters on each system.**

On each system, modify Phase 1 duration parameters the `/etc/inet/ike/config` file.

```
### ike/config file on     system

## Global parameters
#
## Phase 1 transform defaults
#
#expire_timer      300
#retry_limit         5
#retry_timer_init    0.5 (integer or float)
#retry_timer_max    30   (integer or float)
```

`expire_timer` The number of seconds to let a not-yet-complete IKE Phase I negotiation linger before deleting the negotiation attempt. By default, the attempt lingers for 30 seconds.

`retry_limit` The number of retransmits before any IKE negotiation is aborted. By default, IKE tries five times.

`retry_timer_init` The initial interval between retransmits. This interval is doubled until the `retry_timer_max` value is reached. The initial interval is 0.5 seconds.

`retry_timer_max` The maximum interval in seconds between retransmits. The retransmit interval stops growing at this limit. By default, the limit is 30 seconds.

**3 Reboot the system.**

Or, stop and start the `in.iked` daemon.

**Example 23–11** Lengthening IKE Phase 1 Negotiation Times

In the following example, a system is connected to its IKE peers by a high-traffic transmission line. The original settings are in comments in the file. The new settings lengthen the negotiation time.

```
### ike/config file on partym
## Global Parameters
#
## Phase 1 transform defaults
#expire_timer   300
#retry_limit      5
#retry_timer_init 0.5 (integer or float)
#retry_timer_max 30   (integer or float)
#
expire_timer  600
retry_limit  10
retry_timer_init  2.5
retry_timer_max  180
```

**Example 23–12** Shortening IKE Phase 1 Negotiation Times

In the following example, a system is connected to its IKE peers by a high-speed line with little traffic. The original settings are in comments in the file. The new settings shorten the negotiation time.

```
### ike/config file on partym
## Global Parameters
#
## Phase 1 transform defaults
#expire_timer   300
#retry_limit      5
#retry_timer_init 0.5 (integer or float)
#retry_timer_max 30   (integer or float)
#
expire_timer  120
retry_timer_init  0.20
```

◆ ◆ ◆ CHAPTER 24

# Internet Key Exchange (Reference)

This chapter contains the following reference information about IKE:


- "IKE Daemon" on page 597
- "IKE Policy File" on page 598
- "IKE Administration Command" on page 598
- "IKE Preshared Keys Files" on page 599
- "IKE Public Key Databases and Commands" on page 599


For instructions on implementing IKE, see Chapter 23, "Configuring IKE (Tasks)." For overview information, see Chapter 22, "Internet Key Exchange (Overview)."

## IKE Daemon

The `in.iked` daemon automates the management of cryptographic keys for IPsec on a Solaris system. The daemon negotiates with a remote system that is running the same protocol to provide authenticated keying materials for security associations (SAs) in a protected manner. The daemon must be running on all systems that plan to communicate securely.

The IKE daemon is automatically loaded at boot time if the configuration file for the IKE policy, `/etc/inet/ike/config`, exists. The daemon checks the syntax of the configuration file.

When the IKE daemon runs, the system authenticates itself to its peer IKE entity in the Phase 1 exchange. The peer is defined in the IKE policy file, as are the authentication methods. The daemon then establishes the keys for the Phase 2 exchange. At an interval specified in the policy file, the IKE keys are refreshed automatically. The `in.iked` daemon listens for incoming IKE requests from the network and for requests for outbound traffic through the `PF_KEY` socket. For more information, see the `pf_key`(7P) man page.

Two commands support the IKE daemon. The `ikeadm` command enables you to view and modify the IKE policy. The `ikecert` command enables you to view and manage the public key databases. This command manages the local databases, `ike.privatekeys` and `publickeys`. This command also manages public key operations and the storage of public keys on hardware.

# IKE Policy File

The configuration file for the IKE policy, `/etc/inet/ike/config`, manages the keys for the interfaces that are being protected in the IPsec policy file, `/etc/inet/ipsecinit.conf`. The IKE policy file manages keys for IKE, and for the IPsec SAs. The IKE daemon itself requires keying material in the Phase 1 exchange.

Key management with IKE includes rules and global parameters. An IKE rule identifies the systems or networks that the keying material secures. The rule also specifies the authentication method. Global parameters include such items as the path to an attached hardware accelerator. For examples of IKE policy files, see "Configuring IKE With Preshared Keys (Task Map)" on page 554. For examples and descriptions of IKE policy entries, see the `ike.config`(4) man page.

The IPsec SAs that IKE supports protect the IP datagrams according to policies that are set up in the configuration file for the IPsec policy, `/etc/inet/ipsecinit.conf`. The IKE policy file determines if perfect forward security (PFS) is used when creating the IPsec SAs.

The `ike/config` file can include the path to a library that is implemented according to the following standard: RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki). IKE uses this PKCS #11 library to access hardware for key acceleration and key storage.

The security considerations for the `ike/config` file are similar to the considerations for the `ipsecinit.conf` file. For details, see "Security Considerations for `ipsecinit.conf` and `ipsecconf`" on page 539.

# IKE Administration Command

You can use the `ikeadm` command to do the following:

- View aspects of the IKE daemon process.
- Change the parameters that are passed to the IKE daemon.
- Display statistics on SA creation during the Phase 1 exchange.
- Debug IKE processes.
- View aspects of the IKE state.
- Change the properties of the IKE daemon.
- Display statistics on SA creation during the Phase 1 exchange.
- Debug IKE protocol exchanges.

For examples and a full description of this command's options, see the `ikeadm`(1M) man page. The privilege level of the running IKE daemon determines which aspects of the IKE daemon can be viewed and modified. You can choose from three levels of privilege.

| | |
|---|---|
| `0x0`, or base level | You cannot view nor modify keying material. The base level is the default level at which the `in.iked` daemon runs. |
| `0x1`, or modkeys level | You can remove, change, and add preshared keys. |

`0x2`, or keymat level — You can view the actual keying material with the `ikeadm` command.

The security considerations for the `ikeadm` command are similar to the considerations for the `ipseckey` command. For details, see "Security Considerations for `ipseckey`" on page 541.

# IKE Preshared Keys Files

When you create preshared keys manually, the keys are stored in files in the `/etc/inet/secret` directory. The `ike.preshared` file contains the preshared keys for Internet Security Association and Key Management Protocol (ISAKMP) SAs. The `ipseckeys` file contains the preshared keys for IPsec SAs. The files are protected at `0600`. The `secret` directory is protected at `0700`.

- You create an `ike.preshared` file when you configure the `ike/config` file to require preshared keys. You enter keying material for ISAKMP SAs, that is, for IKE authentication, in the `ike.preshared` file. Because the preshared keys are used to authenticate the Phase 1 exchange, the file must be valid before the `in.iked` daemon starts.
- The `ipseckeys` file contains keying material for IPsec SAs. For examples of manually managing the file, see "How to Manually Create IPsec Security Associations" on page 503. The IKE daemon does not use this file. The keying material that IKE generates for IPsec SAs is stored in the kernel.

---

**Note –** Preshared keys cannot take advantage of hardware storage. Preshared keys are generated and are stored on the system.

---

# IKE Public Key Databases and Commands

The `ikecert` command manipulates the local system's public key databases. You use this command when the `ike/config` file requires public key certificates. Because IKE uses these databases to authenticate the Phase 1 exchange, the databases must be populated before activating the `in.iked` daemon. Three subcommands handle each of the three databases: `certlocal`, `certdb`, and `certrldb`.

The `ikecert` command also handles key storage. Keys can be stored on disk, on an attached Sun Crypto Accelerator 4000 board, or in a softtoken keystore. The softtoken keystore is available when the metaslot in the Solaris cryptographic framework is used to communicate with the hardware device. The `ikecert` command uses the PKCS #11 library to locate key storage.

- **Solaris 10 1/06**: Starting in this release, the library does not have to be specified. By default, the PKCS #11 library is `/usr/lib/libpkcs11.so`.
- **Solaris 10**: In this release, the PKCS #11 entry must be specified. Otherwise, the `-T` option to the `ikecert` command cannot work. The entry appears similar to the following:

```
pkcs11_path "/opt/SUNWconn/cryptov2/lib/libvpkcs11.so"
```

For more information, see the `ikecert`(1M) man page. For information about metaslot and the softtoken keystore, see the `cryptoadm`(1M) man page.

## `ikecert tokens` Command

The `tokens` argument lists the token IDs that are available. Token IDs enable the `ikecert certlocal` and `ikecert certdb` commands to generate public key certificates and certificate requests. The certificates and certificate requests can also be stored by the cryptographic framework in the softtoken keystore, or on an attached Sun Crypto Accelerator 4000 board. The `ikecert` command uses the PKCS #11 library to locate certificate storage.

## `ikecert certlocal` Command

The `certlocal` subcommand manages the private key database. Options to this subcommand enable you to add, view, and remove private keys. This subcommand also creates either a self-signed certificate or a certificate request. The `-ks` option creates a self-signed certificate. The `-kc` option creates a certificate request. Keys are stored on the system in the `/etc/inet/secret/ike.privatekeys` directory, or on attached hardware with the `-T` option.

When you create a private key, the options to the `ikecert certlocal` command must have related entries in the `ike/config` file. The correspondences between `ikecert` options and `ike/config` entries are shown in the following table.

**TABLE 24–1** Correspondences Between `ikecert` Options and `ike/config` Entries

| `ikecert` **Option** | `ike/config` **Entry** | **Description** |
|---|---|---|
| `-A` *subject-alternate-name* | `cert_trust` *subject-alternate-name* | A nickname that uniquely identifies the certificate. Possible values are an IP address, an email address, or a domain name. |
| `-D` *X.509-distinguished-name* | *X.509-distinguished-name* | The full name of the certificate authority that includes the country (C), organization name (ON), organizational unit (OU), and common name (CN). |
| `-t dsa-sha1` | `auth_method dss_sig` | An authentication method that is slightly slower than RSA. |
| `-t rsa-md5` and | `auth_method rsa_sig` | An authentication method that is slightly faster than DSA. |
| `-t rsa-sha1` | | The RSA public key must be large enough to encrypt the biggest payload. Typically, an identity payload, such as the X.509 distinguished name, is the biggest payload. |

**TABLE 24–1** Correspondences Between `ikecert` Options and `ike/config` Entries *(Continued)*

| `ikecert` **Option** | `ike/config` **Entry** | **Description** |
|---|---|---|
| `-t rsa-md5` and<br>`-t rsa-sha1` | `auth_method rsa_encrypt` | RSA encryption hides identities in IKE from eavesdroppers, but requires that the IKE peers know each other's public keys. |
| `-T` | `pkcs11_path` | The PKCS #11 library handles key acceleration on the Sun Crypto Accelerator 1000 board and the Sun Crypto Accelerator 4000 board. The library also provides the tokens that handle key storage on the Sun Crypto Accelerator 4000 board. |

If you issue a certificate request with the `ikecert certlocal -kc` command, you send the output of the command to a PKI organization or to a certificate authority (CA). If your company runs its own PKI, you send the output to your PKI administrator. The PKI organization, the CA, or your PKI administrator then creates certificates. The certificates that the PKI or CA returns to you are input to the `certdb` subcommand. The certificate revocation list (CRL) that the PKI returns to you is input for the `certrldb` subcommand.

## `ikecert certdb` Command

The `certdb` subcommand manages the public key database. Options to this subcommand enable you to add, view, and remove certificates and public keys. The command accepts, as input, certificates that were generated by the `ikecert certlocal -ks` command on a remote system. For the procedure, see "How to Configure IKE With Self-Signed Public Key Certificates" on page 565. This command also accepts the certificate that you receive from a PKI or CA as input. For the procedure, see "How to Configure IKE With Certificates Signed by a CA" on page 571.

The certificates and public keys are stored on the system in the `/etc/inet/ike/publickeys` directory. The `-T` option stores the certificates, private keys, and public keys on attached hardware.

## `ikecert certrldb` Command

The `certrldb` subcommand manages the certificate revocation list (CRL) database, `/etc/inet/ike/crls`. The CRL database maintains the revocation lists for public keys. Certificates that are no longer valid are on this list. When PKIs provide you with a CRL, you can install the CRL in the CRL database with the `ikecert certrldb` command. For the procedure, see "How to Handle a Certificate Revocation List" on page 581.

## /etc/inet/ike/publickeys Directory

The /etc/inet/ike/publickeys directory contains the public part of a public-private key pair and its certificate in files, or *slots*. The directory is protected at 0755. The ikecert certdb command populates the directory. The -T option stores the keys on the Sun Crypto Accelerator 4000 board rather than in the publickeys directory.

The slots contain, in encoded form, the X.509 distinguished name of a certificate that was generated on another system. If you are using self-signed certificates, you use the certificate that you receive from the administrator of the remote system as input to the command. If you are using certificates from a PKI, you install two pieces of keying material from the PKI into this database. You install a certificate that is based on material that you sent to the PKI. You also install a CA from the PKI.

## /etc/inet/secret/ike.privatekeys Directory

The /etc/inet/secret/ike.privatekeys directory holds private key files that are part of a public-private key pair, which is keying material for ISAKMP SAs. The directory is protected at 0700. The ikecert certlocal command populates the ike.privatekeys directory. Private keys are not effective until their public key counterparts, self-signed certificates or CAs, are installed. The public key counterparts are stored in the /etc/inet/ike/publickeys directory or on a Sun Crypto Accelerator 4000 board.

## /etc/inet/ike/crls Directory

The /etc/inet/ike/crls directory contains certificate revocation list (CRL) files. Each file corresponds to a public certificate file in the /etc/inet/ike/publickeys directory. PKI organizations provide the CRLs for their certificates. You can use the ikecert certrldb command to populate the database.

◆ ◆ ◆ CHAPTER 25

# Solaris IP Filter (Overview)

This chapter provides an overview of Solaris IP Filter. For Solaris IP Filter tasks, see Chapter 26, "Solaris IP Filter (Tasks)."

This chapter contains the following information:


- "What's New in Solaris IP Filter" on page 603
- "Introduction to Solaris IP Filter" on page 604
- "Solaris IP Filter Packet Processing" on page 605
- "Guidelines for Using Solaris IP Filter" on page 608
- "Using Solaris IP Filter Configuration Files" on page 608
- "Working With Solaris IP Filter Rule Sets" on page 608
- "Packet Filter Hooks" on page 614
- "Solaris IP Filter and the `pfil` STREAMS Module" on page 615
- "IPv6 for Solaris IP Filter" on page 615
- "Solaris IP Filter Man Pages" on page 616


## What's New in Solaris IP Filter

This section describes new Solaris IP Filter features in the Solaris release.

For a complete listing of new Solaris features and a description of Solaris releases, see *Solaris Express Developer Edition What's New*

### Packet Filter Hooks

Solaris 10 8/07 Release: Packet filter hooks are now used for packet filtering in the Solaris Operating System. This feature offers the following advantages in system administration:

- Packet filter hooks simplify the configuration of the Solaris IP filter.
- Support for filtering packets across zones is now available.

- Using filter hooks improves the performance of Solaris IP Filter.

For further details about these hooks, see "Packet Filter Hooks" on page 614. For tasks that are associated with packet filter hooks, see Chapter 26, "Solaris IP Filter (Tasks)."

## IPv6 Packet Filtering for Solaris IP Filter

Solaris 10 6/06: For system administrators who have all or part of their network infrastructure configured with IPv6, Solaris IP Filter has been enhanced to include IPv6 packet filtering. IPv6 packet filtering can filter based on the source/destination IPv6 address, pools containing IPv6 addresses, and IPv6 extension headers.

The `-6` option has been added to both the `ipf` command and the `ipfstat` command to use with IPv6. Although there is no change to the command line interface for the `ipmon` and `ippool` commands, these commands also support IPv6. The `ipmon` command has been enhanced to accommodate the logging of IPv6 packets, and the `ippool` command supports the inclusion of IPv6 addresses in pools.

For more information see IPv6 for Solaris IP Filter. For tasks associated with IPv6 packet filtering, see Chapter 26, "Solaris IP Filter (Tasks)."

# Introduction to Solaris IP Filter

Solaris IP Filter replaces the SunScreen™ firewall as the firewall software for the Solaris Operating System (Solaris OS). Like the SunScreen firewall, Solaris IP Filter provides stateful packet filtering and network address translation (NAT). Solaris IP Filter also includes stateless packet filtering and the ability to create and manage address pools.

Packet filtering provides basic protection against network-based attacks. Solaris IP Filter can filter by IP address, port, protocol, network interface, and traffic direction. Solaris IP Filter can also filter by an individual source IP address, a destination IP address, by a range of IP addresses, or by address pools.

Solaris IP Filter is derived from open source IP Filter software. To view license terms, attribution, and copyright statements for open source IP Filter, the default path is `/usr/lib/ipf/IPFILTER.LICENCE`. If the Solaris OS has been installed anywhere other than the default, modify the given path to access the file at the installed location.

## Information Sources for Open Source IP Filter

The home page for the open source IP Filter software by Darren Reed is found at `http://coombs.anu.edu.au/~avalon/ip-filter.html`. This site includes information for open source IP Filter, including a link to a tutorial entitled "IP Filter Based Firewalls HOWTO"

(Brendan Conoboy and Erik Fichtner, 2002). This tutorial provides step-by-step instructions for building firewalls in a BSD UNIX environment. Although written for a BSD UNIX environment, the tutorial is also relevant for the configuration of Solaris IP Filter.

# Solaris IP Filter Packet Processing

Solaris IP Filter executes a sequence of steps as a packet is processed. The following diagram illustrates the steps of packet processing and how filtering integrates with the TCP/IP protocol stack.

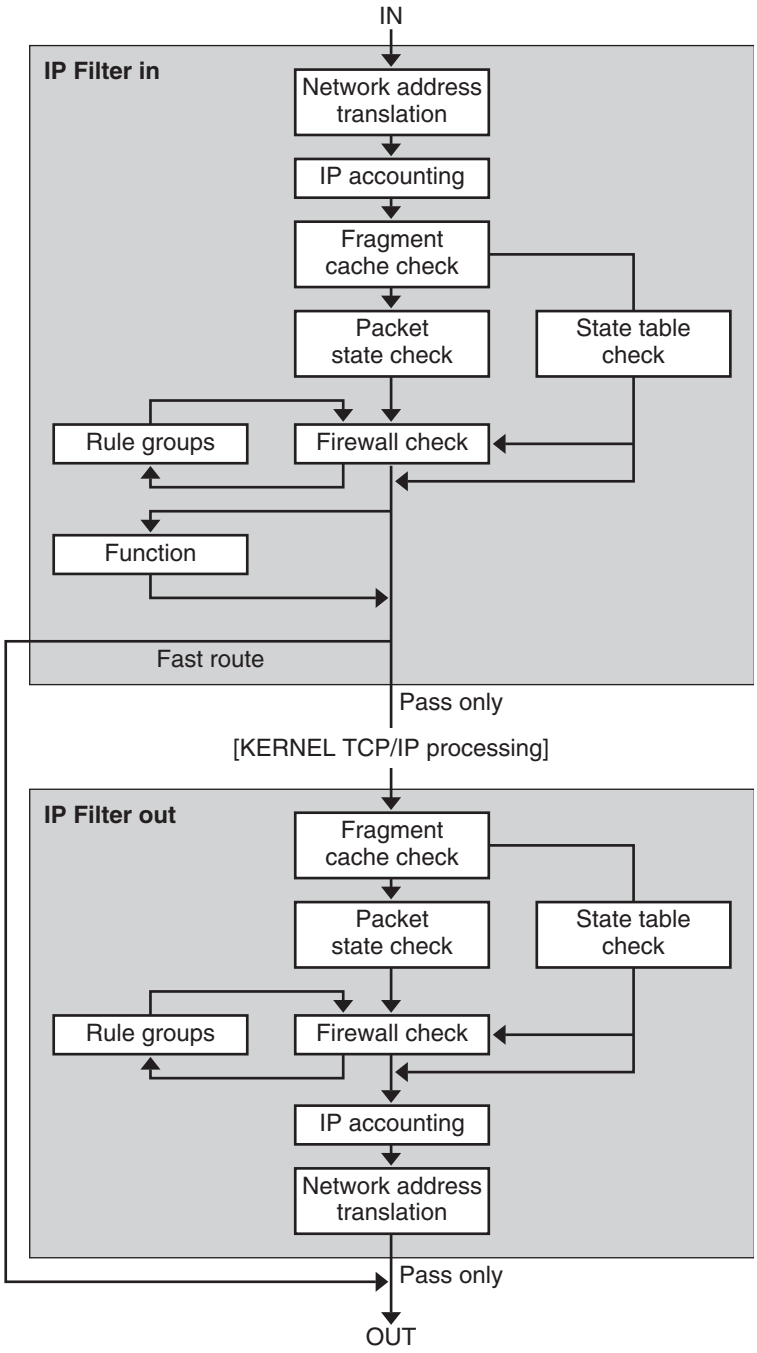



**FIGURE 25–1** Packet Processing Sequence

The packet processing sequence includes the following:

- **Network Address Translation (NAT)**

  The translation of a private IP address to a different public address, or the aliasing of multiple private addresses to a single public one. NAT allows an organization to resolve the problem of IP address depletion when the organization has existing networks and needs to access the Internet.

- **IP Accounting**

  Input and output rules can be separately set up, recording the number of bytes that pass through. Each time a rule match occurs, the byte count of the packet is added to the rule and allows for collection of cascading statistics.

- **Fragment Cache Check**

  If the next packet in the current traffic is a fragment and the previous packet was allowed, the packet fragment is also allowed, bypassing state table and rule checking.

- **Packet State Check**

  If `keep state` is included in a rule, all packets in a specified session are passed or blocked automatically, depending on whether the rule says `pass` or `block`.

- **Firewall Check**

  Input and output rules can be separately set up, determining whether or not a packet will be allowed through Solaris IP Filter, into the kernel's TCP/IP routines, or out onto the network.

- **Groups**

  Groups allow you to write your rule set in a tree fashion.

- **Function**

  A function is the action to be taken. Possible functions include `block`, `pass`, `literal`, and `send ICMP response`.

- **Fast-route**

  Fast-route signals Solaris IP Filter to not pass the packet into the UNIX IP stack for routing, which results in a TTL decrement.

- **IP Authentication**

  Packets that are authenticated are only passed through the firewall loops once to prevent double-processing.

# Guidelines for Using Solaris IP Filter

- Solaris IP Filter is managed by the SMF services svc:/network/pfil and svc:/network/ipfilter. For a complete overview of SMF, see Chapter 15, "Managing Services (Overview)," in *System Administration Guide: Basic Administration*. For information on the step-by-step procedures that are associated with SMF, see Chapter 16, "Managing Services (Tasks)," in *System Administration Guide: Basic Administration*.
- Solaris IP Filter requires direct editing of configuration files.
- Solaris IP Filter is installed as part of the Solaris OS. By default, Solaris IP Filter is not activated after a fresh install. To configure filtering, you must edit configuration files and manually activate Solaris IP Filter. You can activate filtering by either rebooting the system or by plumbing the interfaces using the `ifconfig` command. For more information, see the `ifconfig`(1M) man page. For the tasks associated with enabling Solaris IP Filter, see "Configuring Solaris IP Filter" on page 619.
- To administer Solaris IP Filter, you must be able to assume a role that includes the IP Filter Management rights profile, or become superuser. You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.
- IP Network Multipathing (IPMP) supports stateless filtering only.
- Sun Cluster configurations do not support filtering with Solaris IP Filter.
- Filtering between zones is not currently supported with Solaris IP Filter.

# Using Solaris IP Filter Configuration Files

Solaris IP Filter can be used to provide firewall services or network address translation (NAT). Solaris IP Filter can be implemented using loadable configuration files. Solaris IP Filter includes a directory called `/etc/ipf`. You can create and store configuration files called `ipf.conf`, `ipnat.conf` and `ippool.conf` in the `/etc/ipf` directory. These files are loaded automatically during the boot process when they reside in the `/etc/ipf` directory. You can also store the configuration files in another location and load the files manually. For example configuration files, see "Creating and Editing Solaris IP Filter Configuration Files" on page 651.

# Working With Solaris IP Filter Rule Sets

To manage your firewall, you use Solaris IP Filter to specify rule sets that you use to filter your network traffic. You can create the following types of rule sets:

- Packet filtering rule sets
- Network Address Translation (NAT) rule sets

Additionally, you can create address pools to reference groups of IP addresses. You can then use these pools later in a rule set. The address pools help to speed up rule processing. Address pools also make managing large groups of addresses easier.

# Using Solaris IP Filter's Packet Filtering Feature

You set up packet filtering by using packet filtering rule sets. Use the `ipf` command to work with packet filtering rule sets. For more information on the `ipf` command, see the `ipf`(1M) command.

You can create packet filtering rules either at the command line, using the `ipf` command, or in a packet filtering configuration file. If you want the packet filtering rules to be loaded at boot time, create a configuration file called `/etc/ipf/ipf.conf` in which to put packet filtering rules. If you do not want the packet filtering rules loaded at boot time, put the `ipf.conf` file in a location of your choice, and manually activate packet filtering by using the `ipf` command.

You can maintain two sets of packet filtering rule sets with Solaris IP Filter, the active rule set and the inactive rule set. In most cases, you work with the active rule set. However, the `ipf -I` command enables you to apply the command action to the inactive rule list. The inactive rule list is not used by Solaris IP Filter unless you select it. The inactive rule list provides you with a place to store rules without affecting active packet filtering.

Solaris IP Filter processes the rules in the rules list from the beginning of the configured rules list to the end of the rules list before passing or blocking a packet. Solaris IP Filter maintains a flag that determines whether it will or will not pass a packet. It goes through the entire rule set and determines whether to pass or block the packet based on the last matching rule.

There are two exceptions to this process. The first exception is if the packet matches a rule containing the `quick` keyword. If a rule includes the `quick` keyword, the action for that rule is taken, and no subsequent rules are checked. The second exception is if the packet matches a rule containing the `group` keyword. If a packet matches a group, only rules tagged with the group are checked.

## Configuring Packet Filtering Rules

Use the following syntax to create packet filtering rules:

*action* `[in|out]` *option keyword, keyword...*

1. Each rule begins with an action. Solaris IP Filter applies the action to the packet if the packet matches the rule. The following list includes the commonly used actions applied to a packet.

   | | |
   |---|---|
   | `block` | Prevents the packet from passing through the filter. |
   | `pass` | Allows the packet through the filter. |
   | `log` | Logs the packet but does not determine if the packet is blocked or passed. Use the `ipmon` command to view the log. |

| | |
|---|---|
| `count` | Includes the packet in the filter statistics. Use the `ipfstat` command to view the statistics. |
| `skip` *number* | Makes the filter skip over *number* filtering rules. |
| `auth` | Requests that packet authentication be performed by a user program that validates packet information. The program determines whether the packet is passed or blocked. |
| `preauth` | Requests that the filter look at a pre-authenticated list to determine what to do with the packet. |

2. Following the action, the next word must be either `in` or `out`. Your choice determines whether the packet filtering rule is applied to an incoming packet or to an outgoing packet.
3. Next, you can choose from a list of options. If you use more than one option, they must be in the order shown here.

| | |
|---|---|
| `log` | Logs the packet if the rule is the last matching rule. Use the `ipmon` command to view the log. |
| `quick` | Executes the rule containing the `quick` option if there is a packet match. All further rule checking stops. |
| `on` *interface-name* | Applies the rule only if the packet is moving in or out of the specified interface. |
| `dup-to` *interface-name* | Copies the packet and sends the duplicate out on *interface-name* to an optionally specified IP address. |
| `to` *interface-name* | Moves the packet to an outbound queue on *interface-name*. |

4. After specifying the options, you can choose from a variety of keywords that determine whether the packet matches the rule. The following keywords must be used in the order shown here.

---

**Note –** By default, any packet that does not match any rule in the configuration file is passed through the filter.

---

| | |
|---|---|
| `tos` | Filters the packet based on the type-of-service value expressed as either a hexadecimal or a decimal integer. |
| `ttl` | Matches the packet based on its time-to-live value. The time-to-live value stored in a packet indicates the length of time a packet can be on the network before being discarded. |

| | |
|---|---|
| `proto` | Matches a specific protocol. You can use any of the protocol names specified in the `/etc/protocols` file, or use a decimal number to represent the protocol. The keyword `tcp/udp` can be used to match either a TCP or a UDP packet. |
| `from/to/all/any` | Matches any or all of the following: the source IP address, the destination IP address, and the port number. The `all` keyword is used to accept packets from all sources and to all destinations. |
| `with` | Matches specified attributes associated with the packet. Insert either the word `not` or the word `no` in front of the keyword in order to match the packet only if the option is not present. |
| `flags` | Used for TCP to filter based on TCP flags that are set. For more information on the TCP flags, see the `ipf`(4) man page. |
| `icmp-type` | Filters according to ICMP type. This keyword is used only when the `proto` option is set to `icmp` and is not used if the `flags` option is used. |
| `keep` *keep-options* | Determines the information that is kept for a packet. The *keep-options* available include the `state` option and the `frags` option. The `state` option keeps information about the session and can be kept on TCP, UDP, and ICMP packets. The `frags` option keeps information on packet fragments and applies the information to later fragments. The *keep-options* allow matching packets to pass without going through the access control list. |
| `head` *number* | Creates a new group for filtering rules, which is denoted by the number *number*. |
| `group` *number* | Adds the rule to group number *number* instead of the default group. All filtering rules are placed in group 0 if no other group is specified. |

The following example illustrates how to put together the packet filtering rule syntax to create a rule. To block incoming traffic from the IP address `192.168.0.0/16`, you would include the following rule in the rule list:

```
block in quick from 192.168.0.0/16 to any
```

For the complete grammar and syntax used to write packet filtering rules, see the `ipf`(4) man page. For tasks associated with packet filtering, see "Managing Packet Filtering Rule Sets for Solaris IP Filter" on page 633. For an explanation of the IP address scheme (`192.168.0.0/16`) shown in the example, see Chapter 2, "Planning Your TCP/IP Network (Tasks)."

# Using Solaris IP Filter's NAT Feature

NAT sets up mapping rules that translate source and destination IP addresses into other Internet or intranet addresses. These rules modify the source and destination addresses of incoming or outgoing IP packets and send the packets on. You can also use NAT to redirect traffic from one port to another port. NAT maintains the integrity of the packet during any modification or redirection done on the packet.

Use the `ipnat` command to work with NAT rule lists. For more information on the `ipnat` command, see the `ipnat`(1M) command.

You can create NAT rules either at the command line, using the `ipnat` command, or in a NAT configuration file. NAT configuration rules reside in the `ipnat.conf` file. If you want the NAT rules to be loaded at boot time, create a file called `/etc/ipf/ipnat.conf` in which to put NAT rules. If you do not want the NAT rules loaded at boot time, put the `ipnat.conf` file in a location of your choice, and manually activate packet filtering with the `ipnat` command.

## Configuring NAT Rules

Use the following syntax to create NAT rules:

*command interface-name parameters*

1. Each rule begins with one of the following commands:

   | | |
   |---|---|
   | `map` | Maps one IP address or network to another IP address or network in an unregulated round-robin process. |
   | `rdr` | Redirects packets from one IP address and port pair to another IP address and port pair. |
   | `bimap` | Establishes a bidirectional NAT between an external IP address and an internal IP address. |
   | `map-block` | Establishes static IP address-based translation. This command is based on an algorithm that forces addresses to be translated into a destination range. |

2. Following the command, the next word is the interface name, such as `hme0`.
3. Next, you can choose from a variety of parameters, which determine the NAT configuration. Some of the parameters include:

   | | |
   |---|---|
   | `ipmask` | Designates the network mask. |
   | `dstipmask` | Designates the address that `ipmask` is translated to. |
   | `mapport` | Designates `tcp`, `udp`, or `tcp/udp` protocols, along with a range of port numbers. |

The following example illustrates how to put together the NAT rule syntax together to create a NAT rule. To rewrite a packet that goes out on the `de0` device with a source address of `192.168.1.0/24` and to externally show its source address as `10.1.0.0/16`, you would include the following rule in the NAT rule set:

```
map de0 192.168.1.0/24 -> 10.1.0.0/16
```

For the complete grammar and syntax used to write NAT rules, see the `ipnat`(4) man page.

# Using Solaris IP Filter's Address Pools Feature

Address pools establish a single reference that is used to name a group of address/netmask pairs. Address pools provide processes to reduce the time needed to match IP addresses with rules. Address pools also make managing large groups of addresses easier.

Address pool configuration rules reside in the `ippool.conf` file. If you want the address pool rules to be loaded at boot time, create a file called `/etc/ipf/ippool.conf` in which to put address pool rules. If you do not want the address pool rules loaded at boot time, put the `ippool.conf` file in a location of your choice, and manually activate packet filtering with the `ippool` command.

## Configuring Address Pools

Use the following syntax to create an address pool:

`table role =` *role-name* `type =` *storage-format* `number =` *reference-number*

`table` Defines the reference for the multiple addresses.

`role` Specifies the role of the pool in Solaris IP Filter. At this time, the only role you can reference is `ipf`.

`type` Specifies the storage format for the pool.

`number` Specifies the reference number that is used by the filtering rule.

For example, to reference the group of addresses `10.1.1.1` and `10.1.1.2`, and the network `192.16.1.0` as pool number `13`, you would include the following rule in the address pool configuration file:

```
table role = ipf type = tree number = 13
{ 10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24 };
```

Then, to reference pool number `13` in a filtering rule, you would construct the rule similar to the following example:

```
pass in from pool/13 to any
```

Note that you must load the pool file before loading the rules file that contains a reference to the pool. If you do not, the pool is undefined, as shown in the following output:

```
# ipfstat -io
empty list for ipfilter(out)
block in from pool/13(!) to any
```

Even if you add the pool later, the addition of the pool does not update the kernel rule set. You also need to reload the rules file that references the pool.

For the complete grammar and syntax used to write packet filtering rules, see the `ippool`(4) man page.

# Packet Filter Hooks

Beginning with the Solaris 10 8/07 release, packet filter hooks replace the `pfil` module to enable Solaris IP filter. In previous Solaris releases, configuration of the `pfil` module was required as an additional step to set up Solaris IP Filter. This extra configuration requirement increased the risk of errors that would cause Solaris IP Filter to work improperly. The insertion of the `pfil` STREAMS module between IP and the device driver also caused performance degradation. Lastly, the `pfil` module could not perform packet interception between zones.

The use of packet filter hooks streamlines the procedure to enable Solaris IP Filter. Through these hooks, Solaris IP Filter uses pre-routing (input) and post-routing (output) filter taps to control packet flow into and out of the Solaris system.

Packet filter hooks eliminate the need for the `pfil` module. Thus the following components that are associated with the module are also removed.

- `pfil` driver
- `pfil` daemon
- `svc:/network/pfil` SMF service

For tasks associated with enabling Solaris IP Filter, see Chapter 26, “Solaris IP Filter (Tasks).”

# Solaris IP Filter and the `pfil` STREAMS Module

**Note –** The `pfil` module is used with Solaris IP filter only on the following Solaris 10 releases:

- Solaris 10 3/05 release
- Solaris 10 1/06 release
- Solaris 10 6/06 release
- Solaris 10 11/06 release

Beginning with the Solaris 10 8/07 release, the `pfil` module has been replaced by packet filter hooks and is no longer used with Solaris IP filter.

The `pfil` STREAMS module is required to enable Solaris IP Filter. However, Solaris IP Filter does not provide an automatic mechanism to push the module on to every interface. Instead, the `pfil` STREAMS module is managed by the SMF service `svc:/network/pfil`. To activate filtering on a network interface, you first configure the `pfil.ap` file. Then you activate the `svc:/network/pfil` service to supply the `pfil` STREAMS module to the network interface. For the STREAMS module to take effect, the system must be rebooted or each network interface on which you want filtering must be unplumbed and then re-plumbed. To activate IPv6 packet filtering capabilities, you need to plumb the `inet6` version of the interface. For tasks associated with activating Solaris IP Filter, see "Configuring Solaris IP Filter" on page 619.

If no `pfil` modules are found for the network interfaces, the SMF services are put into a maintenance state. The most common cause of this situation is an incorrectly edited `/etc/ipf/pfil.ap` file. If the service is put into maintenance mode, the occurrence is logged in the filtering log files.

For tasks associated with activating Solaris IP Filter, see "Configuring Solaris IP Filter" on page 619.

# IPv6 for Solaris IP Filter

Beginning with the Solaris 10 6/06 release, support for IPv6 is available with Solaris IP Filter. IPv6 packet filtering can filter based on the source/destination IPv6 address, pools containing IPv6 addresses, and IPv6 extension headers.

IPv6 is similar to IPv4 in many ways. However, header and packet size differ between the two versions of IP, which is an important consideration for IP Filter. IPv6 packets known as *jumbograms* contain a datagram longer than 65,535 bytes. Solaris IP Filter does not support IPv6 jumbograms. To learn more about other IPv6 features, see "Major Features of IPv6" on page 69.

**Note –** For more information on jumbograms, refer to the document IPv6 Jumbograms, RFC 2675 from the Internet Engineering Task Force (IETF). [http://www.ietf.org/rfc/rfc2675.txt]

IP Filter tasks associated with IPv6 do not differ substantially from IPv4. The most notable difference is the use of the `-6` option with certain commands. Both the `ipf` command and the `ipfstat` command include the `-6` option for use with IPv6 packet filtering. Use the `-6` option with the `ipf` command to load and flush IPv6 packet filtering rules. To display IPv6 statistics, use the `-6` option with the `ipfstat` command. The `ipmon` and `ippool` commands also support IPv6, although there is no associated option for IPv6 support. The `ipmon` command has been enhanced to accommodate the logging of IPv6 packets. The `ippool` command supports the pools with IPv6 addresses. You can create pools of only IPv4 or IPv6 addresses, or a pool containing both IPv4 and IPv6 addresses within the same pool.

You can use the `ipf6.conf` file to create packet filtering rule sets for IPv6. By default, the `ipf6.conf` configuration file is included in the `/etc/ipf` directory. As with the other filtering configuration files, the `ipf6.conf` file loads automatically during the boot process when it is stored in the `/etc/ipf` directory. You can also create and store an IPv6 configuration file in another location and load the file manually.

**Note –** Network Address Translation (NAT) does not support IPv6.

Once packet filtering rules for IPv6 have been set up, activate IPv6 packet filtering capabilities by plumbing the `inet6` version of the interface.

For more information on IPv6, see Chapter 3, “Introducing IPv6 (Overview).” For tasks associated with Solaris IP Filter, see Chapter 26, “Solaris IP Filter (Tasks).”

# Solaris IP Filter Man Pages

The following table includes the man page documentation relevant to Solaris IP Filter.

| Man Page | Description |
|---|---|
| ipf(1M) | Use the ipf command to complete the following tasks:<br>■ Work with packet filtering rule sets.<br>■ Disable and enable filtering.<br>■ Reset statistics and resynchronize the in-kernel interface list with the current interface status list. |
| ipf(4) | Contains the grammar and syntax for creating Solaris IP Filter packet filtering rules. |
| ipfilter(5) | Provides open source IP Filter licensing information. |
| ipfs(1M) | Use the ipfs command to save and restore NAT information and state table information across reboots. |
| ipfstat(1M) | Use the ipfstat command to retrieve and display statistics on packet processing. |
| ipmon(1M) | Use the ipmon command to open the log device and view logged packets for both packet filtering and NAT. |
| ipnat(1M) | Use the ipnat command to complete the following tasks:<br>■ Work with NAT rules.<br>■ Retrieve and display NAT statistics. |
| ipnat(4) | Contains the grammar and syntax for creating NAT rules. |
| ippool(1M) | Use the ippool command to create and manage address pools. |
| ippool(4) | Contains the grammar and syntax for creating Solaris IP Filter address pools. |
| ndd(1M) | Displays current filtering parameters of the pfil STREAMS module and the current values of the tunable parameters. |

◆ ◆ ◆ CHAPTER 26

# Solaris IP Filter (Tasks)

This chapter provides step-by-step instructions for Solaris IP Filter tasks. For overview information about Solaris IP Filter, see Chapter 25, "Solaris IP Filter (Overview)."

This chapter contains the following information:


- "Configuring Solaris IP Filter" on page 619
- "Deactivating and Disabling Solaris IP Filter" on page 623
- "Working With the `pfil` Module" on page 625
- "Working With Solaris IP Filter Rule Sets" on page 632
- "Displaying Statistics and Information for Solaris IP Filter" on page 644
- "Working With Log Files for Solaris IP Filter" on page 647
- "Creating and Editing Solaris IP Filter Configuration Files" on page 651


## Configuring Solaris IP Filter

The following task map identifies the procedures associated with configuring Solaris IP Filter.

**TABLE 26–1** Configuring Solaris IP Filter (Task Map)

| Task | Description | For Instructions |
|---|---|---|
| Initially enable Solaris IP Filter. | Solaris IP Filter is not enabled by default. You must either enable it manually or use the configuration files in the `/etc/ipf/` directory and reboot the system. Beginning with Solaris 10 8/07 release, packet filter hooks replaced the `pfil` module to enable Solaris IP filter. | "How to Enable Solaris IP Filter" on page 620 |

**TABLE 26–1** Configuring Solaris IP Filter (Task Map) *(Continued)*

| Task | Description | For Instructions |
|---|---|---|
| Re-enable Solaris IP Filter. | If Solaris IP Filter is deactivated or disabled, you can re-enable Solaris IP Filter either by rebooting the system or by using the `ipf` command. | "How to Re-Enable Solaris IP Filter" on page 621 |
| Enable loopback filtering | As an option, you can enable loopback filtering, for example, to filter traffic between zones. | "How to Enable Loopback Filtering" on page 622 |

## ▼ How to Enable Solaris IP Filter

Use this procedure to enable Solaris IP Filter on a system that is running at least Solaris 10 8/07 OS.

To enable Solaris IP Filters if your system is running a Solaris 10 release previous to Solaris 10 8/07 OS, see "Working With the `pfil` Module" on page 625.

**1 Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**2 Create a packet filtering rule set.**

The packet filtering rule set contains packet filtering rules that are used by Solaris IP Filter. If you want the packet filtering rules to be loaded at boot time, edit the `/etc/ipf/ipf.conf` file to implement IPv4 packet filtering. Use the `/etc/ipf/ipf6.conf` file for IPv6 packet filtering rules. If you do not want the packet filtering rules loaded at boot time, put the rules in a file of your choice, and manually activate packet filtering. For information about packet filtering, see "Using Solaris IP Filter's Packet Filtering Feature" on page 609. For information about working with configuration files, see "Creating and Editing Solaris IP Filter Configuration Files" on page 651.

**3 (Optional) Create a network address translation (NAT) configuration file.**

---

**Note –** Network Address Translation (NAT) does not support IPv6.

---

Create an `ipnat.conf` file if you want to use network address translation. If you want the NAT rules to be loaded at boot time, create a file called `/etc/ipf/ipnat.conf` in which to put NAT rules. If you do not want the NAT rules loaded at boot time, put the `ipnat.conf` file in a location of your choice, and manually activate the NAT rules.

For more information about NAT, see "Using Solaris IP Filter's NAT Feature" on page 612.

4 **(Optional) Create an address pool configuration file.**

Create an `ipool.conf` file if you want to refer to a group of addresses as a single address pool. If you want the address pool configuration file to be loaded at boot time, create a file called `/etc/ipf/ippool.conf` in which to put the address pool. If you do not want the address pool configuration file to be loaded at boot time, put the `ippool.conf` file in a location of your choice, and manually activate the rules.

An address pool can contain only IPv4 addresses or only IPv6 addresses. It can also contain both IPv4 and IPv6 addresses.

For more information about address pools, see "Using Solaris IP Filter's Address Pools Feature" on page 613.

5 **(Optional) Enable filtering of loopback traffic.**

If you intend to filter traffic between zones that are configured in your system, you must enable loopback filtering. See "How to Enable Loopback Filtering" on page 622. Make sure that you also define the appropriate rule sets that apply to the zones.

6 **Activate Solaris IP Filter.**

```
# svcadm enable network/ipfilter
```

## How to Re-Enable Solaris IP Filter

You can re-enable packet filtering after it has been temporarily disabled.

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Enable Solaris IP Filter and activate filtering using one of the following methods:**

- Reboot the machine.

  ```
  # reboot
  ```

  **Note –** When IP Filter is enabled, after a reboot the following files are loaded if they are present: the `/etc/ipf/ipf.conf` file, the `/etc/ipf/ipf6.conf` file when using IPv6, or the `/etc/ipf/ipnat.conf`.

- Perform the following series of commands to enable Solaris IP Filter and activate filtering:

  a. Enable Solaris IP Filter.

```
# ipf -E
```

b. Activate packet filtering.

```
# ipf -f filename
```

c. (Optional) Activate NAT.

```
# ipnat -f filename
```

**Note –** Network Address Translation (NAT) does not support IPv6.

## ▼ How to Enable Loopback Filtering

**Note –** You can filter loopback traffic only if your system is running at least Solaris 10 8/07. In previous Solaris 10 releases, loopback filtering is not supported.

**1 Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**2 Stop Solaris IP Filter if it is running.**

```
# svcadm disable network/ipfilter
```

**3 Edit the `/etc/ipf.conf` or `/etc/ipf6.conf` file by adding the following line at the beginning of the file:**

```
set intercept_loopback true;
```

This line must precede all the IP filter rules that are defined in the file. However, you can insert comments before the line, similar to the following example:

```
#
# Enable loopback filtering to filter between zones
#
set intercept_loopback true;
#
# Define policy
#
block in all
block out all
<other rules>
...
```

4 **Start the Solaris IP filter.**

```
# svcadm enable network/ipfilter
```

5 **To verify the status of loopback filtering, use the following command:**

```
# ipf –T ipf_loopback
ipf_loopback    min 0   max 0x1 current 1
#
```

If loopback filtering is disabled, the command would generate the following output:

```
ipf_loopback    min 0   max 0x1 current 0
```

# Deactivating and Disabling Solaris IP Filter

You might want to deactivate or disable packet filtering and NAT under the following circumstances:

- For testing purposes
- To troubleshoot system problems when you think the problems are caused by Solaris IP Filter

The following task map identifies the procedures associated with deactivating or disabling Solaris IP Filter features.

**TABLE 26–2** Deactivating and Disabling Solaris IP Filter (Task Map)

| Task | Description | For Instructions |
|---|---|---|
| Deactivate packet filtering. | Deactivate packet filtering using the `ipf` command. | "How to Deactivate Packet Filtering" on page 623 |
| Deactivate NAT. | Deactivate NAT using the `ipnat` command. | "How to Deactivate NAT" on page 624 |
| Disable packet filtering and NAT. | Disable packet filtering and NAT using the `ipf` command. | "How to Disable Packet Filtering" on page 625 |

## ▼ How to Deactivate Packet Filtering

The following procedure deactivates Solaris IP Filter packet filtering by flushing the packet filtering rules from the active filtering rule set. The procedure does not disable Solaris IP Filter. You can reactivate Solaris IP Filter by adding rules to the rule set.

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Use one of the following methods to deactivate Solaris IP Filter rules:**

   - Remove the active rule set from the kernel.

     ```
     # ipf -Fa
     ```

     This command deactivates all packet filtering rules.
   - Remove incoming packet filtering rules.

     ```
     # ipf -Fi
     ```

     This command deactivates packet filtering rules for incoming packets.
   - Remove outgoing packet filtering rules.

     ```
     # ipf -Fo
     ```

     This command deactivates packet filtering rules for outgoing packets.

## ▼ How to Deactivate NAT

The following procedure deactivates Solaris IP Filter NAT rules by flushing the NAT rules from the active NAT rules set. The procedure does not disable Solaris IP Filter. You can reactivate Solaris IP Filter by adding rules to the rule set.

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Remove NAT from the kernel.**

   ```
   # ipnat -FC
   ```

   The `-C` option removes all entries in the current NAT rule listing. The `-F` option removes all active entries in the current NAT translation table, which shows the currently active NAT mappings.

### ▼ How to Disable Packet Filtering

When you run this procedure, both packet filtering and NAT are removed from the kernel. If you use this procedure, you must re-enable Solaris IP Filter in order to reactivate packet filtering and NAT. For more information, see "How to Re-Enable Solaris IP Filter" on page 621.

**1 Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**2 Disable packet filtering and allow all packets to pass into the network.**

```
# ipf –D
```

---

**Note –** The `ipf -D` command flushes the rules from the rule set. When you re-enable filtering, you must add rules to the rule set.

---

## Working With the `pfil` Module

This section describes how to use the `pfil` STREAMS module to activate or deactivate Solaris IP Filter and how to view `pfil` statistics. The procedures apply only to systems that run one of the following Solaris 10 releases:

- Solaris 10 3/05 release
- Solaris 10 1/06 release
- Solaris 10 6/06 release
- Solaris 10 11/06 release

The following task map identifies procedures that are associated with configuring the `pfil` module.

**TABLE 26–3** Working With the `pfil` Module (Task Map)

| Task | Description | For Instructions |
|---|---|---|
| Enable Solaris IP Filter | Solaris IP Filter is not enabled by default. You must either enable it manually or use the configuration files in the `/etc/ipf/` directory and reboot the system. | "How to Enable Solaris IP Filter in Previous Solaris 10 Releases" on page 626 |
| Activate a NIC for packet filtering | Configure the `pfil` module to activate packet filtering on a NIC | "How to Activate a NIC for Packet Filtering" on page 628 |

TABLE 26–3 Working With the pfil Module (Task Map) *(Continued)*

| Task | Description | For Instructions |
|---|---|---|
| Deactivate Solaris IP Filter on a NIC | Remove a NIC and allow all packets to pass through the NIC. | "How to Deactivate Solaris IP Filter on a NIC" on page 630 |
| View pfil statistics. | View statistics for the pfil module to help you troubleshoot Solaris IP Filter using the ndd command. | "How to View pfil Statistics for Solaris IP Filter" on page 631 |

## ▼ How to Enable Solaris IP Filter in Previous Solaris 10 Releases

Solaris IP Filter is installed with the Solaris OS. However, packet filtering is not enabled by default. Use the following procedure to activate Solaris IP Filter.

**Note –** If your system is running at least the Solaris 10 8/07 release, follow the procedure that uses packet filter hooks.

**1 Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**2 Start the file editor of your choice, and edit the /etc/ipf/pfil.ap file.**

This file contains the names of network interface cards (NICs) on the host. By default, the names are commented out. Uncomment the device names that carry the network traffic you want to filter. If the name of the NIC for your system is not listed, add a line to specify the NIC.

```
# vi /etc/ipf/pfil.ap
# IP Filter pfil autopush setup
#
# See autopush(1M) manpage for more information.
#
# Format of the entries in this file is:
#
#major  minor lastminor modules

#le     -1      0       pfil
#qe     -1      0       pfil
hme     -1      0       pfil (Device has been uncommented for filtering)
#qfe    -1      0       pfil
#eri    -1      0       pfil
#ce     -1      0       pfil
#bge    -1      0       pfil
```

```
#be     -1      0       pfil
#vge    -1      0       pfil
#ge     -1      0       pfil
#nf     -1      0       pfil
#fa     -1      0       pfil
#ci     -1      0       pfil
#el     -1      0       pfil
#ipdptp -1      0       pfil
#lane   -1      0       pfil
#dmfe   -1      0       pfil
```

3 **Activate your changes to the `/etc/ipf/pfil.ap` file by restarting the `network/pfil` service instance.**

```
# svcadm restart network/pfil
```

4 **Create a packet filtering rule set.**

The packet filtering rule set contains packet filtering rules that are used by Solaris IP Filter. If you want the packet filtering rules to be loaded at boot time, edit the `/etc/ipf/ipf.conf` file to implement IPv4 packet filtering. Use the `/etc/ipf/ipf6.conf` file for IPv6 packet filtering rules. If you do not want the packet filtering rules loaded at boot time, put the rules in a file of your choice, and manually activate packet filtering. For information about packet filtering, see "Using Solaris IP Filter's Packet Filtering Feature" on page 609. For information about working with configuration files, see "Creating and Editing Solaris IP Filter Configuration Files" on page 651.

5 **(Optional) Create a network address translation (NAT) configuration file.**

---

**Note –** Network Address Translation (NAT) does not support IPv6.

---

Create an `ipnat.conf` file if you want to use network address translation. If you want the NAT rules to be loaded at boot time, create a file called `/etc/ipf/ipnat.conf` in which to put NAT rules. If you do not want the NAT rules loaded at boot time, put the `ipnat.conf` file in a location of your choice, and manually activate the NAT rules.

For more information about NAT, see "Using Solaris IP Filter's NAT Feature" on page 612.

6 **(Optional) Create an address pool configuration file.**

Create an `ipool.conf` file if you want to refer to a group of addresses as a single address pool. If you want the address pool configuration file to be loaded at boot time, create a file called `/etc/ipf/ippool.conf` in which to put the address pool. If you do not want the address pool configuration file to be loaded at boot time, put the `ippool.conf` file in a location of your choice, and manually activate the rules.

An address pool can contain only IPv4 addresses or only IPv6 addresses. It can also contain both IPv4 and IPv6 addresses.

For more information about address pools, see "Using Solaris IP Filter's Address Pools Feature" on page 613.

**7 Activate Solaris IP Filter by using one of the following methods:**

- Enable IP Filter and reboot the machine.

```
# svcadm enable network/ipfilter
# reboot
```

**Note –** Rebooting is required if you cannot safely use the `ifconfig unplumb` and `ifconfig plumb` commands on the NICs.

- Enable the NICs by using the `ifconfig unplumb` and `ifconfig plumb` commands. Then enable IP Filter. The `inet6` version of the interface must be plumbed in order to implement IPv6 packet filtering.

```
# ifconfig hme0 unplumb
# ifconfig hme0 plumb 192.168.1.20 netmask 255.255.255.0 up
# ifconfig hme0 inte6 unplumb
# ifconfig hme0 inet6 plumb fec3:f849::1/96 up
# svcadm enable network/ipfilter
```

For more information about the `ifconfig` command, see the `ifconfig`(1M) man page.

## ▼ How to Activate a NIC for Packet Filtering

Solaris IP Filter is enabled at boot time when the `/etc/ipf/ipf.conf` file (or the `/etc/ipf/ipf6.conf` file when using IPv6) exists. If you need to enable filtering on a NIC after Solaris IP Filter is enabled, use the following procedure.

**1 Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**2 Start the file editor of your choice, and edit the `/etc/ipf/pfil.ap` file.**

This file contains the names of NICs on the host. By default, the names are commented out. Uncomment the device names that carry the network traffic you want to filter. If the name of the NIC for your system is not listed, add a line to specify the NIC.

```
# vi /etc/ipf/pfil.ap
# IP Filter pfil autopush setup
#
```

```
# See autopush(1M) manpage for more information.
#
# Format of the entries in this file is:
#
#major  minor lastminor modules

#le     -1      0       pfil
#qe     -1      0       pfil
hme     -1      0       pfil (Device has been uncommented for filtering)
#qfe    -1      0       pfil
#eri    -1      0       pfil
#ce     -1      0       pfil
#bge    -1      0       pfil
#be     -1      0       pfil
#vge    -1      0       pfil
#ge     -1      0       pfil
#nf     -1      0       pfil
#fa     -1      0       pfil
#ci     -1      0       pfil
#el     -1      0       pfil
#ipdptp -1      0       pfil
#lane   -1      0       pfil
#dmfe   -1      0       pfil
```

**3 Activate your changes to the `/etc/ipf/pfil.ap` file by restarting the `network/pfil` service instance.**

```
# svcadm restart network/pfil
```

**4 Enable the NIC by using one of the following methods:**

- Reboot the machine.

  ```
  # reboot
  ```

  **Note –** Rebooting is required if you cannot safely use the `ifconfig unplumb` and `ifconfig plumb` commands on the NICs.

- Enable the NICs that you want to filter by using the `ifconfig` command with the `unplumb` and `plumb` options. The `inet6` version of each interface must be plumbed in order to implement IPv6 packet filtering.

  ```
  # ifconfig hme0 unplumb
  # ifconfig hme0 plumb 192.168.1.20  netmask 255.255.255.0  up
  # ifconfig hme0 inet6 unplumb
  # ifconfig hme0 inet6 plumb fec3:f840::1/96 up
  ```

For more information about the ifconfig command, see the ifconfig(1M) man page.

## How to Deactivate Solaris IP Filter on a NIC

If you need to stop filtering packets on a NIC, use the following procedure.

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Start the file editor of your choice, and edit the /etc/ipf/pfil.ap file.**

This file contains the names of NICs on the host. The NICs that have been used to filter network traffic are uncommented. Comment out the device names that you no longer want to use to filter network traffic.

```
# vi /etc/ipf/pfil.ap
# IP Filter pfil autopush setup
#
# See autopush(1M) manpage for more information.
#
# Format of the entries in this file is:
#
#major  minor lastminor modules

#le     -1      0       pfil
#qe     -1      0       pfil
#hme    -1      0       pfil (Commented-out device no longer filters network traffic)
#qfe    -1      0       pfil
#eri    -1      0       pfil
#ce     -1      0       pfil
#bge    -1      0       pfil
#be     -1      0       pfil
#vge    -1      0       pfil
#ge     -1      0       pfil
#nf     -1      0       pfil
#fa     -1      0       pfil
#ci     -1      0       pfil
#el     -1      0       pfil
#ipdptp -1      0       pfil
#lane   -1      0       pfil
#dmfe   -1      0       pfil
```

3 **Deactivate the NIC by using one of the following methods:**

- Reboot the machine.

```
# reboot
```

**Note –** Rebooting is required if you cannot safely use the `ifconfig unplumb` and `ifconfig plumb` commands on the NICs.

- Deactivate the NICs by using the `ifconfig` command with the `unplumb` and `plumb` options. The `inet6` version of each interface must be unplumbed in order to deactivate IPv6 packet filtering. Perform the following steps. The sample device in the system is `hme`:

  a. Identify the major number for the device you are deactivating.

  ```
  # grep hme /etc/name_to_major
  hme 7
  ```

  b. Display the current `autopush` configuration for `hme0`.

  ```
  # autopush -g -M 7 -m 0
     Major     Minor     Lastminor       Modules
         7       ALL           -           pfil
  ```

  c. Remove the `autopush` configuration.

  ```
  # autopush -r -M 7 -m 0
  ```

  d. Open the device and assign IP addresses to the device.

  ```
  # ifconfig hme0 unplumb
  # ifconfig hme0 plumb 192.168.1.20  netmask 255.255.255.0  up
  # ifconfig hme0 inet6 unplumb
  # ifconfig hme0 inet6 plumb fec3:f840::1/96 up
  ```

  For more information about the `ifconfig` command, see the `ifconfig`(1M) man page.

## ▼ How to View `pfil` Statistics for Solaris IP Filter

You can view `pfil` statistics when you are troubleshooting Solaris IP Filter.

**1 Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**2 View `pfil` statistics.**

```
# ndd -get /dev/pfil qif_status
```

**Example 26–1** Viewing `pfil` Statistics for Solaris IP Filter

The following example shows how to view **`pfil`** statistics.

```
# ndd -get /dev/pfil qif_status
ifname ill q OTHERQ num sap hl nr nw bad copy copyfail drop notip nodata
   notdata
QIF6 0 300011247b8 300011248b0 6 806 0 4 9 0 0 0 0 0 0 0 0
dmfe1 3000200a018 30002162a50 30002162b48 5 800 14 171 13681 0 0 0 0 0 0 0
```

# Working With Solaris IP Filter Rule Sets

The following task map identifies the procedures associated with Solaris IP Filter rule sets.

**TABLE 26–4** Working With Solaris IP Filter Rule Sets (Task Map)

| Task | Description | For Instructions |
|---|---|---|
| Manage, view and modify Solaris IP Filter packet filtering rule sets. | | "Managing Packet Filtering Rule Sets for Solaris IP Filter" on page 633 |
| | View an active packet filtering rule set. | "How to View the Active Packet Filtering Rule Set" on page 633 |
| | View an inactive packet filtering rule set. | "How to View the Inactive Packet Filtering Rule Set" on page 634 |
| | Activate a different active rule set. | "How to Activate a Different or Updated Packet Filtering Rule Set" on page 634 |
| | Remove a rule set. | "How to Remove a Packet Filtering Rule Set" on page 636 |
| | Add rules to the rule sets. | "How to Append Rules to the Active Packet Filtering Rule Set" on page 636<br>"How to Append Rules to the Inactive Packet Filtering Rule Set" on page 637 |
| | Move between active and inactive rule sets. | "How to Switch Between Active and Inactive Packet Filtering Rule Sets" on page 638 |

**TABLE 26–4** Working With Solaris IP Filter Rule Sets (Task Map) *(Continued)*

| Task | Description | For Instructions |
|---|---|---|
| | Delete an inactive rule set from the kernel. | "How to Remove an Inactive Packet Filtering Rule Set From the Kernel" on page 639 |
| Manage, view and modify Solaris IP Filter NAT rules. | | "Managing NAT Rules for Solaris IP Filter" on page 640 |
| | View active NAT rules. | "How to View Active NAT Rules" on page 640 |
| | Remove NAT rules. | "How to Remove NAT Rules" on page 640 |
| | Add additional rules to NAT rules. | "How to Append Rules to the NAT Rules" on page 641 |
| Manage, view and modify Solaris IP Filter address pools. | | "Managing Address Pools for Solaris IP Filter" on page 642 |
| | View active address pools. | "How to View Active Address Pools" on page 642 |
| | Remove an address pool. | "How to Remove an Address Pool" on page 642 |
| | Add additional rules to an address pool. | "How to Append Rules to an Address Pool" on page 643 |

# Managing Packet Filtering Rule Sets for Solaris IP Filter

When Solaris IP Filter is enabled, both active and inactive packet filtering rule sets can reside in the kernel. The active rule set determines what filtering is being done on incoming packets and outgoing packets. The inactive rule set also stores rules. These rules are not used unless you make the inactive rule set the active rule set. You can manage, view, and modify both active and inactive packet filtering rule sets.

## ▼ How to View the Active Packet Filtering Rule Set

**1 Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **View the active packet filtering rule set that is loaded in the kernel.**

```
# ipfstat -io
```

**Example 26–2** Viewing the Active Packet Filtering Rule Set

The following example shows output from the active packet filtering rule set that is loaded in the kernel.

```
# ipfstat -io
empty list for ipfilter(out)
pass in quick on dmfe1 from 192.168.1.0/24 to any
pass in all
block in on dmfe1 from 192.168.1.10/32 to any
```

## ▼ How to View the Inactive Packet Filtering Rule Set

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **View the inactive packet filtering rule set.**

```
# ipfstat -I -io
```

**Example 26–3** Viewing the Inactive Packet Filtering Rule Set

The following example shows output from the inactive packet filtering rule set.

```
# ipfstat -I -io
pass out quick on dmfe1 all
pass in quick on dmfe1 all
```

## ▼ How to Activate a Different or Updated Packet Filtering Rule Set

Use the following procedure if you want to perform either of the following tasks:

- Activate a packet filtering rule set other than the one that is currently in use by Solaris IP Filter.
- Reload the same filtering rule set that has been newly updated.

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Choose one of the following steps:**
   - Create a new rule set in a separate file of your choice if you want to activate an entirely different rule set.
   - Update the current rule set by editing the configuration file that contains that rule set.

3. **Remove the current rule set and load the new rule set.**

   `# ipf -Fa -f` *filename*

   The *filename* can either be the new file with the new rule set or the updated file that contains the active rule set.

   The active rule set is removed from the kernel. The rules in the *filename* file become the active rule set.

---

**Note –** You still need to issue the command even if you are reloading the current configuration file. Otherwise, the old rule set continues to be operative, and the modified rule set in the updated configuration file is not applied.

Do not use commands such as `ipf -D` or `svcadm restart` to load the updated rule set. Such commands expose your network by disabling the firewall first before loading the new rule set.

---

**Example 26–4** Activating a Different Packet Filtering Rule Set

The following example shows how to replace one packet filtering rule set with another packet filtering rule set in a separate configuration file, `/etc/ipf/ipf.conf`.

```
# ipfstat -io
empty list for ipfilter(out)
pass in quick on dmfe all
# ipf -Fa -f /etc/ipf/ipf.conf
# ipfstat -io
empty list for ipfilter(out)
block in log quick from 10.0.0.0/8 to any
```

**Example 26–5** Reloading an Updated Packet Filtering Rule Set

The following example shows how to reload a packet filtering rule set that is currently active and which is then updated. In this example, the file in use is `/etc/ipf/ipf.conf`.

```
# ipfstat -io (Optional)
empty list for ipfilter (out)
block in log quick from 10.0.0.0/8 to any

(Edit the /etc/ipf/ipf.conf configuration file.)
```

```
# ip -Fa -f /etc/ipf/ipf.conf
# ipfstat -io (Optional)
empty list for ipfilter (out)
block in log quick from 10.0.0.0/8 to any
block in quick on elx10 from 192.168.0.0/12 to any
```

## How to Remove a Packet Filtering Rule Set

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Remove the rule set.**

   ```
   # ipf -F [a|i|o]
   ```

   `-a` Removes all filtering rules from the rule set.

   `-i` Removes the filtering rules for incoming packets.

   `-o` Removes the filtering rules for outgoing packets.

**Example 26–6** Removing a Packet Filtering Rule Set

The following example shows how to remove all filtering rules from the active filtering rule set.

```
# ipfstat -io
block out log on dmf0 all
block in log quick from 10.0.0.0/8 to any
# ipf -Fa
# ipfstat -io
empty list for ipfilter(out)
empty list for ipfilter(in)
```

## How to Append Rules to the Active Packet Filtering Rule Set

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Use one of the following methods to append rules to the active rule set:**

   - Append rules to the rule set at the command line using the `ipf -f -` command.

```
# echo "block in on dmfe1 proto tcp from 10.1.1.1/32 to any" | ipf -f -
```

- Perform the following commands:
  a. Create a rule set in a file of your choice.
  b. Add the rules you have created to the active rule set.

     **# ipf -f** *filename*

     The rules in *filename* are added to the end of the active rule set. Because Solaris IP Filter uses a "last matching rule" algorithm, the added rules determine filtering priorities, unless you use the `quick` keyword. If the packet matches a rule containing the `quick` keyword, the action for that rule is taken, and no subsequent rules are checked.

**Example 26–7** Appending Rules to the Active Packet Filtering Rule Set

The following example shows how to add a rule to the active packet filtering rule set from the command line.

```
# ipfstat -io
empty list for ipfilter(out)
block in log quick from 10.0.0.0/8 to any
# echo "block in on dmfe1 proto tcp from 10.1.1.1/32 to any" | ipf -f -
# ipfstat -io
empty list for ipfilter(out)
block in log quick from 10.0.0.0/8 to any
block in on dmfe1 proto tcp from 10.1.1.1/32 to any
```

## ▼ How to Append Rules to the Inactive Packet Filtering Rule Set

**1 Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**2 Create a rule set in a file of your choice.**

**3 Add the rules you have created to the inactive rule set.**

**# ipf -I -f** *filename*

The rules in *filename* are added to the end of the inactive rule set. Because Solaris IP Filter uses a "last matching rule" algorithm, the added rules determine filtering priorities, unless you use the `quick` keyword. If the packet matches a rule containing the `quick` keyword, the action for that rule is taken, and no subsequent rules are checked.

**Example 26–8** Appending Rules to the Inactive Rule Set

The following example shows how to add a rule to the inactive rule set from a file.

```
# ipfstat -I -io
pass out quick on dmfe1 all
pass in quick on dmfe1 all
# ipf -I -f /etc/ipf/ipf.conf
# ipfstat -I -io
pass out quick on dmfe1 all
pass in quick on dmfe1 all
block in log quick from 10.0.0.0/8 to any
```

## ▼ How to Switch Between Active and Inactive Packet Filtering Rule Sets

**1 Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**2 Switch the active and inactive rule sets.**

```
# ipf -s
```

This command enables you to switch between the active and inactive rule sets in the kernel. Note that if the inactive rule set is empty, there is no packet filtering.

**Example 26–9** Switching Between the Active and Inactive Packet Filtering Rule Sets

The following example shows how using the `ipf -s` command results in the inactive rule set becoming the active rule set and the active rule set becoming the inactive rule set.

- Before running the `ipf -s` command, the output from the `ipfstat -I -io` command shows the rules in the inactive rule set. The output from the `ipfstat -io` command shows the rules in the active rule set.

  ```
  # ipfstat -io
  empty list for ipfilter(out)
  block in log quick from 10.0.0.0/8 to any
  block in on dmfe1 proto tcp from 10.1.1.1/32 to any
  # ipfstat -I -io
  pass out quick on dmfe1 all
  pass in quick on dmfe1 all
  block in log quick from 10.0.0.0/8 to any
  ```

- After running the `ipf -s` command, the output from the `ipfstat -I -io` and the `ipfstat -io` command show that the content of the two rules sets have switched.

```
# ipf -s
Set 1 now inactive
# ipfstat -io
pass out quick on dmfe1 all
pass in quick on dmfe1 all
block in log quick from 10.0.0.0/8 to any
# ipfstat -I -io
empty list for inactive ipfilter(out)
block in log quick from 10.0.0.0/8 to any
block in on dmfe1 proto tcp from 10.1.1.1/32 to any
```

## ▼ How to Remove an Inactive Packet Filtering Rule Set From the Kernel

**1 Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**2 Specify the inactive rule set in the "flush all" command.**

```
# ipf -I -Fa
```

This command flushes the inactive rule set from the kernel.

---

**Note –** If you subsequently run `ipf -s`, the empty inactive rule set will become the active rule set. An empty active rule set means that *no* filtering will be done.

---

**Example 26–10** Removing an Inactive Packet Filtering Rule Set From the Kernel

The following example shows how to flush the inactive packet filtering rule set so that all rules have been removed.

```
# ipfstat -I -io
empty list for inactive ipfilter(out)
block in log quick from 10.0.0.0/8 to any
block in on dmfe1 proto tcp from 10.1.1.1/32 to any
# ipf -I -Fa
# ipfstat -I -io
empty list for inactive ipfilter(out)
empty list for inactive ipfilter(in)
```

# Managing NAT Rules for Solaris IP Filter

Use the following procedures to manage, view, and modify NAT rules.

## ▼ How to View Active NAT Rules

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **View the active NAT rules.**

   ```
   # ipnat -l
   ```

**Example 26–11** Viewing Active NAT Rules

The following example shows the output from the active NAT rules set.

```
# ipnat -l
List of active MAP/Redirect filters:
map dmfe0 192.168.1.0/24 -> 20.20.20.1/32

List of active sessions:
```

## ▼ How to Remove NAT Rules

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Remove the current NAT rules.**

   ```
   # ipnat -C
   ```

**Example 26–12** Removing NAT Rules

The following example shows how to remove the entries in the current NAT rules.

```
# ipnat -l
List of active MAP/Redirect filters:
map dmfe0 192.168.1.0/24 -> 20.20.20.1/32
```

```
List of active sessions:
# ipnat -C
1 entries flushed from NAT list
# ipnat -l
List of active MAP/Redirect filters:

List of active sessions:
```

## How to Append Rules to the NAT Rules

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Use one of the following methods to append rules to the active rule set:**

   - Append rules to the NAT rule set at the command line using the `ipnat -f -` command.

     ```
     # echo "map dmfe0 192.168.1.0/24 -> 20.20.20.1/32" | ipnat -f -
     ```

   - Perform the following commands:

     a. Create additional NAT rules in a file of your choice.

     b. Add the rules you have created to the active NAT rules.

        ```
        # ipnat -f filename
        ```

        The rules in *filename* are added to the end of the NAT rules.

### Example 26–13 Appending Rules to the NAT Rule Set

The following example shows how to add a rule to the NAT rule set from the command line.

```
# ipnat -l
List of active MAP/Redirect filters:

List of active sessions:
# echo "map dmfe0 192.168.1.0/24 -> 20.20.20.1/32" | ipnat -f -
# ipnat -l
List of active MAP/Redirect filters:
map dmfe0 192.168.1.0/24 -> 20.20.20.1/32

List of active sessions:
```

# Managing Address Pools for Solaris IP Filter

Use the following procedures to manage, view, and modify address pools.

## ▼ How to View Active Address Pools

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **View the active address pool.**

```
# ippool -l
```

**Example 26–14** Viewing the Active Address Pool

The following example shows how to view the contents of the active address pool.

```
# ippool -l
table role = ipf type = tree number = 13
        { 10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24; };
```

## ▼ How to Remove an Address Pool

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Remove the entries in the current address pool.**

```
# ippool -F
```

**Example 26–15** Removing an Address Pool

The following example shows how to remove an address pool.

```
# ippool -l
table role = ipf type = tree number = 13
        { 10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24; };
# ippool -F
1 object flushed
# ippool -l
```

## How to Append Rules to an Address Pool

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Use one of the following methods to append rules to the active rule set:**

- Append rules to the rule set at the command line using the `ippool -f -` command.

```
# echo "table role = ipf type = tree number = 13
{10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24};" | ippool -f -
```

- Perform the following commands:
  a. Create additional address pools in a file of your choice.
  b. Add the rules you have created to the active address pool.

```
# ippool -f filename
```

The rules in *filename* are added to the end of the active address pool.

**Example 26–16** Appending Rules to an Address Pool

The following example shows how to add an address pool to the address pool rule set from the command line.

```
# ippool -l
table role = ipf type = tree number = 13
        { 10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24; };
# echo "table role = ipf type = tree number = 100
 {10.0.0.0/32, 172.16.1.2/32, 192.168.1.0/24};" | ippool -f -
# ippool -l
table role = ipf type = tree number = 100
        { 10.0.0.0/32, 172.16.1.2/32, 192.168.1.0/24; };
table role = ipf type = tree number = 13
        { 10.1.1.1/32, 10.1.1.2/32, 192.168.1.0/24; };
```

# Displaying Statistics and Information for Solaris IP Filter

**TABLE 26–5** Displaying Solaris IP Filter Statistics and Information (Task Map)

| Task | Description | For Instructions |
|---|---|---|
| View state tables. | View state tables to obtain information about packet filtering using the `ipfstat` command. | "How to View State Tables for Solaris IP Filter" on page 644 |
| View state statistics. | View statistics on packet state information using the `ipfstat -s` command. | "How to View State Statistics for Solaris IP Filter" on page 645 |
| View NAT statistics. | View NAT statistics using the `ipnat -s` command. | "How to View NAT Statistics for Solaris IP Filter" on page 646 |
| View address pool statistics. | View address pool statistics using the `ippool -s` command. | "How to View Address Pool Statistics for Solaris IP Filter" on page 646 |

## ▼ How to View State Tables for Solaris IP Filter

**1 Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**2 View the state table.**

```
# ipfstat
```

---

**Note –** You can use the `-t` option to view the state table in the top utility format.

---

**Example 26–17** Viewing State Tables for Solaris IP Filter

The following example shows how to view a state table.

```
# ipfstat
bad packets:            in 0    out 0
 input packets:         blocked 160 passed 11 nomatch 1 counted 0 short 0
output packets:         blocked 0 passed 13681 nomatch 6844 counted 0 short 0
 input packets logged:  blocked 0 passed 0
output packets logged:  blocked 0 passed 0
 packets logged:        input 0 output 0
 log failures:          input 0 output 0
```

```
fragment state(in):     kept 0  lost 0
fragment state(out):    kept 0  lost 0
packet state(in):       kept 0  lost 0
packet state(out):      kept 0  lost 0
ICMP replies:   0       TCP RSTs sent:  0
Invalid source(in):     0
Result cache hits(in):  152     (out):  6837
IN Pullups succeeded:   0       failed: 0
OUT Pullups succeeded:  0       failed: 0
Fastroute successes:    0       failures:       0
TCP cksum fails(in):    0       (out):  0
IPF Ticks:      14341469
Packet log flags set: (0)
        none
```

## ▼ How to View State Statistics for Solaris IP Filter

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **View the state statistics.**

```
# ipfstat -s
```

**Example 26–18** Viewing State Statistics for Solaris IP Filter

The following example shows how to view state statistics.

```
# ipfstat -s
IP states added:
        0 TCP
        0 UDP
        0 ICMP
        0 hits
        0 misses
        0 maximum
        0 no memory
        0 max bucket
        0 active
        0 expired
        0 closed
State logging enabled

State table bucket statistics:
```

```
0 in use
0.00% bucket usage
0 minimal length
0 maximal length
0.000 average length
```

## How to View NAT Statistics for Solaris IP Filter

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **View NAT statistics.**

```
# ipnat -s
```

**Example 26–19** Viewing NAT Statistics for Solaris IP Filter

The following example shows how to view NAT statistics.

```
# ipnat -s
mapped  in      0       out     0
added   0       expired 0
no memory       0       bad nat 0
inuse   0
rules   1
wilds   0
```

## How to View Address Pool Statistics for Solaris IP Filter

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **View address pool statistics.**

```
# ippool -s
```

**Example 26–20** Viewing Address Pool Statistics for Solaris IP Filter

The following example shows how to view address pool statistics.

```
# ippool -s
Pools:  3
Hash Tables:    0
Nodes:  0
```

# Working With Log Files for Solaris IP Filter

**TABLE 26–6** Working With Solaris IP Filter Log Files (Task Map)

| Task | Description | For Instructions |
|---|---|---|
| Create a log file. | Create a separate Solaris IP filter log file. | "How to Set Up a Log File for Solaris IP Filter" on page 647 |
| View log files. | View state, NAT, and normal log files using the `ipmon` command. | "How to View Solaris IP Filter Log Files" on page 648 |
| Flush the packet log buffer. | Remove the contents of the packet log buffer using the `ipmon -F` command. | "How to Flush the Packet Log File" on page 649 |
| Save logged packets to a file. | Save logged packets to a file for later reference. | "How to Save Logged Packets to a File" on page 650 |

## ▼ How to Set Up a Log File for Solaris IP Filter

By default, all log information for Solaris IP Filter is recorded in the `syslogd` file. You should set up a log file to record Solaris IP Filter traffic information separately from other data that might be logged in the default log file. Perform the following steps.

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Edit the `/etc/syslog.conf` file by adding the following two lines:**

```
# Save IPFilter log output to its own file
local0.debug             /var/log/log-name
```

**Note –** On the second line, make sure to use the Tab key, not the Spacebar, to separate `local0.debug` from `/var/log/`*log-name*.

3 **Create the new log file.**

```
# touch /var/log/log-name
```

4 **Restart the system-log service.**

```
# svcadm restart system-log
```

**Example 26–21** Creating a Solaris IP Filter Log

The following example shows how to create `ipmon.log` to archive IP filter information.

In `/etc/syslog.conf`:

```
# Save IPFilter log output to its own file
local0.debug             /var/log/ipmon.log
```

At the command line:

```
# touch /var/log/ipmon.log
# svcadm restart system-log
```

## ▼ How to View Solaris IP Filter Log Files

**Before You Begin** You should create a separate log file to record Solaris IP Filter data. Refer to "How to Set Up a Log File for Solaris IP Filter" on page 647.

1 **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **View the state, NAT, or normal log files. To view a log file, type the following command, using the appropriate option:**

```
# ipmon -o [S|N|I] filename
```

`S` Displays the state log file.

`N` Displays the NAT log file.

`I` Displays the normal IP log file.

To view all state, NAT, and normal log files, use all the options:

```
# ipmon -o SNI filename
```

- **Provided that you have manually stopped the `ipmon` daemon first, you can also use the following command to display state, NAT, and Solaris IP filter log files:**

  ```
  # ipmon -a filename
  ```

**Note –** Do not use the `ipmon -a` syntax if the `ipmon` daemon is still running. Normally, the daemon is automatically started during system boot. Issuing the `ipmon -a` command also opens another copy of `ipmon`. In such a case, both copies read the same log information, and only one gets a particular log message.

For more information about viewing log files, see the `ipmon`(1M) man page.

**Example 26–22** Viewing Solaris IP Filter Log Files

The following example shows the output from `/var/ipmon.log`.

```
# ipmon -o SNI /var/ipmon.log
02/09/2004 15:27:20.606626 hme0 @0:1 p 129.146.157.149 ->
129.146.157.145 PR icmp len 20 84 icmp echo/0 IN
```

or

```
# pkill ipmon
# ipmon -aD /var/ipmon.log
02/09/2004 15:27:20.606626 hme0 @0:1 p 129.146.157.149 ->
129.146.157.145 PR icmp len 20 84 icmp echo/0 IN
```

## ▼ How to Flush the Packet Log File

1. **Assume a role that includes the IP Filter Management rights profile, or become superuser.**

   You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2. **Flush the pack log buffer.**

   ```
   # ipmon -F
   ```

**Example 26–23** Flushing the Packet Log File

The following example shows the output when a log file is removed. The system provides a report even when there is nothing stored in the log file, as in this example.

```
# ipmon -F
0 bytes flushed from log buffer
0 bytes flushed from log buffer
0 bytes flushed from log buffer
```

## ▼ How to Save Logged Packets to a File

**1 Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**2 Save the logged packets to a file.**

```
# cat /dev/ipl > filename
```

Continue logging packets to the *filename* file until you interrupt the procedure by typing `Control-C` to get the command line prompt back.

**Example 26–24** Saving Logged Packets to a File

The following example shows the result when logged packets are saved to a file.

```
# cat /dev/ipl > /tmp/logfile
^C#

# ipmon -f /tmp/logfile
02/09/2004 15:30:28.708294 hme0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 52 -S IN
02/09/2004 15:30:28.708708 hme0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 40 -A IN
02/09/2004 15:30:28.792611 hme0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 70 -AP IN
02/09/2004 15:30:28.872000 hme0 @0:1 p 129.146.157.149,33923 ->
 129.146.157.145,23 PR tcp len 20 40 -A IN
02/09/2004 15:30:28.872142 hme0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 43 -AP IN
02/09/2004 15:30:28.872808 hme0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 40 -A IN
02/09/2004 15:30:28.872951 hme0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 47 -AP IN
02/09/2004 15:30:28.926792 hme0 @0:1 p 129.146.157.149,33923 ->
  129.146.157.145,23 PR tcp len 20 40 -A IN
.
.
(output truncated)
```

# Creating and Editing Solaris IP Filter Configuration Files

You must directly edit the configuration files to create and modify rule sets and address pools. Configuration files follow standard UNIX syntax rules:

- The pound sign (#) indicates a line containing comments.
- Rules and comments can coexist on the same line.
- Extraneous white space is allowed to keep rules easy to read.
- Rules can be more than one line long. Use the backslash (\) at the end of a line to indicate that the rule continues on the next line.

## ▼ How to Create a Configuration File for Solaris IP Filter

The following procedure describes how to set up the following:

- Packet filtering configuration files
- NAT rules configuration files
- Address pool configuration files

**1 Assume a role that includes the IP Filter Management rights profile, or become superuser.**

You can assign the IP Filter Management rights profile to a role that you create. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**2 Start the file editor of your choice. Create or edit the configuration file for the feature you want to configure.**

- To create a configuration file for packet filtering rules, edit the `ipf.conf` file.

  Solaris IP Filter uses the packet filtering rules that you put in to the `ipf.conf` file. If you locate the rules file for packet filtering in the `/etc/ipf/ipf.conf` file, this file is loaded when the system is booted. If you do not want the filtering rules to be loaded at boot time, put the in a file of your choice. You can then activate the rules with the `ipf` command, as described in "How to Activate a Different or Updated Packet Filtering Rule Set" on page 634.

  See "Using Solaris IP Filter's Packet Filtering Feature" on page 609 for information about creating packet filtering rules.

---

**Note –** If the `ipf.conf` file is empty, there is no filtering. An empty `ipf.conf` file is the same as having a rule set that reads:

```
pass in all
pass out all
```

---

- To create a configuration file for NAT rules, edit the `ipnat.conf` file.

  Solaris IP Filter uses the NAT rules that you put in to the `ipnat.conf` file. If you locate the rules file for NAT in the `/etc/ipf/ipnat.conf` file, this file is loaded when the system is booted. If you do not want the NAT rules loaded at boot time, put the `ipnat.conf` file in a location of your choice. You can then activate the NAT rules with the `ipnat` command.

  See "Using Solaris IP Filter's NAT Feature" on page 612 for information about creating rules for NAT.

- To create a configuration file for address pools, edit the `ippool.conf` file.

  Solaris IP Filter uses the pool of addresses that you put in to the `ippool.conf` file. If you locate the rules file for the pool of addresses in the `/etc/ipf/ippool.conf` file, this file is loaded when the system is booted. If you do not want the pool of addresses loaded at boot time, put the `ippool.conf` file in a location of your choice. You can then activate the pool of addresses with the `ippool` command.

  See "Using Solaris IP Filter's Address Pools Feature" on page 613 for information about creating address pools.

# Solaris IP Filter Configuration File Examples

The following examples provide an illustration of packet filtering rules used in filtering configurations.

**EXAMPLE 26–25** Solaris IP Filter Host Configuration

This example shows a configuration on a host machine with an `elxl` network interface.

```
# pass and log everything by default
pass in log on elxl0 all
pass out log on elxl0 all

# block, but don't log, incoming packets from other reserved addresses
block in quick on elxl0 from 10.0.0.0/8 to any
block in quick on elxl0 from 172.16.0.0/12 to any

# block and log untrusted internal IPs. 0/32 is notation that replaces
# address of the machine running Solaris IP Filter.
block in log quick from 192.168.1.15 to <thishost>
block in log quick from 192.168.1.43 to <thishost>

# block and log X11 (port 6000) and remote procedure call
# and portmapper (port 111) attempts
block in log quick on elxl0 proto tcp from any to elxl0/32 port = 6000 keep state
block in log quick on elxl0 proto tcp/udp from any to elxl0/32 port = 111 keep state
```

**EXAMPLE 26–25** Solaris IP Filter Host Configuration *(Continued)*

This rule set begins with two unrestricted rules that allow everything to pass into and out of the `elxl` interface. The second set of rules blocks any incoming packets from the private address spaces `10.0.0.0` and `172.16.0.0` from entering the firewall. The next set of rules blocks specific internal addresses from the host machine. Finally, the last set of rules blocks packets coming in on port 6000 and port 111.

**EXAMPLE 26–26** Solaris IP Filter Server Configuration

This example shows a configuration for a host machine acting as a web server. This machine has an `eri` network interface.

```
# web server with an eri interface
# block and log everything by default; then allow specific services
# group 100 - inbound rules
# group 200 - outbound rules
# (0/32) resolves to our IP address)
*** FTP proxy ***


# block short packets which are packets fragmented too short to be real.
block in log quick all with short


# block and log inbound and outbound by default, group by destination
block in log on eri0 from any to any head 100
block out log on eri0 from any to any head 200


# web rules that get hit most often
pass in quick on eri0 proto tcp from any \
to eri0/32 port = http flags S keep state group 100
pass in quick on eri0 proto tcp from any \
to eri0/32 port = https flags S keep state group 100


# inbound traffic - ssh, auth
pass in quick on eri0 proto tcp from any \
to eri0/32 port = 22 flags S keep state group 100
pass in log quick on eri0 proto tcp from any \
to eri0/32 port = 113 flags S keep state group 100
pass in log quick on eri0 proto tcp from any port = 113 \
to eri0/32 flags S keep state group 100
```

**EXAMPLE 26–26** Solaris IP Filter Server Configuration *(Continued)*

```
# outbound traffic - DNS, auth, NTP, ssh, WWW, smtp
pass out quick on eri0 proto tcp/udp from eri0/32 \
to any port = domain flags S keep state group 200
pass in quick on eri0 proto udp from any port = domain to eri0/32 group 100

pass out quick on eri0 proto tcp from eri0/32 \
to any port = 113 flags S keep state group 200
pass out quick on eri0 proto tcp from eri0/32 port = 113 \
to any flags S keep state group 200

pass out quick on eri0 proto udp from eri0/32 to any port = ntp group 200
pass in quick on eri0 proto udp from any port = ntp to eri0/32 port = ntp group 100

pass out quick on eri0 proto tcp from eri0/32 \
to any port = ssh flags S keep state group 200

pass out quick on eri0 proto tcp from eri0/32 \
to any port = http flags S keep state group 200
pass out quick on eri0 proto tcp from eri0/32 \
to any port = https flags S keep state group 200

pass out quick on eri0 proto tcp from eri0/32 \
to any port = smtp flags S keep state group 200


# pass icmp packets in and out
pass in quick on eri0 proto icmp from any to eri0/32  keep state group 100
pass out quick on eri0 proto icmp from eri0/32 to any keep state group 200


# block and ignore NETBIOS packets
block in quick on eri0 proto tcp from any \
to any port = 135 flags S keep state group 100

block in quick on eri0 proto tcp from any port = 137 \
to any flags S keep state group 100
block in quick on eri0 proto udp from any to any port = 137 group 100
block in quick on eri0 proto udp from any port = 137 to any group 100

block in quick on eri0 proto tcp from any port = 138 \
to any flags S keep state group 100
block in quick on eri0 proto udp from any port = 138 to any group 100

block in quick on eri0 proto tcp from any port = 139 to any flags S keep state
group 100
```

**EXAMPLE 26–26** Solaris IP Filter Server Configuration *(Continued)*

```
block in quick on eri0 proto udp from any port = 139 to any group 100
```

**EXAMPLE 26–27** Solaris IP Filter Router Configuration

This example shows a configuration for a router that has an internal interface, `ce0`, and an external interface, `ce1`.

```
# internal interface is ce0 at 192.168.1.1
# external interface is ce1 IP obtained via DHCP
# block all packets and allow specific services
*** NAT ***
*** POOLS ***


# Short packets which are fragmented too short to be real.
block in log quick all with short


# By default, block and log everything.
block in log on ce0 all
block in log on ce1 all
block out log on ce0 all
block out log on ce1 all


# Packets going in/out of network interfaces that aren't on the loopback
# interface should not exist.
block in log quick on ce0 from 127.0.0.0/8 to any
block in log quick on ce0 from any to 127.0.0.0/8
block in log quick on ce1 from 127.0.0.0/8 to any
block in log quick on ce1 from any to 127.0.0.0/8


# Deny reserved addresses.
block in quick on ce1 from 10.0.0.0/8 to any
block in quick on ce1 from 172.16.0.0/12 to any
block in log quick on ce1 from 192.168.1.0/24 to any
block in quick on ce1 from 192.168.0.0/16 to any


# Allow internal traffic
pass in quick on ce0 from 192.168.1.0/24 to 192.168.1.0/24
pass out quick on ce0 from 192.168.1.0/24 to 192.168.1.0/24
```

**EXAMPLE 26–27** Solaris IP Filter Router Configuration *(Continued)*

```
# Allow outgoing DNS requests from our servers on .1, .2, and .3
pass out quick on ce1 proto tcp/udp from ce1/32 to any port = domain keep state
pass in quick on ce0 proto tcp/udp from 192.168.1.2 to any port = domain keep state
pass in quick on ce0 proto tcp/udp from 192.168.1.3 to any port = domain keep state


# Allow NTP from any internal hosts to any external NTP server.
pass in quick on ce0 proto udp from 192.168.1.0/24 to any port = 123 keep state
pass out quick on ce1 proto udp from any to any port = 123 keep state


# Allow incoming mail
pass in quick on ce1 proto tcp from any to ce1/32 port = smtp keep state
pass in quick on ce1 proto tcp from any to ce1/32 port = smtp keep state
pass out quick on ce1 proto tcp from 192.168.1.0/24 to any port = smtp keep state


# Allow outgoing connections: SSH, WWW, NNTP, mail, whois
pass in quick on ce0 proto tcp from 192.168.1.0/24 to any port = 22 keep state
pass out quick on ce1 proto tcp from 192.168.1.0/24 to any port = 22 keep state

pass in quick on ce0 proto tcp from 192.168.1.0/24 to any port = 80 keep state
pass out quick on ce1 proto tcp from 192.168.1.0/24 to any port = 80 keep state
pass in quick on ce0 proto tcp from 192.168.1.0/24 to any port = 443 keep state
pass out quick on ce1 proto tcp from 192.168.1.0/24 to any port = 443 keep state

pass in quick on ce0 proto tcp from 192.168.1.0/24 to any port = nntp keep state
block in quick on ce1 proto tcp from any to any port = nntp keep state
pass out quick on ce1 proto tcp from 192.168.1.0/24 to any port = nntp keep state

pass in quick on ce0 proto tcp from 192.168.1.0/24 to any port = smtp keep state

pass in quick on ce0 proto tcp from 192.168.1.0/24 to any port = whois keep state
pass out quick on ce1 proto tcp from any to any port = whois keep state


# Allow ssh from offsite
pass in quick on ce1 proto tcp from any to ce1/32 port = 22 keep state


# Allow ping out
pass in quick on ce0 proto icmp all keep state
pass out quick on ce1 proto icmp all keep state
```

**EXAMPLE 26–27** Solaris IP Filter Router Configuration *(Continued)*

```
# allow auth out
pass out quick on ce1 proto tcp from ce1/32 to any port = 113 keep state
pass out quick on ce1 proto tcp from ce1/32 port = 113 to any keep state


# return rst for incoming auth
block return-rst in quick on ce1 proto tcp from any to any port = 113 flags S/SA


# log and return reset for any TCP packets with S/SA
block return-rst in log on ce1 proto tcp from any to any flags S/SA


# return ICMP error packets for invalid UDP packets
block return-icmp(net-unr) in proto udp all
```

PART V

# Mobile IP

This part introduces Mobile Internet Protocol (Mobile IP) and contains tasks for Mobile IP administration. You can install Mobile IP on systems such as laptops and wireless communications, enabling these computers to operate on foreign networks.

---

**Note –** The Mobile IP feature has been removed from all Solaris updates after Solaris 10 8/07.

---

◆ ◆ ◆ CHAPTER 27

# Mobile IP (Overview)

Mobile Internet Protocol (IP) enables the transfer of information between mobile computers. *Mobile computers* include lap tops and wireless communications. The mobile computer can change its location to a foreign network. At the foreign network, the mobile computer can still communicate through the home network of the mobile computer. The Solaris implementation of Mobile IP supports only IPv4.

This chapter contains the following information:


- "Introduction to Mobile IP" on page 662
- "Mobile IP Functional Entities" on page 663
- "How Mobile IP Works" on page 664
- "Agent Discovery" on page 666
- "Care-of Addresses" on page 667
- "Mobile IP With Reverse Tunneling" on page 668
- "Mobile IP Registration" on page 670
- "Routing Datagrams to and From Mobile Nodes" on page 674
- "Security Considerations for Mobile IP" on page 676


For Mobile IP-related tasks, refer to Chapter 28, "Administering Mobile IP (Tasks)." For Mobile IP reference materials, refer to Chapter 29, "Mobile IP Files and Commands (Reference)."

## What's New in Mobile IP

The Mobile IP feature is removed from Solaris 10 updates after Solaris 10 8/07.

# Introduction to Mobile IP

Current versions of the Internet Protocol (IP) assume that the point at which a computer attaches to the Internet or a network is fixed. IP also assumes that the IP address of the computer identifies the network to which the computer is attached. Datagrams that are sent to a computer are based on the location information that is contained in the IP address. Many Internet Protocols require that a node's IP address remain unchanged. If any of these protocols are active on a Mobile IP computing device, their applications fail. Even HTTP would fail if not for the short-lived nature of its TCP connections. Updating an IP address and refreshing the web page is not a burden.

If a mobile computer, or *mobile node*, moves to a new network while its IP address is unchanged, the mobile node address does not reflect the new point of attachment. Consequently, routing protocols that exist cannot route datagrams to the mobile node correctly. You must reconfigure the mobile node with a different IP address that represents the new location. Assigning a different IP address is cumbersome. Thus, under the current Internet Protocol, if the mobile node moves without changing its address, it loses routing. If the mobile node does change its address, it loses connections.

Mobile IP solves this problem by allowing the mobile node to use two IP addresses. The first address is a fixed *home address*. The second address is a *care-of address* that changes at each new point of attachment. Mobile IP enables a computer to roam freely on the Internet. Mobile IP also enables a computer to roam freely on an organization's network while still maintaining the same home address. Consequently, communication activities are not disrupted when the user changes the computer's point of attachment. Instead, the network is updated with the new location of the mobile node. See the Glossary for definitions of terms that are associated with Mobile IP.

The following figure illustrates the general Mobile IP topology.

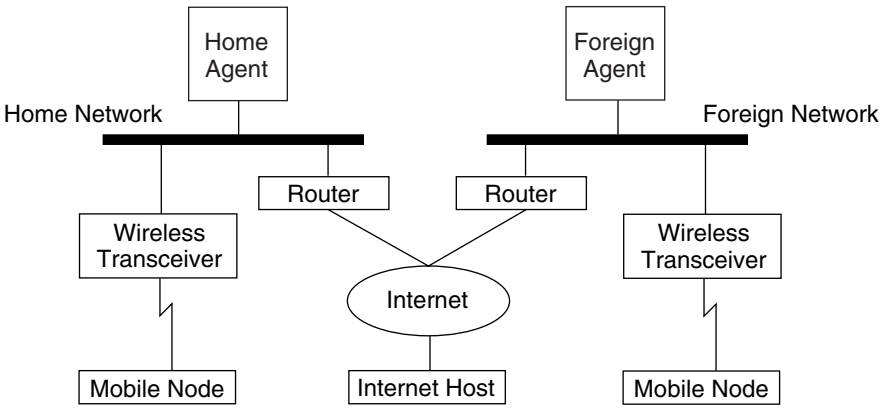



**FIGURE 27–1** Mobile IP Topology

By using this figure's Mobile IP topology, the following scenario shows how a datagram moves from one point to another point within the Mobile IP framework:

1. The Internet host sends a datagram to the mobile node by using the mobile node's home address (normal IP routing process).
2. If the mobile node is on its home network, the datagram is delivered through the normal IP process to the mobile node. Otherwise, the home agent receives the datagram.
3. If the mobile node is on a foreign network, the home agent forwards the datagram to the foreign agent. The home agent must encapsulate the datagram in an outer datagram so that the foreign agent's IP address appears in the outer IP header.
4. The foreign agent delivers the datagram to the mobile node.
5. Datagrams from the mobile node to the Internet host are sent by using normal IP routing procedures. If the mobile node is on a foreign network, the packets are delivered to the foreign agent. The foreign agent forwards the datagram to the Internet host.
6. In situations with ingress filtering present, the source address must be topologically correct for the subnet that the datagram is coming from, or a router cannot forward the datagram. If this scenario exists on links between the mobile node and the correspondent node, the foreign agent needs to provide reverse tunneling support. Then, the foreign agent can deliver every datagram that the mobile node sends to its home agent. The home agent then forwards the datagram through the path that the datagram would have taken had the mobile node resided on the home network. This process guarantees that the source address is correct for all links that the datagram must traverse.

Regarding wireless communications, Figure 27–1 depicts the use of wireless transceivers to transmit the datagrams to the mobile node. Also, all datagrams between the Internet host and the mobile node use the home address of the mobile node. The home address is used even when the mobile node is located on the foreign network. The care-of address is used only for communication with mobility agents. The care-of address is invisible to the Internet host.

# Mobile IP Functional Entities

Mobile IP introduces the following new functional entities:

- **Mobile node (MN)** – Host or router that changes its point of attachment from one network to another network while maintaining all existing communications by using its IP home address.
- **Home agent (HA)** – Router or server on the home network of a mobile node. The router intercepts datagrams that are destined for the mobile node. The router then delivers the datagrams through the care-of address. The home agent also maintains current information on the location of the mobile node.
- **Foreign agent (FA)** – Router or server on the foreign network that the mobile node visits. Provides host routing services to the mobile node. The foreign agent might also provide a care-of address to the mobile node while the mobile node is registered.

# How Mobile IP Works

Mobile IP enables routing of IP datagrams to mobile nodes. The home address of the mobile node always identifies the mobile node regardless of where the mobile node is attached. When away from home, a care-of address is associated with the mobile node's home address. The care-of address provides information about the current point of attachment of the mobile node. Mobile IP uses a registration mechanism to register the care-of address with a home agent.

The home agent redirects datagrams from the home network to the care-of address. The home agent constructs a new IP header that contains the care-of address of the mobile node as the destination IP address. This new header encapsulates the original IP datagram. Consequently, the home address of the mobile node has no effect on the routing of the encapsulated datagram until the datagram arrives at the care-of address. This type of encapsulation is called *tunneling*. After the datagram arrives at the care-of address, the datagram is de-encapsulated. Then the datagram is delivered to the mobile node.

The following figure shows a mobile node that resides on its home network, Network A, before the mobile node moves to a foreign network, Network B. Both networks support Mobile IP. The mobile node is always associated with the home address of the mobile node, `128.226.3.30`.

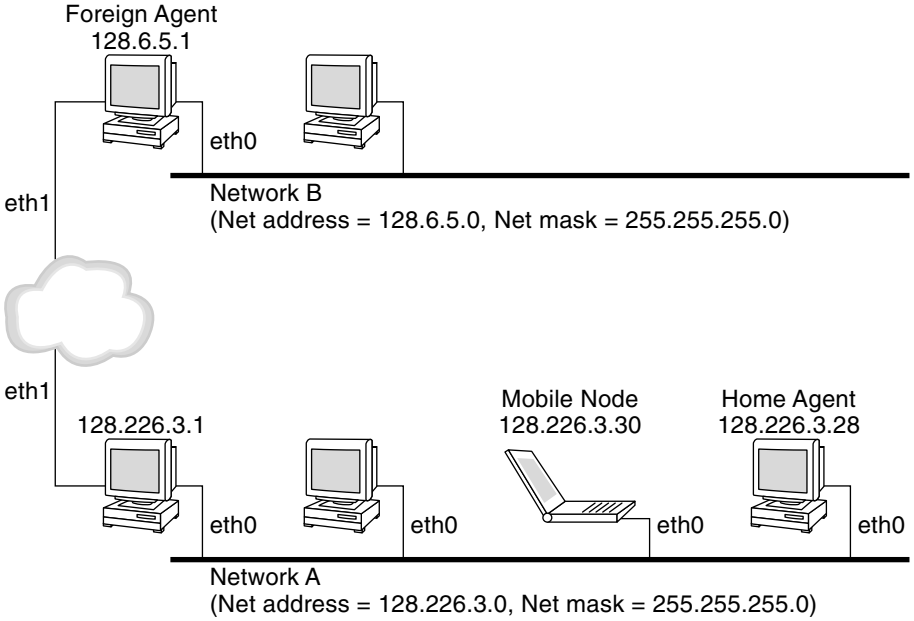



**FIGURE 27–2** Mobile Node Residing on Home Network

The following figure shows a mobile node that has moved to a foreign network, Network B. Datagrams that are destined for the mobile node are intercepted by the home agent on the home network, Network A. The datagrams are encapsulated. Then, the datagrams are sent to

the foreign agent on Network B. The foreign agent strips off the outer header. Then the foreign agent delivers the datagram to the mobile node that is located on Network B.

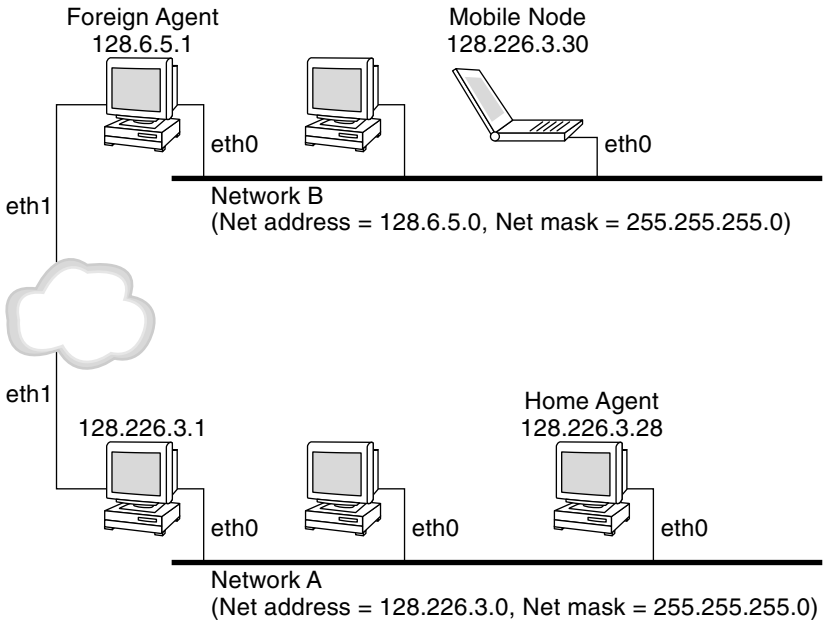



**FIGURE 27–3** Mobile Node Moving to a Foreign Network

The care-of address might belong to a foreign agent. The care-of address might be acquired by the mobile node through the Dynamic Host Configuration Protocol (DHCP) or the Point-to-Point Protocol (PPP). In the latter situation, a mobile node has a colocated care-of address.

Mobility agents (home agents and foreign agents) advertise their presence by using *agent advertisement* messages. Optionally, a mobile node can solicit an agent advertisement message. The mobile node uses any mobility agent that is attached locally through an *agent solicitation* message. A mobile node uses the agent advertisements to determine whether the mobile node is on the home network or a foreign network.

The mobile node uses a special registration process to inform the home agent about the current location of the mobile node. The mobile node is always "listening" for mobility agents advertising their presence. The mobile node uses these advertisements to help determine when the mobile node moves to another subnet. When a mobile node determines that the mobile node has moved its location, the mobile node uses the new foreign agent to forward a registration message to the home agent. The mobile node uses the same process when the mobile node moves from one foreign network to another foreign network.

When the mobile node detects that it is located on the home network, the mobile node does not use mobility services. When the mobile node returns to the home network, the mobile node *deregisters* with the home agent.

# Agent Discovery

A mobile node uses a method that is known as *agent discovery* to determine the following information:

- When the node has moved from one network to another network
- Whether the network is the home network or a foreign network
- The foreign agent care-of address that is offered by each foreign agent on that network
- Mobility services that are provided by the mobility agent, advertised as flags, and additional extensions in the agent advertisement

Mobility agents transmit *agent advertisements* to advertise services on a network. In the absence of agent advertisements, a mobile node can solicit advertisements. This capability is known as *agent solicitation*. If a mobile node is capable of supporting its own colocated care-of address, the mobile node can use regular router advertisements for the same purposes.

## Agent Advertisement

Mobile nodes use agent advertisements to determine the current point of attachment to the Internet or to an organization's network. An agent advertisement is an Internet Control Message Protocol (ICMP) router advertisement that has been extended to also carry a mobility agent advertisement extension.

A foreign agent (FA) can be too busy to serve additional mobile nodes. However, a foreign agent must continue to send agent advertisements. Then, the mobile node, which is already registered with a foreign agent, knows that the mobile node has not moved out of range of the foreign agent. The mobile node also knows that the foreign agent has not failed. A mobile node that is registered with a foreign agent from which it no longer receives agent advertisements probably knows that the mobile node can no longer contact that foreign agent.

### Agent Advertisement Over Dynamic Interfaces

You can configure the implementation of the foreign agent to send advertisements over dynamically created interfaces. You have options to enable or disable limited unsolicited advertisements over the advertising interfaces. Dynamically created interfaces are defined as only those interfaces that are configured after the `mipagent` daemon starts. Advertisement over dynamic interfaces is useful for applications that support transient mobility interfaces. Moreover, by limiting unsolicited advertisement, network bandwidth might be saved.

## Agent Solicitation

Every mobile node should implement agent solicitation. The mobile node uses the same procedures, defaults, and constants for agent solicitation that are specified for solicitation messages of ICMP routers.

The rate that a mobile node sends solicitations is limited by the mobile node. The mobile node can send three initial solicitations at a maximum rate of one solicitation per second while the mobile node searches for an agent. After the mobile node registers with an agent, the rate that solicitations are sent is reduced to limit the overhead on the local network.

# Care-of Addresses

Mobile IP provides the following alternative modes for the acquisition of a care-of address:

- A foreign agent provides a *foreign agent care-of address*, which is advertised to the mobile node through agent advertisement messages. The care-of address is usually the IP address of the foreign agent that sends the advertisements. The foreign agent is the endpoint of the tunnel. When the foreign agent receives datagrams through a tunnel, the foreign agent de-encapsulates the datagrams. Then, the foreign agent delivers the inner datagram to the mobile node. Consequently, many mobile nodes can share the same care-of address. Bandwidth is important on wireless links. Wireless links are good candidates from which foreign agents can provide Mobile IP services to higher bandwidth-wired links.
- A mobile node acquires a *colocated care-of address* as a local IP address through some external means. The mobile node then associates with one of its own network interfaces. The mobile node might acquire the address through DHCP as a temporary address. The address might also be owned by the mobile node as a long-term address. However, the mobile node can only use the address while visiting the subnet to which this care-of address belongs. When using a colocated care-of address, the mobile node serves as the endpoint of the tunnel. The mobile node performs de-encapsulation of the datagrams that are tunneled to the mobile node.

A colocated care-of address enables a mobile node to function without a foreign agent. Consequently, a mobile node can use a colocated care-of address in networks that have not deployed a foreign agent.

If a mobile node is using a colocated care-of address, the mobile node must be located on the link that is identified by the network prefix of the care-of address. Otherwise, datagrams that are destined to the care-of address cannot be delivered.

# Mobile IP With Reverse Tunneling

The section "How Mobile IP Works" on page 664 assumes that the routing within the Internet is independent of the source address of the datagram. However, intermediate routers might check for a topologically correct source address. If an intermediate router does check, the mobile node needs to set up a reverse tunnel. By setting up a reverse tunnel from the care-of address to the home agent, you ensure a topologically correct source address for the IP data packet. Reverse tunnel support is advertised by foreign agents and home agents. A mobile node can request a reverse tunnel between the foreign agent and the home agent when the mobile node registers. A reverse tunnel is a tunnel that starts at the care-of address of the mobile node and terminates at the home agent. The following figure shows the Mobile IP topology that uses a reverse tunnel.

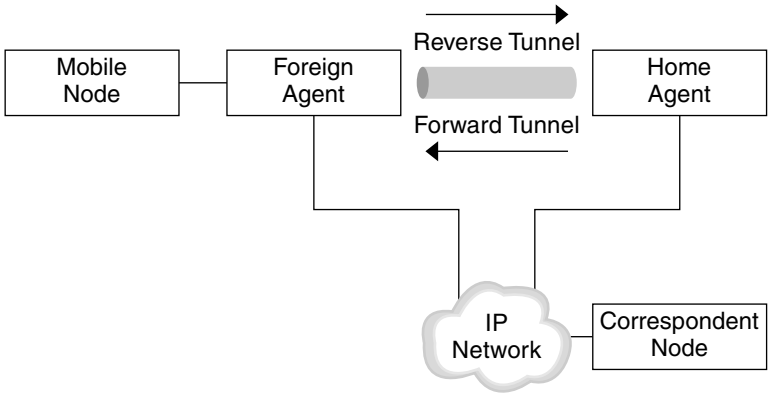



**FIGURE 27–4** Mobile IP With a Reverse Tunnel

## Limited Private Addresses Support

Mobile nodes that have private addresses that are not globally routeable through the Internet require reverse tunnels. Solaris Mobile IP supports mobile nodes that are privately addressed. See "Overview of the Solaris Mobile IP Implementation" on page 695 for the functions that Solaris Mobile IP does not support.

Enterprises employ private addresses when external connectivity is not required. Private addresses are not routeable through the Internet. When a mobile node has a private address, the mobile node can only communicate with a correspondent node by having its datagrams reverse-tunneled to its home agent. The home agent then delivers the datagram to the correspondent node in whatever manner the datagram is normally delivered when the mobile node is at home. The following figure shows a network topology with two mobile nodes that are privately addressed. The two mobile nodes use the same care-of address when they are registered to the same foreign agent.

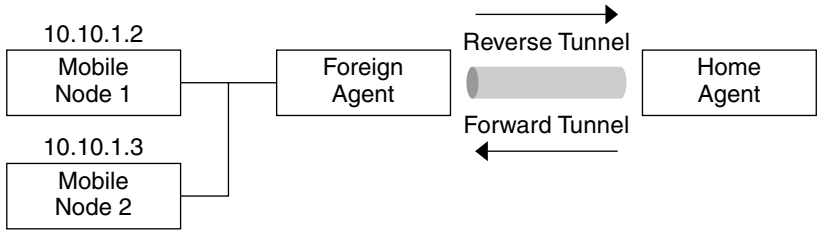



**FIGURE 27–5** Privately Addressed Mobile Nodes Residing on the Same Foreign Network

The care-of address and the home agent address must be globally routeable addresses if these addresses belong to different domains that are connected by a public Internet.

The same foreign network can include two mobile nodes that are privately addressed with the same IP address. However, each mobile node must have a different home agent. Also, each mobile node must be on different advertising subnets of a single foreign agent. The following figure shows a network topology that depicts this situation.

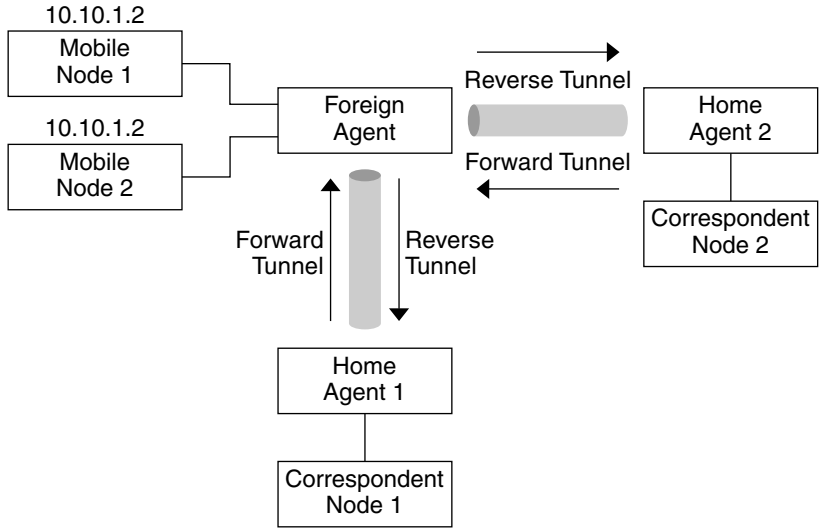



**FIGURE 27–6** Privately Addressed Mobile Nodes Residing on Different Foreign Networks

# Mobile IP Registration

Mobile nodes detect when they have moved from one subnet to another subnet through the use of agent advertisements. When the mobile node receives an agent advertisement that indicates that the mobile node has changed locations, the mobile node registers through a foreign agent. Even though the mobile node might have acquired its own colocated care-of address, this feature is provided to enable sites to restrict access to mobility services.

Mobile IP registration provides a flexible mechanism for mobile nodes to communicate the current reachability information to the home agent. The registration process enables mobile nodes to perform the following tasks:

- Request forwarding services when visiting a foreign network
- Inform the home agent of the current care-of address
- Renew a registration that is about to expire
- Deregister when the mobile node returns home
- Request a reverse tunnel

Registration messages exchange information between a mobile node, a foreign agent, and the home agent. Registration creates or modifies a mobility binding at the home agent. Registration associates the home address of the mobile node with the care-of address of the mobile node for the specified lifetime.

The registration process also enables mobile nodes to do the following functions:

- Register with multiple foreign agents
- Deregister specific care-of addresses while retaining other mobility bindings
- Discover the address of a home agent if the mobile node is not configured with this information

Mobile IP defines the following registration processes for a mobile node:

- If a mobile node registers a foreign agent care-of address, the mobile node is informing the home agent that it is reachable through that foreign agent.
- If a mobile node receives an agent advertisement that requires the mobile node to register through a foreign agent, the mobile node can still attempt to obtain a colocated care-of address. The mobile node can also register with that foreign agent or any other foreign agent on that link.
- If a mobile node uses a colocated care-of address, the mobile node registers directly with the home agent.
- If a mobile node returns to the home network, the mobile node deregisters with the home agent.

These registration processes involve the exchange of registration requests and registration reply messages. When the mobile node registers by using a foreign agent, the registration process takes the following steps, which the subsequent figure shows:

1. The mobile node sends a registration request to the prospective foreign agent to begin the registration process.
2. The foreign agent processes the registration request and then relays the request to the home agent.
3. The home agent sends a registration reply to the foreign agent to grant or deny the request.
4. The foreign agent processes the registration reply and then relays the reply to the mobile node to inform the mobile node of the disposition of the request.

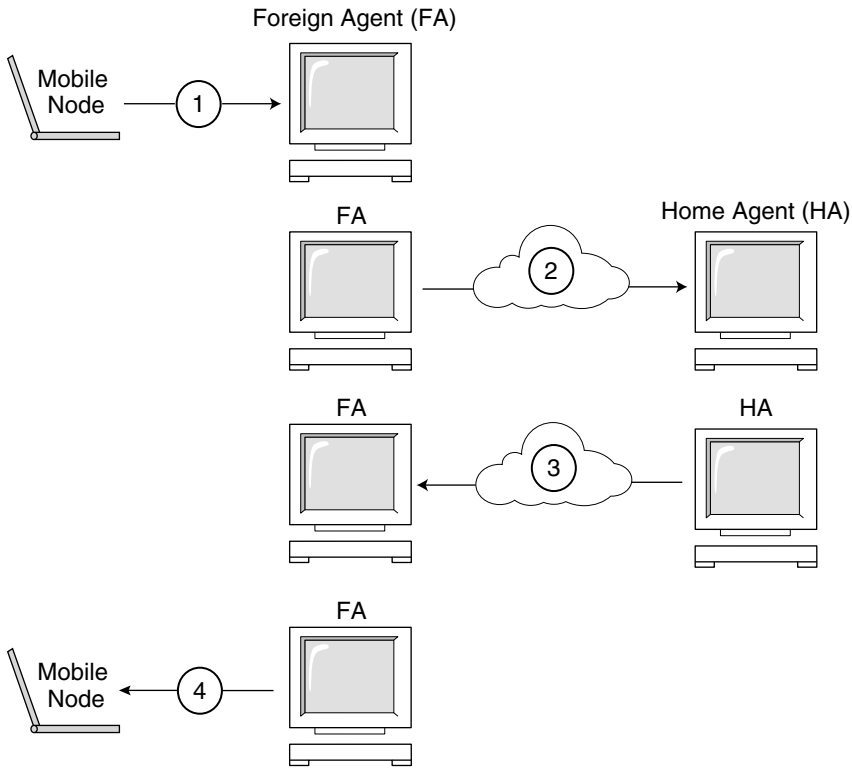



**FIGURE 27–7** Mobile IP Registration Process

When the mobile node registers directly with the home agent, the registration process requires only the following steps:

- The mobile node sends a registration request to the home agent.
- The home agent sends a registration reply to the mobile node that grants or denies the request.

Also, either the foreign agent or the home agent might require a reverse tunnel. If the foreign agent supports reverse tunneling, the mobile node uses the registration process to request a reverse tunnel. The mobile node sets the reverse tunnel flag in the registration request to request a reverse tunnel.

## Network Access Identifier (NAI)

Authentication, authorization, and accounting (AAA) servers, in use within the Internet, provide authentication and authorization services for dialup computers. These services are likely to be equally valuable for mobile nodes that use Mobile IP when the nodes attempt to connect to foreign domains with AAA servers. AAA servers use the Network Access Identifier (NAI) to identify clients. A mobile node can identify itself by including the NAI in the Mobile IP registration request.

Because the NAI is typically used to uniquely identify the mobile node, the home address of the mobile node is not always necessary to provide that function. Thus, a mobile node can authenticate itself. Consequently, a mobile node can be authorized for connection to the foreign domain without even having a home address. To request that a home address be assigned, a message that contains the mobile node NAI extension can set the home address field to zero in the registration request.

## Mobile IP Message Authentication

Each mobile node, foreign agent, and home agent supports a mobility security association between the various Mobile IP components. The security association is indexed by the security parameter index (SPI) and IP address. In the instance of the mobile node, this address is the home address of the mobile node. Registration messages between a mobile node and the home agent are authenticated with the mobile-home authentication extension. In addition to mobile-home authentication, which is mandatory, you can use the optional mobile-foreign agent and home-foreign agent authentications.

## Mobile Node Registration Request

A mobile node uses a *registration request* message to register with the home agent. Thus, the home agent can create or modify a mobility binding for that mobile node (for example, with a new lifetime). The foreign agent can relay the registration request to the home agent. However, if the mobile node is registering a colocated care-of address, then the mobile node can send the registration request directly to the home agent. If the foreign agent advertises that registration messages must be sent to the foreign agent, then the mobile node must send the registration request to the foreign agent.

## Registration Reply Message

A mobility agent returns a *registration reply* message to a mobile node that has sent a registration request message. If the mobile node requests service from a foreign agent, that foreign agent receives the reply from the home agent. Subsequently, the foreign agent relays the reply to the mobile node. The reply message contains the necessary codes to inform the mobile node and the foreign agent about the status of the registration request. The message also contains the lifetime that is granted by the home agent. The lifetime can be smaller than the original request. The registration reply can also contain a dynamic home address assignment.

## Foreign Agent Considerations

The foreign agent plays a mostly passive role in Mobile IP registration. The foreign agent adds all mobile nodes that are registered to the visitor table. The foreign agent relays registration requests between mobile nodes and home agents. Also, when the foreign agent provides the care-of address, the foreign agent de-encapsulates datagrams for delivery to the mobile node. The foreign agent also sends periodic agent advertisement messages to advertise the presence of the foreign agent.

If home agents and foreign agents support reverse tunnels, and the mobile node requests a reverse tunnel, the foreign agent then tunnels all the packets from the mobile node to the home agent. The home agent then sends the packets to the correspondent node. This process is the reverse of the home agent tunneling all of the mobile node's packets to the foreign agent for delivery to the mobile node. A foreign agent that supports reverse tunnels advertises that the reverse tunnel is supported for registration. Because of the local policy, the foreign agent can deny a registration request when the reverse tunnel flag is not set. The foreign agent can only distinguish multiple mobile nodes with the same (private) IP address when these mobile nodes are visiting different interfaces on the foreign agent. In the forward tunnel situation, the foreign agent distinguishes between multiple mobile nodes that share the same private addresses by looking at the incoming tunnel interface. The incoming tunnel interface maps to a unique home agent address.

## Home Agent Considerations

Home agents play an active role in the registration process. The home agent receives registration requests from the mobile node. The registration request might be relayed by the foreign agent. The home agent updates its record of the mobility bindings for this mobile node. The home agent issues a suitable registration reply in response to each registration request. The home agent also forwards packets to the mobile node when the mobile node is away from the home network.

A home agent might not have to have a physical subnet configured for mobile nodes. However, the home agent must recognize the home address of the mobile node through the

`mipagent.conf` file or some other mechanism when the home agent grants registration. For more information about `mipagent.conf`, refer to "Creating the Mobile IP Configuration File" on page 680.

A home agent can support private addressed mobile nodes by configuring the private addressed mobile nodes in the `mipagent.conf` file. The home addresses that are used by the home agent must be unique.

## Dynamic Home Agent Discovery

In some situations, the mobile node might not know the home agent address when the mobile node attempts to register. If the mobile node does not know the home agent address, the mobile node can use dynamic home agent address resolution to learn the address. In this situation, the mobile node sets the home agent field of the registration request to the subnet-directed broadcast address of its home network. Each home agent that receives a registration request with a broadcast destination address rejects the mobile node's registration by returning a rejection registration reply. By doing so, the mobile node can use the home agent's unicast IP address that is indicated in the rejection reply when the mobile node next attempts registration.

# Routing Datagrams to and From Mobile Nodes

This section describes how mobile nodes, home agents, and foreign agents cooperate to route datagrams for mobile nodes that are connected to a foreign network. See "Overview of the Solaris Mobile IP Implementation" on page 695 for Mobile IP functions that are supported in the Solaris OS.

## Encapsulation Methods

Home agents and foreign agents use one of the available encapsulation methods to support datagrams that use a tunnel. Defined encapsulation methods are IP-in-IP Encapsulation, Minimal Encapsulation, and Generic Routing Encapsulation. Foreign agent and home agent cases, or indirect colocated mobile node and home agent cases, must support the same encapsulation method. All Mobile IP entities are required to support IP-in-IP Encapsulation.

## Unicast Datagram Routing

When registered on a foreign network, the mobile node uses the following rules to choose a default router:

- If the mobile node is registered and uses a foreign agent care-of address, the process is straightforward. The mobile node chooses its default router from among the router addresses that are advertised in the ICMP router advertisement portion of that agent advertisement. The mobile node can also consider the IP source address of the agent advertisement as another possible choice for the IP address of a default router.
- The mobile node might be registered directly with the home agent by using a colocated care-of address. Then, the mobile node chooses its default router from among those routers that are advertised in any ICMP router advertisement message that it receives. The network prefix of the chosen default router must match the network prefix of the care-of address of the mobile node that is externally obtained. The address might match the IP source address of the agent advertisement under the network prefix. Then, the mobile node can also consider that IP source address as another possible choice for the IP address of a default router.
- If the mobile node is registered, a foreign agent that supports reverse tunnels routes unicast datagrams from the mobile node to the home agent through the reverse tunnel. If the mobile node is registered with a foreign agent that provides reverse tunnel support, the mobile node must use that foreign agent as its default router.

## Broadcast Datagrams

When a home agent receives a broadcast datagram or multicast datagram, the home agent only forwards the datagram to mobile nodes that have specifically requested that they receive them. How the home agent forwards broadcast and multicast datagrams to mobile nodes depends primarily on two factors. Either that mobile node is using a foreign-agent provided care-of address, or the mobile node is using its own colocated care-of address. The former means that the datagram must be double encapsulated. The first IP header identifies the mobile node for which the datagram is to be delivered. This first IP header is not present in the broadcast or multicast datagram. The second IP header identifies the care-of address, and is the usual tunnel header. In the latter instance, the mobile node is decapsulating its own datagrams, and the datagram needs only to be sent through the regular tunnel.

## Multicast Datagram Routing

To begin receiving multicast traffic when a mobile node is visiting a foreign subnet, a mobile node can join a multicast group in any of the following ways:

- If the mobile node is using a colocated care-of address, the mobile node can use this address as the source IP address of any Internet Group Management Protocol (IGMP) join messages. However, a multicast router must be present on the visited subnet.
- If the mobile node wants to join the ICMP group from its home subnet, the mobile node must use a reverse tunnel to send IGMP join messages to the home agent. However, the mobile node's home agent must be a multicast router. The home agent then forwards multicast datagrams through the tunnel to the mobile node.

- If the mobile node is using a colocated care-of address, the mobile node can use this address as the source IP address of any IGMP join messages. However, a multicast router must be present on the visited subnet. After the mobile node has joined the group, the mobile node can participate by sending its own multicast packets directly on the visited network.
- Send directly on the visited network.
- Send through a tunnel to the home agent.

Multicast routing depends on the IP source address. A mobile node that is sending a multicast datagram must send the datagram from a valid source address on that link. So a mobile node that is sending multicast datagrams directly on the visited network must use a colocated care-of address as the IP source address. Also, the mobile node must have joined the multicast group that is associated with the address. Similarly, a mobile node that joined a multicast group while on its home subnet before roaming, or joined the multicast group while roaming through a reverse tunnel to its home agent, must use its home address as the IP source address of the multicast datagram. Thus, the mobile node must have these datagrams reverse-tunneled to its home subnet as well, either through itself by using its colocated care-of address, or through a foreign agent reverse tunnel.

While it seems more efficient for a mobile node to always join from the subnet that the mobile node is visiting, it is still a mobile node. Consequently, the mobile node would have to repeat the join every time the mobile node switches subnets. The more efficient way is for the mobile node to join through its home agent, and not have to carry this overhead. Also, multicast sessions might be present that are only available from the home subnet. Other considerations might also force the mobile node to participate in a specific way.

# Security Considerations for Mobile IP

In many situations, mobile computers use wireless links to connect to the network. Wireless links are particularly vulnerable to passive eavesdropping, active replay attacks, and other active attacks.

Because Mobile IP recognizes its inability to reduce or eliminate this vulnerability, Mobile IP uses a form of authentication to protect Mobile IP registration messages from these types of attack. The default algorithm that is used is MD5, with a key size of 128 bits. The default operational mode requires that this 128-bit key precede and succeed the data to be hashed. The foreign agent uses MD5 to support authentication. The foreign agent also uses key sizes of 128 bits or greater, with manual key distribution. Mobile IP can support more authentication algorithms, algorithm modes, key distribution methods, and key sizes.

These methods do prevent Mobile IP registration messages from being altered. However, Mobile IP also uses a form of replay protection to alert Mobile IP entities when they receive duplicates of previous Mobile IP registration messages. If this protection method were not used, the mobile node and its home agent might become unsynchronized when either of them

receives a registration message. Hence, Mobile IP updates its state. For example, a home agent receives a duplicate deregistration message while the mobile node is registered through a foreign agent.

Replay protection is ensured either by a method known as *nonces,* or *timestamps*. Nonces and timestamps are exchanged by home agents and mobile nodes within the Mobile IP registration messages. Nonces and timestamps are protected from change by an authentication mechanism. Consequently, if a home agent or mobile node receives a duplicate message, the duplicate message can be thrown away.

The use of tunnels can be a significant vulnerability, especially if registration is not authenticated. Also, the Address Resolution Protocol (ARP) is not authenticated, and can potentially be used to steal another host's traffic.

## Use of IPsec With Mobile IP

In general, because home agents and foreign agents are fixed entities, they can use IPsec authentication or encryption to protect both Mobile IP registration messages and forward and reverse tunnel traffic. This process works completely independently of Mobile IP, and only depends on the workstation's ability to perform IPsec functions. Mobile nodes can also use IPsec authentication to protect their registration traffic. If the mobile node registers through a foreign agent, in general the mobile node cannot use IPsec encryption. The reason that the mobile node cannot use IPsec encryption is because the foreign agent must be able to check the information in the registration packet. While IPsec encryption could be used when a foreign agent is not needed, the issue of colocation makes this difficult to achieve. IPsec is an IP-level security relationship. Consequently, a home agent would have to know the mobile node's colocated address without prior information or registration messages. For more information about IPsec, see Chapter 19, “IP Security Architecture (Overview),” or Chapter 20, “Configuring IPsec (Tasks).”

◆ ◆ ◆ CHAPTER 28

# Administering Mobile IP (Tasks)

This chapter provides procedures for modifying, adding, deleting, and displaying parameters in the Mobile IP configuration file. This chapter also shows you how to display mobility agent status.

This chapter contains the following information:


- "Creating the Mobile IP Configuration File (Task Map)" on page 679
- "Creating the Mobile IP Configuration File" on page 680
- "Modifying the Mobile IP Configuration File" on page 685
- "Modifying the Mobile IP Configuration File (Task Map)" on page 684
- "Displaying Mobility Agent Status" on page 692
- "Displaying Mobility Routes on a Foreign Agent" on page 694


For an introduction to Mobile IP, refer to Chapter 27, "Mobile IP (Overview)." For detailed information about Mobile IP, refer to Chapter 29, "Mobile IP Files and Commands (Reference)."

**Note –** The Mobile IP feature is removed from Solaris 10 updates after Solaris 10 8/07.

## Creating the Mobile IP Configuration File (Task Map)

| Task | Description | For Instructions |
|---|---|---|
| Create the Mobile IP configuration file. | Involves creating the `/etc/inet/mipagent.conf` file or copying one of the sample files. | "How to Create the Mobile IP Configuration File" on page 681 |

| Task | Description | For Instructions |
|---|---|---|
| Configure the `General` section. | Involves typing the version number into the `General` section of the Mobile IP configuration file. | "How to Configure the `General` Section" on page 681 |
| Configure the `Advertisements` section. | Involves adding labels and values, or changing them, in the `Advertisements` section of the Mobile IP configuration file. | "How to Configure the `Advertisements` Section" on page 682 |
| Configure the `GlobalSecurityParameters` section. | Involves adding labels and values, or changing them, in the `GlobalSecurityParameters` section of the Mobile IP configuration file. | "How to Configure the `GlobalSecurityParameters` Section" on page 682 |
| Configure the `Pool` section. | Involves adding labels and values, or changing them, in the `Pool` section of the Mobile IP configuration file. | "How to Configure the `Pool` Section" on page 683 |
| Configure the `SPI` section. | Involves adding labels and values, or changing them, in the `SPI` section of the Mobile IP configuration file. | "How to Configure the `SPI` Section" on page 683 |
| Configure the `Address` section. | Involves adding labels and values, or changing them, in the `Address` section of the Mobile IP configuration file. | "How to Configure the `Address` Section" on page 683 |

# Creating the Mobile IP Configuration File

This section explains how to plan for Mobile IP and create the `/etc/inet/mipagent.conf`file.

## ▼ How to Plan for Mobile IP

When you configure the `mipagent.conf` file for the first time, you need to perform the following tasks:

1 **Depending on your organization's requirements for its hosts, determine what functionality your Mobile IP agent can provide:**

- Foreign agent functionality only
- Home agent functionality only
- Both foreign agent and home agent functionality

2 **Create the `/etc/inet/mipagent.conf` file and specify the settings you require by using the procedures that are described in this section. You can also copy one of the following files to `/etc/inet/mipagent.conf` and modify it according to your requirements:**

- For foreign agent functionality, copy `/etc/inet/mipagent.conf.fa-sample`.

- For home agent functionality, copy `/etc/inet/mipagent.conf.ha-sample`.
- For both foreign agent and home agent functionality, copy `/etc/inet/mipagent.conf-sample`.

3 **You can reboot your system to invoke the boot script that starts the `mipagent` daemon. Or, you can also start `mipagent` by typing the following command:**

```
# /etc/inet.d/mipagent start
```

## ▼ How to Create the Mobile IP Configuration File

1 **Assume the Primary Administrator role, or become superuser, on the system where you want to enable Mobile IP.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Create the `/etc/inet/mipagent.conf` file by using one of the following options:**

- In the `/etc/inet` directory, create an empty file named `mipagent.conf`.
- From the following list, copy the sample file that provides the functionality you want for the `/etc/inet/mipagent.conf` file.
  - `/etc/inet/mipagent.conf.fa-sample`
  - `/etc/inet/mipagent.conf.ha-sample`
  - `/etc/inet/mipagent.conf-sample`

3 **Add or change configuration parameters in the `/etc/inet/mipagent.conf` file to conform to your configuration requirements.**

The remaining procedures in this section describe the steps to modify sections in `/etc/inet/mipagent.conf`.

## ▼ How to Configure the `General` Section

If you copied one of the sample files in the `/etc/inet` directory, you can omit this procedure because the sample file contains this entry. "`General` Section" on page 701 provides descriptions of the labels and values that are used in this section.

● **Edit the `/etc/inet/mipagent.conf` file and add the following lines:**

```
[General]
     Version = 1.0
```

**Note –** The `/etc/inet/mipagent.conf` file must contain this entry.

## ▼ How to Configure the `Advertisements` Section

"`Advertisements` Section" on page 701 provides descriptions of the labels and values that are used in this section.

- **Edit the `/etc/inet/mipagent.conf` file and add or change the following lines by using the values that are required for your configuration.**

```
[Advertisements interface]
     HomeAgent = <yes/no>
     ForeignAgent = <yes/no>
     PrefixFlags = <yes/no>
     AdvertiseOnBcast = <yes/no>
     RegLifetime = n
     AdvLifetime = n
     AdvFrequency = n
     ReverseTunnel = <yes/no/FA/HA/both>
     ReverseTunnelRequired = <yes/no/FA/HA>
```

**Note –** You must include a different `Advertisements` section for each interface on the local host that provides Mobile IP services.

## ▼ How to Configure the `GlobalSecurityParameters` Section

"`GlobalSecurityParameters` Section" on page 703 provides descriptions of the labels and values that are used in this section.

- **Edit the `/etc/inet/mipagent.conf` file and add or change the following lines by using the values that are required for your configuration:**

```
[GlobalSecurityParameters]
     MaxClockSkew = n
     HA-FAauth = <yes/no>
     MN-FAauth = <yes/no>
     Challenge = <yes/no>
     KeyDistribution = files
```

## How to Configure the `Pool` Section

"`Pool` Section" on page 704 provides descriptions of the labels and values that are used in this section:

1. **Edit the `/etc/inet/mipagent.conf` file**

2. **Add or change the following lines by using the values that are required for your configuration:**

```
[Pool pool-identifier]
     BaseAddress = IP-address
     Size = size
```

## How to Configure the `SPI` Section

"`SPI` Section" on page 705 provides descriptions of the labels and values that are used in this section.

1. **Edit the `/etc/inet/mipagent.conf` file.**

2. **Add or change the following lines by using the values that are required for your configuration:**

```
[SPI SPI-identifier]
     ReplayMethod = <none/timestamps>
     Key = key
```

---

**Note –** You must include a different `SPI` section for each security context that is deployed.

---

## How to Configure the `Address` Section

"`Address` Section" on page 706 provides descriptions of the labels and values that are used in this section.

1. **Edit the `/etc/inet/mipagent.conf` file.**

2. **Add or change the following lines by using the values that are required for your configuration:**

   - **For a mobile node, use the following:**

```
[Address address]
     Type = node
     SPI = SPI-identifier
```

   - **For an agent, use the following:**

```
[Address address]
     Type = agent
     SPI = SPI-identifier
     IPsecRequest = action {properties} [: action {properties}]
     IPsecReply = action {properties} [: action {properties}]
     IPsecTunnel = action {properties} [: action {properties}]
```

where *action* and *{properties}* are any action and associated properties that are defined in the `ipsec`(7P) man page.

---

**Note –** The SPI that is configured previously corresponds to the MD5 protection mechanism that is required by RFC 2002. The SPI that is configured previously does not correspond to the SPI that is used by IPsec. For more information about IPsec, see Chapter 19, "IP Security Architecture (Overview)," and Chapter 20, "Configuring IPsec (Tasks)." Also see the `ipsec`(7P) man page.

---

- **For a mobile node that is identified by its NAI, use the following:**

```
[Address NAI]
     Type = Node
     SPI = SPI-identifier
     Pool = pool-identifier
```

- **For a default mobile node, use the following:**

```
[Address Node-Default]
     Type = Node
     SPI = SPI-identifier
     Pool = pool-identifier
```

# Modifying the Mobile IP Configuration File (Task Map)

| Task | Description | For Instructions |
|---|---|---|
| Modify the `General` section. | Uses the `mipagentconfig change` command to change the value of a label in the `General` section of the Mobile IP configuration file. | "How to Modify the `General` Section" on page 685 |
| Modify the `Advertisements` section. | Uses the `mipagentconfig change` command to change the value of a label in the `Advertisements` section of the Mobile IP configuration file. | "How to Modify the `Advertisements` Section" on page 686 |

| | | |
|---|---|---|
| Modify the `GlobalSecurityParameters` section. | Uses the `mipagentconfig change` command to change the value of a label in the `GlobalSecurityParameters` section of the Mobile IP configuration file. | "How to Modify the `GlobalSecurityParameters` Section" on page 687 |
| Modify the `Pool` section. | Uses the `mipagentconfig change` command to change the value of a label in the `Pool` section of the Mobile IP configuration file. | "How to Modify the `Pool` Section" on page 687 |
| Modify the `SPI` section. | Uses the `mipagentconfig change` command to change the value of a label in the `SPI` section of the Mobile IP configuration file. | "How to Modify the `SPI` Section" on page 688 |
| Modify the `Address` section. | Uses the `mipagentconfig change` command to change the value of a label in the `Address` section of the Mobile IP configuration file. | "How to Modify the `Address` Section" on page 688 |
| Add or delete parameters. | Uses the `mipagentconfig add` or `delete` commands to add new parameters, labels, and values or to delete existing ones in any section of the Mobile IP configuration file. | "How to Add or Delete Configuration File Parameters" on page 689 |
| Display the current settings of parameter destinations. | Uses the `mipagentconfig get` command to display current settings of any section of the Mobile IP configuration file. | "How to Display Current Parameter Values in the Configuration File" on page 691 |

# Modifying the Mobile IP Configuration File

This section shows you how to modify the Mobile IP configuration file by using the `mipagentconfig` command. This section also shows you how to display the current settings of parameter destinations.

"Configuring the Mobility IP Agent" on page 710 provides a conceptual description of the `mipagentconfig` command's usage. You can also review the `mipagentconfig`(1M) man page.

## ▼ How to Modify the `General` Section

**1 Assume the Primary Administrator role, or become superuser on the system where you want to enable Mobile IP.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **On a command line, type the following command for each label that you want to modify in the `General` section.**

```
# mipagentconfig change <label> <value>
```

**Example 28–1** Modifying a Parameter in the `General` Section

The following example shows how you might change the version number in the configuration file's `General` section.

```
# mipagentconfig change version 2
```

## ▼ How to Modify the `Advertisements` Section

1 **Assume the Primary Administrator role, or become superuser, on the system where you want to enable Mobile IP.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Type the following command for each label that you want to modify in the `Advertisements` section:**

```
# mipagentconfig change adv device-name <label> <value>
```

For example, if you are changing the agent's advertised lifetime to 300 seconds for device `hme0`, use the following command.

```
# mipagentconfig change adv hme0 AdvLifetime 300
```

**Example 28–2** Modifying the `Advertisements` Section

The following example shows how you might change other parameters in the configuration file's `Advertisements` section.

```
# mipagentconfig change adv hme0 HomeAgent yes
# mipagentconfig change adv hme0 ForeignAgent no
# mipagentconfig change adv hme0 PrefixFlags no
# mipagentconfig change adv hme0 RegLifetime 300
# mipagentconfig change adv hme0 AdvFrequency 4
# mipagentconfig change adv hme0 ReverseTunnel yes
```

## ▼ How to Modify the `GlobalSecurityParameters` Section

1. **Assume the Primary Administrator role, or become superuser, on the system where you want to enable Mobile IP.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Type the following command for each label that you want to modify in the `GlobalSecurityParameters` section:**

   ```
   # mipagentconfig change <label> <value>
   ```

   For example, if you are enabling home agent and foreign agent authentication, use the following command:

   ```
   # mipagentconfig change HA-FAauth yes
   ```

**Example 28–3** Modifying the Global Security Parameters Section

The following example shows how you might change other parameters in the configuration file's `GlobalSecurityParameters` section.

```
# mipagentconfig change MaxClockSkew 200
# mipagentconfig change MN-FAauth yes
# mipagentconfig change Challenge yes
# mipagentconfig change KeyDistribution files
```

## ▼ How to Modify the `Pool` Section

1. **Assume the Primary Administrator role, or become superuser, on the system where you want to enable Mobile IP.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Type the following command for each label that you want to modify in the `Pool` section:**

   ```
   # mipagentconfig change Pool pool-identifier <label> <value>
   ```

**Example 28–4** Modifying the `Pool` Section

The following example shows the commands to use for changing the base address to `192.168.1.1` and the size of Pool `10` to `100`.

```
# mipagentconfig change Pool 10 BaseAddress 192.168.1.1
# mipagentconfig change Pool 10 Size 100
```

## ▼ How to Modify the `SPI` Section

1. **Assume the Primary Administrator role, or become superuser, on the system where you want to enable Mobile IP.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Type the following command for each label that you want to modify in the `SPI` section:**

   ```
   # mipagentconfig change SPI SPI-identifier <label> <value>
   ```

   For example, if you are changing the key for `SPI 257` to `5af2aee39ff0b332`, use the following command.

   ```
   # mipagentconfig change SPI 257 Key 5af2aee39ff0b332
   ```

**Example 28–5** Modifying the `SPI` Section

The following example shows how to change the `ReplayMethod` label in the configuration file's `SPI` section.

```
# mipagentconfig change SPI 257 ReplayMethod timestamps
```

## ▼ How to Modify the `Address` Section

1. **Assume the Primary Administrator role, or become superuser, on the system where you want to enable Mobile IP.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Type the following command for each label that you want to modify in the `Address` section:**

   ```
   # mipagentconfig change addr [NAI | IPaddr | node-default] <label> <value>
   ```

See "Address Section" on page 706 for a description of the three configuration methods (NAI, IP address, and node-default).

For example, if you are changing the SPI of IP address `10.1.1.1` to `258`, use the following command:

```
# mipagentconfig change addr 10.1.1.1 SPI 258
```

**Example 28–6** Modifying the `Address` Section

The following example shows how you can change other parameters that are provided in the sample configuration file's `Address` section.

```
# mipagentconfig change addr 10.1.1.1 Type agent
# mipagentconfig change addr 10.1.1.1 SPI 259
# mipagentconfig change addr mobilenode@abc.com Type node
# mipagentconfig change addr mobilenode@abc.com SPI 258
# mipagentconfig change addr mobilenode@abc.com Pool 2
# mipagentconfig change addr node-default SPI 259
# mipagentconfig change addr node-default Pool 3
# mipagentconfig change addr 10.68.30.36 Type agent
# mipagentconfig change addr 10.68.30.36 SPI 260
# mipagentconfig change IPsecRequest apply {auth_algs md5 sa shared}
```

## ▼ How to Add or Delete Configuration File Parameters

1. **Assume the Primary Administrator role, or become superuser, on the system where you want to enable Mobile IP.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Type the appropriate command for each label that you want to add or delete for the designated section:**

   - For the `General` section use the following:

     ```
     # mipagentconfig [add | delete] <label> <value>
     ```

   - For the `Advertisements` section use the following:

     ```
     # mipagentconfig [add | delete] adv device-name <label> <value>
     ```

**Note –** You can add an interface by typing the following:

```
# mipagentconfig add adv device-name
```

In this instance, default values are assigned to the interface (for both the foreign agent and the home agent).

- For the `GlobalSecurityParameters`, section use the following:

```
# mipagentconfig [add | delete] <label> <value>
```

- For the `Pool` section, use the following:

```
# mipagentconfig [add | delete] Pool pool-identifier <label> <value>
```

- For the `SPI` section, use the following:

```
# mipagentconfig [add | delete] SPI SPI-identifier <label> <value>
```

- For the `Address` section, use the following:

```
# mipagentconfig [add | delete] addr [NAI | IP-address | node-default] \
<label> <value>
```

**Note –** Do not create identical `Advertisements`, `Pool`, `SPI`, and `Address` sections.

**Example 28–7** Modifying File Parameters

For example, to create a new address pool, Pool 11, that has a base address of `192.167.1.1` and a size of `100`, use the following commands.

```
# mipagentconfig add Pool 11 BaseAddress 192.167.1.1
# mipagentconfig add Pool 11 size 100
```

**Example 28–8** Deleting SPI

The following example shows how to delete the SPI security parameter `SPI 257`.

```
# mipagentconfig delete SPI 257
```

## ▼ How to Display Current Parameter Values in the Configuration File

You can use the `mipagentconfig get` command to display current settings that are associated with parameter destinations.

1 **Assume the Primary Administrator role, or become superuser, on the system where you are enabling Mobile IP.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Type the following command for each parameter for which you want to display settings:**

```
# mipagentconfig get [<parameter> | <label>]
```

For example, if you are displaying the advertisement settings for the `hme0` device, use the following command:

```
# mipagentconfig get adv hme0
```

As a result, the following output might be displayed:

```
[Advertisements hme0]
   HomeAgent = yes
   ForeignAgent = yes
```

**Example 28–9** Using the `mipagentconfig get` Command to Display Parameter Values

The following example shows the results of using the `mipagentconfig get` command with other parameter destinations.

```
# mipagentconfig get MaxClockSkew
      [GlobalSecurityParameters]
         MaxClockSkew=300

# mipagentconfig get HA-FAauth
      [GlobalSecurityParameters]
         HA-FAauth=no

# mipagentconfig get MN-FAauth
      [GlobalSecurityParameters]
         MN-FAauth=no

# mipagentconfig get Challenge
      [GlobalSecurityParameters]
         Challenge=no
```

```
# mipagentconfig get Pool 10
      [Pool 10]
         BaseAddress=192.168.1.1
         Size=100

# mipagentconfig get SPI 257
      [SPI 257]
         Key=11111111111111111111111111111111
         ReplayMethod=none

# mipagentconfig get SPI 258
      [SPI 258]
         Key=15111111111111111111111111111111
         ReplayMethod=none

# mipagentconfig get addr 10.1.1.1
      [Address 10.1.1.1]
         SPI=258
         Type=agent

# mipagentconfig get addr 192.168.1.200
      [Address 192.168.1.200]
         SPI=257
         Type=node# mipagentconfig get addr 10.1.1.1
      [Address 10.1.1.1]
         Type=agent
         SPI=258
         IPsecRequest = apply {auth_algs md5 sa shared}
         IPsecReply = permit {auth_algs md5}
         IPsecTunnel = apply {encr_algs 3des sa shared}
```

# Displaying Mobility Agent Status

You can use the `mipagentstat` command to display a foreign agent's visitors list and a home agent's binding table. "Mobile IP Mobility Agent Status" on page 711 provides a conceptual description of the `mipagentstat` command. You can also review the `mipagentstat`(1M) man page.

## ▼ How to Display Mobility Agent Status

**1 Become superuser or assume an equivalent role on the system where you are enabling Mobile IP.**

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**2 Display the mobility agent status.**

```
# mipagentstat options
```

`-f` Shows the list of active mobile nodes in the foreign agent's visitor list.

`-h` Shows the list of active mobile nodes in the home agent's binding table.

`-p` Shows the list of security associations with an agent's mobility agent peers.

**Example 28–10** Displaying Mobility Agent Status

This example shows how to display the visitor list for all mobile nodes that are registered with a foreign agent.

```
# mipagentstat -f
```

As a result, output similar to the following is displayed:

```
Mobile Node     Home Agent     Time (s)     Time (s)  Flags
                               Granted      Remaining
--------------- -------------- ------------ --------- -----
foobar.xyz.com  ha1.xyz.com    600          125       .....T.
10.1.5.23       10.1.5.1       1000         10        .....T.
```

This example shows how to display foreign agent security associations.

```
# mipagentstat -p
```

As a result, output similar to the following is displayed:

```
Foreign                 ..... Security Association(s).....
Agent                   Requests Replies  FTunnel  RTunnel
----------------------  -------- -------- -------- --------
forn-agent.eng.sun.com  AH       AH       ESP      ESP
```

This example shows how to display home agent security associations.

```
# mipagentstat -fp
```

As a result, output similar to the following is displayed:

```
Home                    ..... Security Association(s) .....
Agent                   Requests Replies  FTunnel  RTunnel
----------------------  -------- -------- -------- --------
home-agent.eng.sun.com  AH       AH       ESP      ESP
ha1.xyz.com             AH,ESP   AH       AH,ESP   AH,ESP
```

# Displaying Mobility Routes on a Foreign Agent

You can use the `netstat` command to display additional information about source-specific routes that are created by forward tunnels and reverse tunnels. See the `netstat`(1M) man page for more information about this command.

## ▼ How to Display Mobility Routes on a Foreign Agent

**1 Become superuser or assume an equivalent role on the system where you are enabling Mobile IP.**

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

**2 Display the mobility routes.**

```
# netstat -rn
```

**Example 28–11** Displaying Mobility Routes on a Foreign Agent

The following example shows the routes for a foreign agent that uses a reverse tunnel.

```
Routing Table:   IPv4 Source-Specific
Destination      In If     Source      Gateway Flags  Use  Out If
--------------  ------- ------------ --------- -----  ---- -------
10.6.32.11      ip.tun1      --      10.6.32.97  UH      0 hme1
    --          hme1    10.6.32.11       --      U       0 ip.tun1
```

The first line indicates that the destination IP address `10.6.32.11` and the incoming interface `ip.tun1` select `hme1` as the interface that forwards the packets. The next line indicates that any packet originating from interface `hme1` and source address `10.6.32.11` must be forwarded to `ip.tun1`.

◆ ◆ ◆ CHAPTER 29

# Mobile IP Files and Commands (Reference)

This chapter describes the components that are provided with the Solaris implementation of Mobile IP. To use Mobile IP, you must first configure the Mobile IP configuration file by using the parameters and commands that are described in this chapter.

This chapter contains the following information:


- "Overview of the Solaris Mobile IP Implementation" on page 695
- "Mobile IP Configuration File" on page 696
- "Configuring the Mobility IP Agent" on page 710
- "Mobile IP Mobility Agent Status" on page 711
- "Mobile IP State Information" on page 712
- "`netstat` Extensions for Mobile IP" on page 712
- "`snoop` Extensions for Mobile IP" on page 712


**Note –** The Mobile IP feature is removed from Solaris 10 updates after Solaris 10 8/07.

## Overview of the Solaris Mobile IP Implementation

The mobility agent software incorporates home agent and foreign agent functionality. The Solaris Mobile IP software does not provide a client mobile node. Only the agent functionality is provided. Each network with mobility support should have at least one static (non-mobile) host running this software.

The following RFC functions are supported in the Solaris implementation of Mobile IP:

- RFC 1918, "Address Allocation for Private Internets" (`http://www.ietf.org/rfc/rfc1918.txt?number=1918`)
- RFC 2002, "IP Mobility Support" (Agent only) (`http://www.ietf.org/rfc/rfc2002.txt?number=2002`)
- RFC 2003, "IP Encapsulation Within IP" (`http://www.ietf.org/rfc/rfc2003.txt?number=2003`)

- RFC 2794, "Mobile IP Network Access Identifier Extension for IPv4" (`http://www.ietf.org/rfc/rfc2794.txt?number=2794`)
- RFC 3012, "Mobile IPv4 Challenge/Response Extensions" (`http://www.ietf.org/rfc/rfc3012.txt?number=3012`)
- RFC 3024, "Reverse Tunneling for Mobile IP" (`http://www.ietf.org/rfc/rfc3024.txt?number=3024`)

The base Mobile IP protocol (RFC 2002) does not address the problem of scalable key distribution and treats key distribution as an orthogonal issue. The Solaris Mobile IP software utilizes only manually configured keys, specified in a configuration file.

The following RFC functions are not supported in the Solaris implementation of Mobile IP:

- RFC 1701, "General Routing Encapsulation" (`http://www.ietf.org/rfc/rfc1701.txt?number-1701`)
- RFC 2004, "Minimal Encapsulation Within IP" (`http://www.ietf.org/rfc/rfc2004.txt?number=2004`)

The following functions are not supported in the Solaris implementation of Mobile IP:

- The forwarding of multicast traffic or broadcast traffic by the home agent to the foreign agent for a mobile node that is visiting a foreign network
- The routing of broadcast and multicast datagrams through reverse tunnels
- Private care-of addresses or private home agent addresses

See the `mipagent`(1M) man page for additional information.

# Mobile IP Configuration File

The `mipagent` command reads configuration information from the `/etc/inet/mipagent.conf` configuration file at startup. Mobile IP uses the `/etc/inet/mipagent.conf` configuration file to initialize the Mobile IP mobility agent. When configured and deployed, the mobility agent issues periodic router advertisements and responds to router discovery solicitation messages as well as Mobile IP registration messages.

See the `mipagent.conf`(4) man page for a description of file attributes. See the `mipagent`(1M) man page for a description of this file's usage.

## Configuration File Format

The Mobile IP configuration file consists of sections. Each section has a unique name and is enclosed in square brackets. Each section contains one or more labels. You assign values to the labels by using the following format:

```
[Section_name]
     Label-name = value-assigned
```

"Configuration File Sections and Labels" on page 701 describes the section names, labels, and possible values.

# Sample Configuration Files

The default Solaris installation provides the following sample configuration files in the `/etc/inet` directory:

- `mipagent.conf-sample` – Contains a sample configuration for a Mobile IP agent that provides both foreign agent and home agent functionality
- `mipagent.conf.fa-sample` – Contains a sample configuration for a Mobile IP agent that provides only foreign agent functionality
- `mipagent.conf.ha-sample` – Contains a sample configuration for a Mobile IP agent that provides only home agent functionality

These sample configuration files contain mobile node address and security settings. Before you can implement Mobile IP, you must create a configuration file with the name `mipagent.conf` and place it in the `/etc/inet` directory. This file contains the configuration settings that satisfy your Mobile IP implementation requirements. You can also choose one of the sample configuration files, modify it with your addresses and security settings, and copy it to `/etc/inet/mipagent.conf`.

For more information, see "How to Create the Mobile IP Configuration File" on page 681.

## `mipagent.conf-sample` File

The following listing shows the sections, labels, and values that are contained in the `mipagent.conf-sample` file. "Configuration File Sections and Labels" on page 701 describes the syntax, sections, labels, and values.

```
[General]
   Version = 1.0    # version number for the configuration file. (required)

[Advertisements hme0]
   HomeAgent = yes
   ForeignAgent = yes
   PrefixFlags = yes
   AdvertiseOnBcast = yes
   RegLifetime = 200
   AdvLifetime = 200
   AdvFrequency = 5
   ReverseTunnel = no
```

```
   ReverseTunnelRequired = no

[GlobalSecurityParameters]
   MaxClockSkew = 300
   HA-FAauth = yes
   MN-FAauth = yes
   Challenge = no
   KeyDistribution = files

[Pool 1]
   BaseAddress = 10.68.30.7
   Size = 4

[SPI 257]
   ReplayMethod = none
   Key = 11111111111111111111111111111111

[SPI 258]
   ReplayMethod = none
   Key = 15111111111111111111111111111111

[Address 10.1.1.1]
   Type = node
   SPI = 258

[Address mobilenode@sun.com]
   Type = node
   SPI = 257
   Pool = 1

[Address Node-Default]
   Type = node
   SPI = 258
   Pool = 1

[Address 10.68.30.36]
   Type = agent
   SPI = 257[Address 10.68.30.36]
   Type = agent
   SPI = 257
   IPsecRequest = apply {auth_algs md5 sa shared}
   IPsecReply = permit {auth_algs md5}
   IPsecTunnel =  apply {encr_algs 3des sa shared}
```

## `mipagent.conf.fa-sample` File

The following listing shows the sections, labels, and values that are contained in the `mipagent.conf.fa-sample` file. "Configuration File Sections and Labels" on page 701 describes the syntax, sections, labels, and values.

The `mipagent.conf.fa-sample` file shows a configuration that provides only foreign agent functionality. This sample file does not contain a `Pool` section because pools are used only by a home agent. Otherwise, this file is the same as the `mipagent.conf-sample` file.

```
[General]
   Version = 1.0    # version number for the configuration file. (required)

[Advertisements hme0]
   HomeAgent = no
   ForeignAgent = yes
   PrefixFlags = yes
   AdvertiseOnBcast = yes
   RegLifetime = 200
   AdvLifetime = 200
   AdvFrequency = 5
   ReverseTunnel = yes
   ReverseTunnelRequired = no

[GlobalSecurityParameters]
   MaxClockSkew = 300
   HA-FAauth = yes
   MN-FAauth = yes
   Challenge = no
   KeyDistribution = files

[SPI 257]
   ReplayMethod = none
   Key = 11111111111111111111111111111111

[SPI 258]
   ReplayMethod = none
   Key = 15111111111111111111111111111111

[Address 10.1.1.1]
   Type = node
   SPI = 258

[Address 10.68.30.36]
   Type = agent
   SPI = 257[Address 10.68.30.36]
   Type = agent
   SPI = 257
   IPsecRequest = apply {auth_algs md5 sa shared}
   IPsecReply = permit {auth_algs md5}
   IPsecTunnel = apply {encr_algs 3des sa shared}
```

## `mipagent.conf.ha-sample` File

The following listing shows the sections, labels, and values that are contained in the `mipagent.conf.ha-sample` file. "Configuration File Sections and Labels" on page 701 describes the syntax, sections, labels, and values.

The `mipagent.conf.ha-sample` file shows a configuration that provides only home agent functionality. Otherwise, this file is the same as the `mipagent.conf-sample` file.

```
[General]
   Version = 1.0    # version number for the configuration file. (required)

[Advertisements hme0]
   HomeAgent = yes
   ForeignAgent = no
   PrefixFlags = yes
   AdvertiseOnBcast = yes
   RegLifetime = 200
   AdvLifetime = 200
   AdvFrequency = 5
   ReverseTunnel = yes
   ReverseTunnelRequired = no

[GlobalSecurityParameters]
   MaxClockSkew = 300
   HA-FAauth = yes
   MN-FAauth = yes
   Challenge = no
   KeyDistribution = files

[Pool 1]
   BaseAddress = 10.68.30.7
   Size = 4

[SPI 257]
   ReplayMethod = none
   Key = 11111111111111111111111111111111

[SPI 258]
   ReplayMethod = none
   Key = 15111111111111111111111111111111

[Address 10.1.1.1]
   Type = node
   SPI = 258

[Address mobilenode@sun.com]
   Type = node
```

```
   SPI = 257
   Pool = 1

[Address Node-Default]
   Type = node
   SPI = 258
   Pool = 1[Address 10.68.30.36]
    Type = agent
    SPI = 257
    IPsecRequest = apply {auth_algs md5 sa shared}
    IPsecReply = permit {auth_algs md5}
    IPsecTunnel = apply {encr_algs 3des sa shared}
```

# Configuration File Sections and Labels

The Mobile IP configuration file contains the following sections:

- `General` (Required)
- `Advertisements` (Required)
- `GlobalSecurityParameters` (Optional)
- `Pool` (Optional)
- `SPI` (Optional)
- `Address` (Optional)

The `General` and `GlobalSecurityParameters` sections contain information relevant to the operation of the Mobile IP agent. These sections can appear only once in the configuration file.

## `General` Section

The `General` section contains only one label: the version number of the configuration file. The `General` section has the following syntax:

```
[General]
     Version = 1.0
```

## `Advertisements` Section

The `Advertisements` section contains the `HomeAgent` and `ForeignAgent` labels, as well as other labels. You must include a different `Advertisements` section for each interface on the local host that provides Mobile IP services. The `Advertisements` section has the following syntax:

```
[Advertisements interface]
     HomeAgent = <yes/no>
     ForeignAgent = <yes/no>
     .
     .
```

Typically, your system has a single interface, such as `eri0` or `hme0`, and supports both home agent and foreign agent operations. If this situation exists for the example `hme0`, then the `yes` value is assigned to both the `HomeAgent` and `ForeignAgent` labels as follows:

```
[Advertisements hme0]
     HomeAgent = yes
     ForeignAgent = yes
     .
     .
```

For advertisement over dynamic interfaces, use '*' for the device ID part. For example, *Interface-name* ppp* actually implies all PPP interfaces that are configured after the `mipagent` daemon has been started. All the attributes in the advertisement section of a dynamic interface type remain the same.

The following table describes the labels and values that you can use in the `Advertisements` section.

**TABLE 29–1** `Advertisements` Section Labels and Values

| Label | Value | Description |
|---|---|---|
| `HomeAgent` | `yes` or `no` | Determines if the `mipagent` daemon provides home agent functionality. |
| `ForeignAgent` | `yes` or `no` | Determines if `mipagent` provides foreign agent functionality. |
| `PrefixFlags` | `yes` or `no` | Specifies if advertisements include the optional prefix-length extension. |
| `AdvertiseOnBcast` | `yes` or `no` | If `yes`, advertisements are sent on `255.255.255.255`, rather than `224.0.0.1`. |
| `RegLifetime` | `n` | The maximum lifetime value that is accepted in registration requests, in seconds. |
| `AdvLifetime` | `n` | The maximum length of time that the advertisement is considered valid in the absence of further advertisements, in seconds. |
| `AdvFrequency` | `n` | Time between two consecutive advertisements, in seconds. |

**TABLE 29–1** `Advertisements` Section Labels and Values *(Continued)*

| Label | Value | Description |
|---|---|---|
| `ReverseTunnel` | `yes` or `noFA` or `HA` or `both` | Determines if `mipagent` provides reverse-tunnel functionality.<br>The value `yes` means that both the foreign agent and home agent support reverse tunneling. The value `no` means that the interface does not support reverse tunneling.<br>The value `FA` means that the foreign agent supports reverse tunneling. The value `HA` means that the home agent supports reverse tunneling. The value `both` means that both the foreign agent and home agent support reverse tunneling. |
| `ReverseTunnelRequired` | `yes` or `no` | Determines if `mipagent` requires reverse tunnel functionality. Consequently, determines if a mobile node must request a reverse tunnel during registration.<br>The value `yes` means that both the foreign agent and home agent require a reverse tunnel. The value `no` means that the interface does not require a reverse tunnel.<br>The value `FA` means that the foreign agent requires a reverse tunnel. The value `HA` means that the home agent requires a reverse tunnel. |
| `AdvInitCount` | `n` | Determines the initial number of unsolicited advertisements. The default value is 1. This value is meaningful only if `AdvLimitUnsolicited` is `yes`. |
| `AdvLimitUnsolicited` | `yes` or `no` | Enables or disables a limited number of unsolicited advertisements over the mobility interface. |

## `GlobalSecurityParameters` Section

The `GlobalSecurityParameters` section contains the labels `maxClockSkew`, `HA-FAauth`, `MN-FAauth`, `Challenge`, and `KeyDistribution`. This section has the following syntax:

```
[GlobalSecurityParameters]
     MaxClockSkew = n
     HA-FAauth = <yes/no>
     MN-FAauth = <yes/no>
     Challenge = <yes/no>
     KeyDistribution = files
```

The Mobile IP protocol provides message replay protection by allowing timestamps to be present in the messages. If the clocks differ, the home agent returns an error to the mobile node

with the current time and the mobile node can register again by using the current time. You use the `MaxClockSkew` label to configure the maximum number of seconds that differ between the home agent and the mobile node's clocks. The default value is 300 seconds.

The `HA-FAauth` and `MN-FAauth` labels enable or disable the requirement for home-foreign and mobile-foreign authentication, respectively. The default value is disabled. You use the `challenge` label so that the foreign agent issues challenges to the mobile node in its advertisements. The label is used for replay protection. The default value is disabled here, also.

The following table describes the labels and values that you can use in the `GlobalSecurityParameters` section.

**TABLE 29–2** `GlobalSecurityParameters` Section Labels and Values

| Label | Value | Description |
|---|---|---|
| `MaxClockSkew` | `n` | The number of seconds that `mipagent` accepts as a difference between its own local time and the time that is found in registration requests |
| `HA-FAauth` | `yes` or `no` | Specifies if HA-FA authentication extensions must be present in registration requests and replies |
| `MN-FAauth` | `yes` or `no` | Specifies if MN-FA authentication extensions must be present in registration requests and replies |
| `Challenge` | `yes` or `no` | Specifies if the foreign agent includes challenges in its mobility advertisements |
| `KeyDistribution` | `files` | Must be set to files |

## `Pool` Section

Mobile nodes can be assigned dynamic addresses by the home agent. Dynamic address assignment is done within the `mipagent` daemon independently of DHCP. You can create an address pool that can be used by mobile nodes by requesting a home address. Address pools are configured through the `Pool` section in the configuration file.

The `Pool` section contains the `BaseAddress` and `Size` labels. The `Pool` section has the following syntax:

```
[Pool pool-identifier]
     BaseAddress = IP-address
     Size = size
```

**Note –** If you use a `Pool` identifier, then it must also exist in the mobile node's `Address` section.

You use the `Pool` section to define address pools that can be assigned to the mobile nodes. You use the `BaseAddress` label to set the first IP address in the pool. You use the `Size` label to specify the number of addresses available in the pool.

For example, if IP addresses `192.168.1.1` through `192.168.1.100` are reserved in pool 10, the `Pool` section has the following entry:

```
[Pool 10]
     BaseAddress = 192.168.1.1
     Size = 100
```

---

**Note –** Address ranges should not encompass the broadcast address. For example, you should not assign `BaseAddress = 192.168.1.200` and `Size = 60`, because this range encompasses the broadcast address `192.168.1.255`.

---

The following table describes the labels and values that are used in the `Pool` section.

**TABLE 29–3** `Pool` Section Labels and Values

| Label | Value | Description |
|---|---|---|
| `BaseAddress` | `n.n.n.n` | First address in the address pool |
| `Size` | `n` | Number of addresses in the pool |

## `SPI` Section

Because the Mobile IP protocol requires message authentication, you must identify the security context by using a security parameter index (SPI). You define the security context in the `SPI` section. You must include a different `SPI` section for each security context that is defined. A numerical ID identifies the security context. The Mobile IP protocol reserves the first 256 SPIs. Therefore, you should use only SPI values greater than 256. The `SPI` section contains security-related information, such as shared secrets and replay protection.

The `SPI` section also contains the `ReplayMethod` and `Key` labels. The `SPI` section has the following syntax:

```
[SPI SPI-identifier]
     ReplayMethod = <none/timestamps>
     Key = key
```

Two communicating peers must share the same SPI identifier. You must configure them with the same key and replay method. You specify the key as a string of hexadecimal digits. The maximum length is 16 bytes. For example, if the key is 16 bytes long, and contains the hexadecimal values `0` through `f`, the key string might resemble the following:

```
Key = 0102030405060708090a0b0c0d0e0f10
```

Keys must have an even number of digits, corresponding to the two digits per byte representation.

The following table describes the labels and values that you can use in the `SPI` section.

**TABLE 29–4** `SPI` Section Labels and Values

| Label | Value | Description |
|---|---|---|
| `ReplayMethod` | `none` or `timestamps` | Specifies the type of replay authentication used for the SPI |
| `Key` | `x` | Authentication key in hexadecimal |

## `Address` Section

The Solaris implementation of Mobile IP enables you to configure mobile nodes using one of three methods. Each method is configured in the `Address` section. The first method follows the traditional Mobile IP protocol, and requires that each mobile node have a home address. The second method enables a mobile node to be identified through its Network Access Identifier (NAI). The last method enables you to configure a *default* mobile node, which can be used by any mobile node that has the proper SPI value and related keying material.

### Mobile Node

The `Address` section for a mobile node contains the `Type` and `SPI` labels that define the address type and SPI identifier. The `Address` section has the following syntax:

```
[Address address]
     Type = node
     SPI = SPI-identifier
```

You must include an `Address` section in a home agent's configuration file for each mobile node that is supported.

If Mobile IP message authentication is required between the foreign agent and home agent, you must include an `Address` section for each peer with which an agent needs to communicate.

The SPI value that you configure must represent an `SPI` section that is present in the configuration file.

You can also configure private addresses for a mobile node.

The following table describes the labels and values that you can use in the `Address` section for a mobile node.

**TABLE 29–5** `Address` Section Labels and Values (Mobile Node)

| Label | Value | Description |
|---|---|---|
| `Type` | `node` | Specifies that the entry is for a mobile node |
| `SPI` | `n` | Specifies the SPI value for the associated entry |

## Mobility Agent

The `Address` section for a mobility agent contains the `Type` and `SPI` labels that define the address type and SPI identifier. This section also contains IPsec request, reply, and tunnel labels. The `Address` section for a mobility agent has the following syntax:

```
[Address address]
     Type = agent
     SPI = SPI-identifier
     IPsecRequest = action {properties} [: action {properties}]
     IPsecReply = action {properties} [: action {properties}]
     IPsecTunnel = action {properties} [: action {properties}]
```

You must include an `Address` section in a home agent's configuration file for each mobility agent that is supported.

If Mobile IP message authentication is required between the foreign agent and the home agent, you must include an `Address` section for each peer with which an agent needs to communicate.

The SPI value that you configure must represent an `SPI` section that is present in the configuration file.

The following table describes the labels and values that you can use in the `Address` section for a mobility agent.

**TABLE 29–6** `Address` Section Labels and Values (Mobility Agent)

| Label | Value | Description |
|---|---|---|
| `Type` | `agent` | Specifies that the entry is for a mobility agent |
| `SPI` | `n` | Specifies the SPI value for the associated entry |
| `IPsecRequest` | `apply` or `permit` (see following note) | IPsec properties to invoke for registration requests to and from this mobility agent peer |
| `IPsecReply` | `apply` or `permit` (see following note) | IPsec properties to invoke for registration replies to and from this mobility agent peer |
| `IPsecTunnel` | `apply` or `permit` (see following note) | IPsec properties to invoke for tunnel traffic to and from this mobility agent peer |

**Note –** The `apply` values correspond to outbound datagrams. The `permit` values correspond to inbound datagrams. Therefore, `IPsecRequest apply` values and `IPsecReply permit` values are used by the foreign agent to send and receive registration datagrams. The `IPsecRequest permit` values and the `IPsecReply apply` values are used by the home agent to receive and send registration datagrams.

## Mobile Node Identified by Its NAI

The `Address` section for a mobile node that is identified by its NAI contains the `Type`, `SPI`, and `Pool` labels. The NAI parameter enables you to identify mobile nodes through their NAI. The `Address` section, using the NAI parameter, has the following syntax:

```
[Address NAI]
     Type = Node
     SPI = SPI-identifier
     Pool = pool-identifier
```

To use pools, you identify mobile nodes through their NAI. The `Address` section permits you to configure an NAI, as opposed to a home address. An NAI uses the format `user@domain`. You use the `Pool` label to specify which address pool to use in order to allocate the home address to the mobile node.

The following table describes the labels and values that you can use in the `Address` section for a mobile node that is identified by its NAI.

**TABLE 29–7** `Address` Section Labels and Values (Mobile Node Identified by Its NAI)

| Label | Value | Description |
|---|---|---|
| `Type` | `node` | Specifies that the entry is for a mobile node |
| `SPI` | `n` | Specifies the SPI value for the associated entry |
| `Pool` | `n` | Allocates the pool from which an address is assigned to a mobile node |

You must have corresponding `SPI` and `Pool` sections for the `SPI` and `Pool` labels that are defined in an `Address` section with a mobile node that is identified by its NAI, as shown in the following figure.

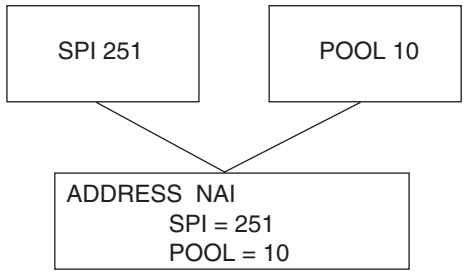



**FIGURE 29–1** Corresponding `SPI` and `Pool` Sections for Address Section With Mobile Node Identified by Its NAI

## Default Mobile Node

The `Address` section for a default mobile node contains the `Type`, `SPI`, and `Pool` labels. The `Node-Default` parameter enables you to permit all mobile nodes to get service if they have the correct SPI (defined in this section). The `Address` section, using the `Node-Default` parameter, has the following syntax:

```
[Address Node-Default]
     Type = Node
     SPI = SPI-identifier
     Pool = pool-identifier
```

The `Node-Default` parameter enables you to reduce the size of the configuration file. Otherwise, each mobile node requires its own section. However, the `Node-Default` parameter does pose a security risk. If a mobile node is no longer trusted for any reason, you need to update the security information on all trusted mobile nodes. This task can be very tedious. However, you can use the `Node-Default` parameter in networks that consider security risks unimportant.

The following table describes the labels and values that you can use in the `Address` section for a default mobile node.

**TABLE 29–8** `Address` Section Labels and Values (Default Mobile Node)

| Label | Value | Description |
|---|---|---|
| `Type` | `node` | Specifies that the entry is for a mobile node |
| `SPI` | `n` | Specifies the SPI value for the associated entry |
| `Pool` | `n` | Allocates the pool from which an address is assigned to a mobile node |

You must have corresponding `SPI` and `Pool` sections for the `SPI` and `Pool` labels that are defined in the `Address` section with a default mobile node, as shown in the following figure.

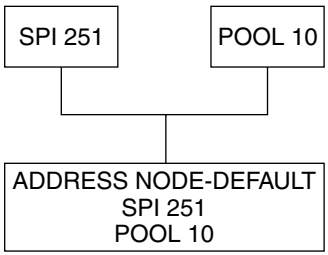



**FIGURE 29–2** Corresponding `SPI` and `Pool` Sections for Address Section With a Default Mobile Node

# Configuring the Mobility IP Agent

You can use the `mipagentconfig` command to configure the mobility agent. This command enables you to create or modify any parameter in the `/etc/inet/mipagent.conf` configuration file. Specifically, you can change any setting. Also, you can add or delete mobility clients, pools, and SPIs. The `mipagentconfig` command has the following syntax:

```
# mipagentconfig <command> <parameter> <value>
```

The following table describes the commands that you can use with `mipagentconfig` to create or modify parameters in the `/etc/inet/mipagent.conf` configuration file.

**TABLE 29–9** `mipagentconfig` Subcommands

| Command | Description |
|---|---|
| `add` | Used to add advertisement parameters, security parameters, SPIs, and addresses to the configuration file |
| `change` | Used to change advertisement parameters, security parameters, SPIs, and addresses in the configuration file |
| `delete` | Used to delete advertisement parameters, security parameters, SPIs, and addresses from the configuration file |
| `get` | Used to display current values in the configuration file |

See the `mipagentconfig`(1M) man page for a description of command parameters and acceptable values. "Modifying the Mobile IP Configuration File" on page 685 provides procedures that use the `mipagentconfig` command.

# Mobile IP Mobility Agent Status

You can use the `mipagentstat` command to display a foreign agent's visitor list and a home agent's binding table. You can also display the security associations with an agent's mobility agent peers. To display the foreign agent visitor list, you use the `mipagentstat` command's `-f` option. To display the home agent binding table, you use the `mipagentstat` command's `-h` option. To display the security associations with an agent's mobility agent peers, you use the `mipagentstat` command's `-p` option. The following examples show typical output when the `mipagentstat` command is used with these options.

**EXAMPLE 29–1** Foreign Agent Visitor List

```
Mobile Node     Home Agent     Time (s)     Time (s)  Flags
                               Granted      Remaining
--------------- -------------- ------------ --------- -----
foobar.xyz.com  ha1.xyz.com    600          125       .....T.
10.1.5.23       10.1.5.1       1000         10        .....T.
```

**EXAMPLE 29–2** Home Agent Binding Table

```
Mobile Node     Home Agent     Time (s)     Time (s)  Flags
                               Granted      Remaining
--------------- -------------- ------------ --------- -----
foobar.xyz.com  fa1.tuv.com    600          125       .....T.
10.1.5.23       123.2.5.12     1000         10        .....T.
```

**EXAMPLE 29–3** Mobility Agent Peer Security Association Table

```
Foreign                  ..... Security Association(s).....
Agent                    Requests Replies  FTunnel  RTunnel
----------------------   -------- -------- -------- --------
forn-agent.eng.sun.com   AH       AH       ESP      ESP

Home                     ..... Security Association(s) .....
Agent                    Requests Replies  FTunnel  RTunnel
----------------------   -------- -------- -------- --------
home-agent.eng.sun.com   AH       AH       ESP      ESP
ha1.xyz.com              AH,ESP   AH       AH,ESP   AH,ESP
```

See the `mipagentstat`(1M) man page for more information about the command's options. "Displaying Mobility Agent Status" on page 692 provides procedures that use the `mipagentstat` command.

# Mobile IP State Information

On shutdown, the `mipagent` daemon stores internal state information in `/var/inet/mipagent_state`. This event occurs only when the `mipagent` provides services as a home agent. This state information includes the list of mobile nodes that are being supported as a home agent, their current care-of addresses, and remaining registration lifetimes. This state information also includes the security association configuration with mobility agent peers. If the `mipagent` daemon is terminated for maintenance and restarted, `mipagent_state` is used to recreate as much of the mobility agent's internal state as possible. The intention is to minimize service disruption for mobile nodes that might be visiting other networks. If `mipagent_state` exists, it is read immediately after `mipagent.conf` every time `mipagent` is started or restarted.

# `netstat` Extensions for Mobile IP

Mobile IP extensions have been added to the `netstat` command to identify Mobile IP forwarding routes. Specifically, you can use the `netstat` command to display a new routing table that is called "Source-Specific." See the `netstat`(1M) man page for more information.

The following example shows the output of `netstat` when you use the `-nr` flags.

**EXAMPLE 29–4** Mobile IP Output From `netstat` Command

```
Routing Table:   IPv4 Source-Specific
Destination      In If     Source      Gateway Flags  Use  Out If
--------------  ------- ------------ --------- -----  ---- -------
10.6.32.11      ip.tun1      --      10.6.32.97  UH      0 hme1
    --          hme1    10.6.32.11       --      U       0 ip.tun1
```

The example shows the routes for a foreign agent that uses a reverse tunnel. The first line indicates that the destination IP address `10.6.32.11` and the incoming interface `ip.tun1` select `hme1` as the interface that forwards the packets. The next line indicates that any packet that originates from interface `hme1` and source address `10.6.32.11` must be forwarded to `ip.tun1`.

# `snoop` Extensions for Mobile IP

Mobile IP extensions have been added to the `snoop` command to identify Mobile IP traffic on the link. See the `snoop`(1M) man page for more information.

The following example shows the output of `snoop` that runs on the mobile node, `mip-mn2`.

**EXAMPLE 29–5** Mobile IP Output From `snoop` Command

```
mip-mn2# snoop
Using device /dev/hme (promiscuous mode)
  mip-fa2 -> 224.0.0.1    ICMP Router advertisement (Lifetime 200s [1]:
```

**EXAMPLE 29–5** Mobile IP Output From snoop Command *(Continued)*

```
{mip-fa2-80 2147483648}), (Mobility Agent Extension), (Prefix Lengths),
(Padding)
  mip-mn2 -> mip-fa2   Mobile IP reg rqst
  mip-fa2 -> mip-mn2   Mobile IP reg reply (OK code 0)
```

This example shows that the mobile node received one of the periodically sent mobility agent advertisements from the foreign agent, `mip-fa2`. Then, `mip-mn2` sent a registration request to `mip-fa2`, and in response, received a registration reply. The registration reply indicates that the mobile node successfully registered with its home agent.

The `snoop` command also supports IPsec extensions. Consequently, you can show how registration and tunnel packets are being protected.

PART VI

# IPMP

This part introduces IP network multipathing (IPMP) and contains tasks for administering IPMP. IPMP provides failure detection and failover for interfaces on a system that are attached to the same link.

◆ ◆ ◆ CHAPTER 30

# Introducing IPMP (Overview)

IP network multipathing (IPMP) provides physical interface failure detection and transparent network access failover for a system with multiple interfaces on the same IP link. IPMP also provides load spreading of packets for systems with multiple interfaces.

This chapter contains the following information:


- "Why You Should Use IPMP" on page 717
- "Basic Requirements of IPMP" on page 721
- "IPMP Addressing" on page 721
- "Solaris IPMP Components" on page 718
- "IPMP Interface Configurations" on page 724
- "IPMP Failure Detection and Recovery Features" on page 725
- "IPMP and Dynamic Reconfiguration" on page 729


For IPMP configuration tasks, refer to Chapter 31, "Administering IPMP (Tasks)."

## Why You Should Use IPMP

IPMP provides increased reliability, availability, and network performance for systems with multiple physical interfaces. Occasionally, a physical interface or the networking hardware attached to that interface might fail or require maintenance. Traditionally, at that point, the system can no longer be contacted through any of the IP addresses that are associated with the failed interface. Additionally, any existing connections to the system using those IP addresses are disrupted.

By using IPMP, you can configure one or more physical interfaces into an IP multipathing group, or *IPMP group*. After configuring IPMP, the system automatically monitors the interfaces in the IPMP group for failure. If an interface in the group fails or is removed for maintenance, IPMP automatically migrates, or *fails over*, the failed interface's IP addresses. The recipient of these addresses is a functioning interface in the failed interface's IPMP group. The failover feature of IPMP preserves connectivity and prevents disruption of any existing

connections. Additionally, IPMP improves overall network performance by automatically spreading out network traffic across the set of interfaces in the IPMP group. This process is called *load spreading*.

# Solaris IPMP Components

Solaris IPMP involves the following software:

- The `in.mpathd` daemon, which is explained fully in the `in.mpathd`(1M) man page.
- The `/etc/default/mpathd` configuration file, which is also described in the `in.mpathd`(1M) man page.
- `ifconfig` options for IPMP configuration, as described in the `ifconfig`(1M) man page.

## Multipathing Daemon, `in.mpathd`

The `in.mpathd` daemon detects interface failures, and then implements various procedures for failover and failback. After `in.mpathd` detects a failure or a repair, the daemon sends an `ioctl` to perform the failover or failback. The `ip` kernel module, which implements the `ioctl`, does the network access failover transparently and automatically.

---

**Note –** Do not use Alternate Pathing while using IPMP on the same set of network interface cards. Likewise, you should not use IPMP while you are using Alternate Pathing. You can use Alternate Pathing and IPMP at the same time on different sets of interfaces. For more information about Alternate Pathing, refer to the *Sun Enterprise Server Alternate Pathing 2.3.1 User Guide*.

---

The `in.mpathd` daemon detects failures and repairs by sending out probes on all the interfaces that are part of an IPMP group. The `in.mpathd` daemon also detects failures and repairs by monitoring the `RUNNING` flag on each interface in the group. Refer to the `in.mpathd`(1M) man page for more information.

---

**Note –** DHCP is not supported to manage IPMP data addresses. If you attempt to use DHCP on these addresses, DHCP eventually abandons control of these addresses. Do not use DHCP on data addresses.

---

# IPMP Terminology and Concepts

This section introduces terms and concepts that are used throughout the IPMP chapters in this book.

## IP Link

In IPMP terminology, an *IP link* is a communication facility or medium over which nodes can communicate at the data-link layer of the Internet protocol suite. Types of IP links might include simple Ethernets, bridged Ethernets, hubs, or Asynchronous Transfer Mode (ATM) networks. An IP link can have one or more IPv4 subnet numbers, and, if applicable, one or more IPv6 subnet prefixes. A subnet number or prefix cannot be assigned to more than one IP link. In ATM LANE, an IP link is a single emulated local area network (LAN). With the Address Resolution Protocol (ARP), the scope of the ARP protocol is a single IP link.

---

**Note –** Other IP-related documents, such as RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification*, use the term *link* instead of *IP link*. Part VI uses the term *IP link* to avoid confusion with IEEE 802. In IEEE 802, *link* refers to a single wire from an Ethernet network interface card (NIC) to an Ethernet switch.

---

## Physical Interface

The *physical interface* provides a system's attachment to an IP link. This attachment is often implemented as a device driver and a NIC. If a system has multiple interfaces attached to the same link, you can configure IPMP to perform failover if one of the interfaces fails. For more information on physical interfaces, refer to "IPMP Interface Configurations" on page 724.

## Network Interface Card

A *network interface card* is a network adapter that can be built in to the system. Or, the NIC can be a separate card that serves as an interface from the system to an IP link. Some NICs can have multiple physical interfaces. For example, a `qfe` NIC can have four interfaces, `qfe0` through `qfe3`, and so on.

## IPMP Group

An IP multipathing group, or *IPMP* group, consists of one or more physical interfaces on the same system that are configured with the same IPMP group name. All interfaces in the IPMP group must be connected to the same IP link. The same (non-null) character string IPMP group name identifies all interfaces in the group. You can place interfaces from NICs of different speeds within the same IPMP group, as long as the NICs are of the same type. For example, you can configure the interfaces of 100-megabit Ethernet NICs and the interfaces of one gigabit Ethernet NICs in the same group. As another example, suppose you have two 100-megabit Ethernet NICs. You can configure one of the interfaces down to 10 megabits and still place the two interfaces into the same IPMP group.

You cannot place two interfaces of different media types into an IPMP group. For example, you cannot place an ATM interface in the same group as an Ethernet interface.

## Failure Detection and Failover

*Failure detection* is the process of detecting when an interface or the path from an interface to an Internet layer device no longer works. IPMP provides systems with the ability to detect when an interface has failed. IPMP detects the following types of communication failures:

- The transmit or receive path of the interface has failed.
- The attachment of the interface to the IP link is down.
- The port on the switch does not transmit or receive packets.
- The physical interface in an IPMP group is not present at system boot.

After detecting a failure, IPMP begins failover. *Failover* is the automatic process of switching the network access from a failed interface to a functioning physical interface in the same group. Network access includes IPv4 unicast, multicast, and broadcast traffic, as well as IPv6 unicast and multicast traffic. Failover can only occur when you have configured more than one interface in the IPMP group. The failover process ensures uninterrupted access to the network.

## Repair Detection and Failback

*Repair detection* is the process of detecting when a NIC or the path from a NIC to an Internet layer device starts operating correctly after a failure. After detecting that a NIC has been repaired, IPMP performs *failback*, the process of switching network access back to the repaired interface. Repair detection assumes that you have enabled failbacks. See "Detecting Physical Interface Repairs" on page 727 for more information.

## Target Systems

Probe-based failure detection uses *target systems* to determine the condition of an interface. Each target system must be attached to the same IP link as the members of the IPMP group. The `in.mpathd` daemon on the local system sends ICMP probe messages to each target system. The probe messages help to determine the health of each interface in the IPMP group.

For more information about target system use in probe-based failure detection, refer to "Probe-Based Failure Detection" on page 726.

## Outbound Load Spreading

With IPMP configured, outbound network packets are spread across multiple NICs without affecting the ordering of packets. This process is known as *load spreading*. As a result of load spreading, higher throughput is achieved. Load spreading occurs only when the network traffic is flowing to multiple destinations that use multiple connections.

## Dynamic Reconfiguration

*Dynamic reconfiguration* (DR) is the ability to reconfigure a system while the system is running, with little or no impact on existing operations. Not all Sun platforms support DR. Some Sun

platforms might only support DR of certain types of hardware. On platforms that support DR of NIC's, IPMP can be used to transparently fail over network access, providing uninterrupted network access to the system.

For more information on how IPMP supports DR, refer to "IPMP and Dynamic Reconfiguration" on page 729.

# Basic Requirements of IPMP

IPMP is built into the Solaris Operating System (Solaris OS) and does not require any special hardware. Any interface that is supported by the Solaris OS can be used with IPMP. However, IPMP does impose the following requirements on your network configuration and topology:

- All interfaces in an IPMP group must have unique MAC addresses.

  Note that by default, the network interfaces on SPARC based systems all share a single MAC address. Thus, you must explicitly change the default in order to use IPMP on SPARC based systems. For more information, refer to "How to Plan for an IPMP Group" on page 735.
- All interfaces in an IPMP group must be of the same media type. For more information, refer to "IPMP Group" on page 719.
- All interfaces in an IPMP group must be on the same IP link. For more information, refer to "IPMP Group" on page 719.
- Depending on your failure detection requirements, you might need to either use specific types of network interfaces or configure additional IP addresses on each network interface. Refer to "Link-Based Failure Detection" on page 725 and "Probe-Based Failure Detection" on page 726.

# IPMP Addressing

You can configure IPMP failure detection on both IPv4 networks and dual-stack, IPv4 and IPv6 networks. Interfaces that are configured with IPMP support two types of addresses: data addresses and test addresses.

## Data Addresses

*Data addresses* are the conventional IPv4 and IPv6 addresses that are assigned to an interface of a NIC at boot time or manually, through the `ifconfig` command. The standard IPv4 and, if applicable, IPv6 packet traffic through an interface is considered to be *data traffic*.

# Test Addresses

*Test addresses* are IPMP-specific addresses that are used by the `in.mpathd` daemon. For an interface to use probe-based failure and repair detection, that interface must be configured with at least one test address.

---

**Note –** You need to configure test addresses only if you want to use probe-based failure detection.

---

The `in.mpathd` daemon uses test addresses to exchange ICMP probes, also called *probe traffic*, with other targets on the IP link. Probe traffic helps to determine the status of the interface and its NIC, including whether an interface has failed. The probes verify that the send and receive path to the interface is working correctly.

Each interface can be configured with an IP test address. For an interface on a dual-stack network, you can configure an IPv4 test address, an IPv6 test address, or both IPv4 and IPv6 test addresses.

After an interface fails, the test addresses remain on the failed interface so that `in.mpathd` can continue to send probes to check for subsequent repair. You must specifically configure test addresses so that applications do not accidentally use them. For more information, refer to "Preventing Applications From Using Test Addresses" on page 723.

For more information on probe-based failure detection, refer to "Probe-Based Failure Detection" on page 726.

## IPv4 Test Addresses

In general, you can use any IPv4 address on your subnet as a test address. IPv4 test addresses do not need to be routeable. Because IPv4 addresses are a limited resource for many sites, you might want to use non-routeable RFC 1918 private addresses as test addresses. Note that the `in.mpathd` daemon exchanges only ICMP probes with other hosts on the same subnet as the test address. If you do use RFC 1918-style test addresses, be sure to configure other systems, preferably routers, on the IP link with addresses on the appropriate RFC 1918 subnet. The `in.mpathd` daemon can then successfully exchange probes with target systems.

The IPMP examples use RFC 1918 addresses from the `192.168.0/24` network as IPv4 test addresses. For more information about RFC 1918 private addresses, refer to RFC 1918, Address Allocation for Private Internets. (`http://www.ietf.org/rfc/rfc1918.txt?number=1918`)

To configure IPv4 test addresses, refer to the task "How to Configure an IPMP Group With Multiple Interfaces" on page 737.

### IPv6 Test Addresses

The only valid IPv6 test address is the link-local address of a physical interface. You do not need a separate IPv6 address to serve as an IPMP test address. The IPv6 link-local address is based on the Media Access Control (MAC ) address of the interface. Link-local addresses are automatically configured when the interface becomes IPv6-enabled at boot time or when the interface is manually configured through `ifconfig`.

To identify the link-local address of an interface, run the `ifconfig` *interface* command on an IPv6-enabled node. Check the output for the address that begins with the prefix `fe80`, the link-local prefix. The `NOFAILOVER` flag in the following `ifconfig` output indicates that the link-local address `fe80::a00:20ff:feb9:17fa/10` of the `hme0` interface is used as the test address.

```
hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
            inet6 fe80::a00:20ff:feb9:17fa/10
```

For more information on link-local addresses, refer to "Link-Local Unicast Address" on page 78.

When an IPMP group has both IPv4 and IPv6 plumbed on all the group's interfaces, you do not need to configure separate IPv4 test addresses. The `in.mpathd` daemon can use the IPv6 link-local addresses as test addresses.

To create an IPv6 test address, refer to the task "How to Configure an IPMP Group With Multiple Interfaces" on page 737.

## Preventing Applications From Using Test Addresses

After you have configured a test address, you need to ensure that this address is not used by applications. Otherwise, if the interface fails, the application is no longer reachable because test addresses do not fail over during the failover operation. To ensure that IP does not choose the test address for normal applications, mark the test address as `deprecated`.

IPv4 does not use a deprecated address as a source address for any communication, unless an application explicitly binds to the address. The `in.mpathd` daemon explicitly binds to such an address in order to send and receive probe traffic.

Because IPv6 link-local addresses are usually not present in a name service, DNS and NIS applications do not use link-local addresses for communication. Consequently, you must not mark IPv6 link-local addresses as `deprecated`.

IPv4 test addresses should not be placed in the DNS and NIS name service tables. In IPv6, link-local addresses are not normally placed in the name service tables.

# IPMP Interface Configurations

An IPMP configuration typically consists of two or more physical interfaces on the same system that are attached to the same IP link. These physical interfaces might or might not be on the same NIC. The interfaces are configured as members of the same IPMP group. If the system has additional interfaces on a second IP link, you must configure these interfaces as another IPMP group.

A single interface can be configured in its own IPMP group. The single interface IPMP group has the same behavior as an IPMP group with multiple interfaces. However, failover and failback cannot occur for an IPMP group with only one interface.

## Standby Interfaces in an IPMP Group

The *standby interface* in an IPMP group is not used for data traffic unless some other interface in the group fails. When a failure occurs, the data addresses on the failed interface migrate to the standby interface. Then, the standby interface is treated the same as other active interfaces until the failed interface is repaired. Some failovers might not choose a standby interface. Instead, these failovers might choose an active interface with fewer data addresses that are configured as UP than the standby interface.

You should configure only test addresses on a standby interface. IPMP does not permit you to add a data address to an interface that is configured through the `ifconfig` command as `standby`. Any attempt to create this type of configuration will fail. Similarly, if you configure as `standby` an interface that already has data addresses, these addresses automatically fail over to another interface in the IPMP group. Due to these restrictions, you must use the `ifconfig` command to mark any test addresses as `deprecated` and `-failover` prior to setting the interface as `standby`. To configure standby interfaces, refer to "How to Configure a Standby Interface for an IPMP Group" on page 743.

## Common IPMP Interface Configurations

As mentioned in "IPMP Addressing" on page 721, interfaces in an IPMP group handle regular data traffic and probe traffic, depending on the interfaces' configuration. You use IPMP options of the `ifconfig` command to create the configuration.

An *active interface* is a physical interface that transmits both data traffic and probe traffic. You configure the interface as "active" by performing either the task "How to Configure an IPMP Group With Multiple Interfaces" on page 737 or the task "How to Configure a Single Interface IPMP Group" on page 745.

The following are two common types of IPMP configurations:

| | |
|---|---|
| **Active-active configuration** | A two interface IPMP group where both interfaces are "active," that is they might be transmitting both probe and data traffic at all times. |
| **Active-standby configuration** | A two interface IPMP group where one interface is configured as "standby." |

### Checking the Status of an Interface

You can check the status of an interface by issuing the `ifconfig` *interface* command. For general information on `ifconfig` status reporting, refer to "How to Get Information About a Specific Interface" on page 205.

For example, you can use the `ifconfig` command to obtain the status of a standby interface. When the standby interface is not hosting any data address, the interface has the `INACTIVE` flag for its status. You can observe this flag in the status lines for the interface in the `ifconfig` output.

# IPMP Failure Detection and Recovery Features

The `in.mpathd` daemon handles the following types of failure detection:

- Link-based failure detection, if supported by the NIC driver
- Probe-based failure detection, when test addresses are configured
- Detection of interfaces that were missing at boot time

The `in.mpathd`(1M) man page completely describes how the `in.mpathd` daemon handles the detection of interface failures.

## Link-Based Failure Detection

Link-based failure detection is always enabled, provided that the interface supports this type of failure detection. The following Sun network drivers are supported in the current release of the Solaris OS:

- `hme`
- `eri`
- `ce`
- `ge`
- `bge`
- `qfe`
- `dmfe`
- `e1000g`

- ixgb
- nge
- nxge
- rge
- xge

To determine whether a third-party interface supports link-based failure detection, refer to the manufacturer's documentation.

These network interface drivers monitor the interface's link state and notify the networking subsystem when that link state changes. When notified of a change, the networking subsystem either sets or clears the `RUNNING` flag for that interface, as appropriate. When the daemon detects that the interface's `RUNNING` flag has been cleared, the daemon immediately fails the interface.

## Probe-Based Failure Detection

The `in.mpathd` daemon performs probe-based failure detection on each interface in the IPMP group that has a test address. Probe-based failure detection involves the sending and receiving of ICMP probe messages that use test addresses. These messages go out over the interface to one or more target systems on the same IP link. For an introduction to test addresses, refer to "Test Addresses" on page 722. For information on configuring test addresses, refer to "How to Configure an IPMP Group With Multiple Interfaces" on page 737.

The `in.mpathd` daemon determines which target systems to probe dynamically. Routers that are connected to the IP link are automatically selected as targets for probing. If no routers exist on the link, `in.mpathd` sends probes to neighbor hosts on the link. A multicast packet that is sent to the all hosts multicast address, `224.0.0.1` in IPv4 and `ff02::1` in IPv6, determines which hosts to use as target systems. The first few hosts that respond to the echo packets are chosen as targets for probing. If `in.mpathd` cannot find routers or hosts that responded to the ICMP echo packets, `in.mpathd` cannot detect probe-based failures.

You can use host routes to explicitly configure a list of target systems to be used by `in.mpathd`. For instructions, refer to "Configuring Target Systems" on page 741.

To ensure that each interface in the IPMP group functions properly, `in.mpathd` probes all the targets separately through all the interfaces in the IPMP group. If no replies are made in response to five consecutive probes, `in.mpathd` considers the interface to have failed. The probing rate depends on the *failure detection time* (FDT). The default value for failure detection time is 10 seconds. However, you can tune the failure detection time in the `/etc/default/mpathd` file. For instructions, go to "How to Configure the `/etc/default/mpathd` File" on page 754.

For a repair detection time of 10 seconds, the probing rate is approximately one probe every two seconds. The minimum repair detection time is twice the failure detection time , 20 seconds by

default, because replies to 10 consecutive probes must be received. The failure and repair detection times apply only to probe-based failure detection.

## Group Failures

A *group failure* occurs when all interfaces in an IPMP group appear to fail at the same time. The `in.mpathd` daemon does not perform failovers for a group failure. Also, no failover occurs when all the target systems fail at the same time. In this instance, `in.mpathd` flushes all of its current target systems and discovers new target systems.

## Detecting Physical Interface Repairs

For the `in.mpathd` daemon to consider an interface to be repaired, the `RUNNING` flag must be set for the interface. If probe-based failure detection is used, the `in.mpathd` daemon must receive responses to 10 consecutive probe packets from the interface before that interface is considered repaired. When an interface is considered repaired, any addresses that failed over to another interface then fail back to the repaired interface. If the interface was configured as "active" before it failed, after repair that interface can resume sending and receiving traffic.

## What Happens During Interface Failover

The following two examples show a typical configuration and how that configuration automatically changes when an interface fails. When the `hme0` interface fails, notice that all data addresses move from `hme0` to `hme1`.

**EXAMPLE 30–1** Interface Configuration Before an Interface Failure

```
hme0: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
     inet 192.168.85.19 netmask ffffff00 broadcast 192.168.85.255
     groupname test
hme0:1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500
     index 2 inet 192.168.85.21 netmask ffffff00 broadcast 192.168.85.255
hme1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
8     inet 192.168.85.20 netmask ffffff00 broadcast 192.168.85.255
     groupname test
hme1:1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500
     index 2 inet 192.168.85.22 netmask ffffff00 broadcast 192.168.85.255
hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
     inet6 fe80::a00:20ff:feb9:19fa/10
     groupname test
hme1: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
     inet6 fe80::a00:20ff:feb9:1bfc/10
```

**EXAMPLE 30–1** Interface Configuration Before an Interface Failure *(Continued)*

```
     groupname test
```

**EXAMPLE 30–2** Interface Configuration After an Interface Failure

```
hme0: flags=19000842<BROADCAST,RUNNING,MULTICAST,IPv4,NOFAILOVER,FAILED> mtu 0 index 2
        inet 0.0.0.0 netmask 0
        groupname test
hme0:1: flags=19040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER,FAILED>
        mtu 1500 index 2 inet 192.168.85.21 netmask ffffff00 broadcast 10.0.0.255
hme1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.85.20 netmask ffffff00 broadcast 192.168.85.255
        groupname test
hme1:1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER> mtu 1500
        index 2 inet 192.168.85.22 netmask ffffff00 broadcast 10.0.0.255
hme1:2: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 6
        inet 192.168.85.19 netmask ffffff00 broadcast 192.168.18.255
hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER,FAILED> mtu 1500 index 2
        inet6 fe80::a00:20ff:feb9:19fa/10
        groupname test
hme1: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
        inet6 fe80::a00:20ff:feb9:1bfc/10
        groupname test
```

You can see that the `FAILED` flag is set on `hme0` to indicate that this interface has failed. You can also see that `hme1:2` has been created. `hme1:2` was originally `hme0`. The address `192.168.85.19` then becomes accessible through `hme1`.

Multicast memberships that are associated with `192.168.85.19` can still receive packets, but they now receive packets through `hme1`. When the failover of address `192.168.85.19` from `hme0` to `hme1` occurred, a dummy address `0.0.0.0` was created on `hme0`. The dummy address was created so that `hme0` can still be accessed. `hme0:1` cannot exist without `hme0`. The dummy address is removed when a subsequent failback takes place.

Similarly, failover of the IPv6 address from `hme0` to `hme1` occurred. In IPv6, multicast memberships are associated with interface indexes. Multicast memberships also fail over from `hme0` to `hme1`. All the addresses that `in.ndpd` configured also moved. This action is not shown in the examples.

The `in.mpathd` daemon continues to probe through the failed interface `hme0`. After the daemon receives 10 consecutive replies for a default repair detection time of 20 seconds, the daemon determines that the interface is repaired. Because the `RUNNING` flag is also set on `hme0`, the daemon invokes the failback. After failback, the original configuration is restored.

For a description of all error messages that are logged on the console during failures and repairs, see the `in.mpathd`(1M) man page.

# IPMP and Dynamic Reconfiguration

The dynamic reconfiguration (DR) feature enables you to reconfigure system hardware, such as interfaces, while the system is running. This section explains how DR interoperates with IPMP.

On a system that supports DR of NICs, IPMP can be used to preserve connectivity and prevent disruption of existing connections. You can safely attach, detach, or reattach NIC's on a system that supports DR and uses IPMP. This is possible because IPMP is integrated into the Reconfiguration Coordination Manager (RCM) framework. *RCM* manages the dynamic reconfiguration of system components.

You typically use the `cfgadm` command to perform DR operations. However, some platforms provide other methods. Consult your platform's documentation for details. You can find specific documentation about DR from the following resources.

**TABLE 30–1** Documentation Resources for Dynamic Reconfiguration

| Description | For Information |
|---|---|
| Detailed information on the `cfgadm` command | `cfgadm`(1M) man page |
| Specific information about DR in the Sun Cluster environment | *Sun Cluster 3.1 System Administration Guide* |
| Specific information about DR in the Sun Fire environment | *Sun Fire 880 Dynamic Reconfiguration Guide* |
| Introductory information about DR and the `cfgadm` command | Chapter 6, "Dynamically Configuring Devices (Tasks)," in *System Administration Guide: Devices and File Systems* |
| Tasks for administering IPMP groups on a system that supports DR | "Replacing a Failed Physical Interface on Systems That Support Dynamic Reconfiguration" on page 750 |

## Attaching NICs

You can add interfaces to an IPMP group at any time by using the `ifconfig` command, as explained in "How to Configure an IPMP Group With Multiple Interfaces" on page 737. Thus, any interfaces on system components that you attach after system boot can be plumbed and added to an existing IPMP group. Or, if appropriate, you can configure the newly added interfaces into their own IPMP group.

These interfaces and the data addresses that are configured on them are immediately available for use by the IPMP group. However, for the system to automatically configure and use the interfaces after a reboot, you must create an `/etc/hostname.`*interface* file for each new interface. For instructions, refer to"How to Configure a Physical Interface After System Installation" on page 150.

If an `/etc/hostname.`*interface* file already exists when the interface is attached, then RCM automatically configures the interface according to the contents of this file. Thus, the interface receives the same configuration that it would have received after system boot.

## Detaching NICs

All requests to detach system components that contain NICs are first checked to ensure that connectivity can be preserved. For instance, by default you cannot detach a NIC that is not in an IPMP group. You also cannot detach a NIC that contains the only functioning interfaces in an IPMP group. However, if you must remove the system component, you can override this behavior by using the `-f` option of `cfgadm`, as explained in the `cfgadm`(1M) man page.

If the checks are successful, the data addresses associated with the detached NIC fail over to a functioning NIC in the same group, as if the NIC being detached had failed. When the NIC is detached, all test addresses on the NIC's interfaces are unconfigured. Then, the NIC is unplumbed from the system. If any of these steps fail, or if the DR of other hardware on the same system component fails, then the previous configuration is restored to its original state. You should receive a status message regarding this event. Otherwise, the detach request completes successfully. You can remove the component from the system. No existing connections are disrupted.

## Reattaching NICs

RCM records the configuration information associated with any NIC's that are detached from a running system. As a result, RCM treats the reattachment of a NIC that had been previously detached identically as it would to the attachment of a new NIC. That is, RCM only performs plumbing.

However, reattached NICs typically have an existing `/etc/hostname.`*interface* file. In this case, RCM automatically configures the interface according to the contents of the existing `/etc/hostname.`*interface* file. Additionally, RCM informs the `in.mpathd` daemon of each data address that was originally hosted on the reattached interface. Thus, once the reattached interface is functioning properly, all of its data addresses are failed back to the reattached interface as if it had been repaired.

If the NIC being reattached does not have an `/etc/hostname.`*interface* file, then no configuration information is available. RCM has no information regarding how to configure the interface. One consequence of this situation is that addresses that were previously failed over to another interface are not failed back.

## NICs That Were Missing at System Boot

NICs that are not present at system boot represent a special instance of failure detection. At boot time, the startup scripts track any interfaces with `/etc/hostname.`*interface* files that cannot be plumbed. Any data addresses in such an interface's `/etc/hostname.`*interface* file are automatically hosted on an alternative interface in the IPMP group.

In such an event, you receive error messages similar to the following

```
moving addresses from failed IPv4 interfaces: hme0 (moved to hme1)
moving addresses from failed IPv6 interfaces: hme0 (moved to hme1)
```

If no alternative interface exists, you receive error messages similar to the following:

```
moving addresses from failed IPv4 interfaces: hme0 (couldn't move;
   no alternative interface)
 moving addresses from failed IPv6 interfaces: hme0 (couldn't move;
   no alternative interface)
```

---

**Note –** In this instance of failure detection, only data addresses that are explicitly specified in the missing interface's `/etc/hostname.`*interface* file move to an alternative interface. Any addresses that are usually acquired through other means, such as through RARP or DHCP, are not acquired or moved.

---

If an interface with the same name as another interface that was missing at system boot is reattached using DR, RCM automatically plumbs the interface. Then, RCM configures the interface according to the contents of the interface's `/etc/hostname.`*interface* file. Finally, RCM fails back any data addresses, just as if the interface had been repaired. Thus, the final network configuration is identical to the configuration that would have been made if the system had been booted with the interface present.

◆ ◆ ◆ CHAPTER 31

# Administering IPMP (Tasks)

This chapter provides tasks for administering interface groups with IP network multipathing (IPMP). The following major topics are discussed:


- "Configuring IPMP (Task Maps)" on page 733
- "Configuring IPMP Groups" on page 735
- "Maintaining IPMP Groups" on page 746
- "Replacing a Failed Physical Interface on Systems That Support Dynamic Reconfiguration" on page 750
- "Recovering a Physical Interface That Was Not Present at System Boot" on page 752
- "Modifying IPMP Configurations" on page 754


For an overview of IPMP concepts, refer to Chapter 30, "Introducing IPMP (Overview)."

## Configuring IPMP (Task Maps)

This section contains links to the tasks that are described in this chapter.

### Configuring and Administering IPMP Groups (Task Map)

| Task | Description | For Instructions |
|---|---|---|
| Plan for an IPMP group. | Lists all ancillary information and required tasks before you can configure an IPMP group. | "How to Plan for an IPMP Group" on page 735 |

| Task | Description | For Instructions |
|---|---|---|
| Configure an IPMP interface group with multiple interfaces. | Configures multiple interfaces as members of an IPMP group. | "How to Configure an IPMP Group With Multiple Interfaces" on page 737 |
| Configure an IPMP group where one of the interfaces is a standby interface. | Configures one of the interfaces in a multiple interface IPMP group as a standby interface. | "How to Configure a Standby Interface for an IPMP Group" on page 743 |
| Configure an IPMP group that consists of a single interface. | Creates a single interface IPMP group. | "How to Configure a Single Interface IPMP Group" on page 745 |
| Display the IPMP group to which a physical interface belongs. | Explains how to obtain the name of an interface's IPMP group from the output of the `ifconfig` command. | "How to Display the IPMP Group Membership of an Interface" on page 747 |
| Add an interface to an IPMP group. | Configures a new interface as a member of an existing IPMP group. | "How to Add an Interface to an IPMP Group" on page 747 |
| Remove an interface from an IPMP group. | Explains how to remove an interface from an IPMP group. | "How to Remove an Interface From an IPMP Group" on page 748 |
| Move an interface from an existing IPMP group to a different group. | Moves interfaces among IPMP groups. | "How to Move an Interface From One IPMP Group to Another Group" on page 749 |
| Change three default settings for the `in.mpathd` daemon. | Customizes failure detection time and other parameters of the `in.mpathd` daemon. | "How to Configure the `/etc/default/mpathd` File" on page 754 |

# Administering IPMP on Interfaces That Support Dynamic Reconfiguration (Task Map)

| Task | Description | For Instructions |
|---|---|---|
| Remove an interface that has failed. | Removes a failed interface on a system. | "How to Remove a Physical Interface That Has Failed (DR-Detach)" on page 750 |
| Replace an interface that has failed. | Replaces a failed interface. | "How to Replace a Physical Interface That Has Failed (DR-Attach)" on page 751 |
| Recover an interface that was not configured at boot time. | Recovers a failed interface. | "How to Recover a Physical Interface That Was Not Present at System Boot" on page 752 |

# Configuring IPMP Groups

This section provides procedures for configuring IPMP groups. It also describes how to configure an interface as a standby.

## Planning for an IPMP Group

Before you configure interfaces on a system as part of an IPMP group, you need to do some preconfiguration planning.

### ▼ How to Plan for an IPMP Group

The following procedure includes the planning tasks and information to be gathered prior to configuring the IPMP group. The tasks do not have to be performed in sequence.

**1 Decide which interfaces on the system are to be part of the IPMP group.**

An IPMP group usually consists of at least two physical interfaces that are connected to the same IP link. However, you can configure a single interface IPMP group, if required. For an introduction to IPMP groups, refer to "IPMP Interface Configurations" on page 724. For example, you can configure the same Ethernet switch or the same IP subnet under the same IPMP group. You can configure any number of interfaces into the same IPMP group.

You cannot use the `group` parameter of the `ifconfig` command with logical interfaces. For example, you can use the `group` parameter with `hme0`, but not with `hme0:1`.

**2 Verify that each interface in the group has a unique MAC address.**

For instructions, refer to "SPARC: How to Ensure That the MAC Address of an Interface Is Unique" on page 154.

**3 Choose a name for the IPMP group.**

Any non-null name is appropriate for the group. You might want to use a name that identifies the IP link to which the interfaces are attached.

**4 Ensure that the same set of STREAMS modules is pushed and configured on all interfaces in the IPMP group.**

All interfaces in the same group must have the same STREAMS modules configured in the same order.

**a. Check the order of STREAMS modules on all interfaces in the prospective IPMP group.**

You can print out a list of STREAMS modules by using the `ifconfig` *interface* `modlist` command. For example, here is the `ifconfig` output for an `hme0` interface:

```
# ifconfig hme0 modlist
    0 arp
    1 ip
    2 hme
```

Interfaces normally exist as network drivers directly below the IP module, as shown in the output from `ifconfig hme0 modlist`. They should not require additional configuration.

However, certain technologies, such as NCA or IP Filter, insert themselves as STREAMS modules between the IP module and the network driver. Problems can result in the way interfaces of the same IPMP group behave.

If a STREAMS module is stateful, then unexpected behavior can occur on failover, even if you push the same module onto all of the interfaces in a group. However, you can use stateless STREAMS modules, provided that you push them in the same order on all interfaces in the IPMP group.

**b. Push the modules of an interface in the standard order for the IPMP group.**

**`ifconfig`** *interface* **`modinsert`** *module-name*

```
ifconfig hme0 modinsert ip
```

**5 Use the same IP addressing format on all interfaces of the IPMP group.**

If one interface is configured for IPv4, then all interfaces of the group must be configured for IPv4. Suppose you have an IPMP group that is composed of interfaces from several NICs. If you add IPv6 addressing to the interfaces of one NIC, then all interfaces in the IPMP group must be configured for IPv6 support.

**6 Check that all interfaces in the IPMP group are connected to the same IP link.**

**7 Verify that the IPMP group does not contain interfaces with different network media types.**

The interfaces that are grouped together should be of the same interface type, as defined in `/usr/include/net/if_types.h`. For example, you cannot combine Ethernet and Token ring interfaces in an IPMP group. As another example, you cannot combine a Token bus interface with asynchronous transfer mode (ATM) interfaces in the same IPMP group.

8 **For IPMP with ATM interfaces, configure the ATM interfaces in LAN emulation mode.**

IPMP is not supported for interfaces using Classical IP over ATM.

# Configuring IPMP Groups

This section contains configuration tasks for a typical IPMP group with at least two physical interfaces.

- For an introduction to multiple interface IPMP groups, refer to "IPMP Group" on page 719.
- For planning tasks, refer to "Planning for an IPMP Group" on page 735.
- To configure an IPMP group with only one physical interface, refer to "Configuring IPMP Groups With a Single Physical Interface" on page 745.

## ▼ How to Configure an IPMP Group With Multiple Interfaces

**Before You Begin** You need to have already configured the IPv4 addresses, and, if appropriate, the IPv6 addresses of all interfaces in the prospective IPMP group.

1 **On the system with the interfaces to be configured, assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Place each physical interface into an IPMP group.**

```
# ifconfig interface group group-name
```

For example, to place `hme0` and `hme1` under group `testgroup1`, you would type the following commands:

```
# ifconfig hme0 group testgroup1
# ifconfig hme1 group testgroup1
```

Avoid using spaces in group names. The `ifconfig` status display does not show spaces. Consequently, do not create two similar group names where the only difference is that one name also contains a space. If one of the group names contains a space, these group names look the same in the status display.

In a dual-stack environment, placing the IPv4 instance of an interface under a particular group automatically places the IPv6 instance under the same group.

3 **(Optional) Configure an IPv4 test address on one or more physical interfaces.**

You need to configure a test address only if you want to use probe-based failure detection on a particular interface. Test addresses are configured as logical interfaces of the physical interface that you specify to the `ifconfig` command.

If one interface in the group is to become the standby interface, do not configure a test address for that interface at this time. You configure a test address for the standby interface as part of the task "How to Configure a Standby Interface for an IPMP Group" on page 743.

Use the following syntax of the `ifconfig` command for configuring a test address:

```
# ifconfig interface addif ip-address <parameters> -failover deprecated up
```

For example, you would create the following test address for the primary network interface `hme0`:

```
# ifconfig hme0 addif 192.168.85.21 netmask + broadcast + -failover deprecated up
```

This command sets the following parameters for the primary network interface `hme0`:

- Address set to `192.168.85.21`
- Netmask and broadcast address set to the default value
- `-failover` and `deprecated` options set

---

**Note –** You must mark an IPv4 test address as `deprecated` to prevent applications from using the test address.

---

**4 Check the IPv4 configuration for a specific interface.**

You can always view the current status of an interface by typing `ifconfig` *interface*. For more information on viewing an interface's status, refer to "How to Get Information About a Specific Interface" on page 205.

You can get information about test address configuration for a physical interface by specifying the logical interface that is assigned to the test address.

```
# ifconfig hme0:1
    hme0:1: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
    mtu 1500 index 2
    inet 192.168.85.21 netmask ffffff00 broadcast 192.168.85.255
```

**5 (Optional) If applicable, configure an IPv6 test address.**

```
# ifconfig interface inet6 -failover
```

Physical interfaces with IPv6 addresses are placed into the same IPMP group as the interfaces' IPv4 addresses. This happens when you configure the physical interface with IPv4 addresses into an IPMP group. If you first place physical interfaces with IPv6 addresses into an IPMP group, physical interfaces with IPv4 addresses are also implicitly placed in the same IPMP group.

For example, to configure `hme0` with an IPv6 test address, you would type the following:

```
# ifconfig hme0 inet6 -failover
```

You do not need to mark an IPv6 test address as deprecated to prevent applications from using the test address.

**6 Check the IPv6 configuration.**

```
# ifconfig hme0 inet6
    hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6,NOFAILOVER> mtu 1500 index 2
            inet6 fe80::a00:20ff:feb9:17fa/10
            groupname test
```

The IPv6 test address is the link-local address of the interface.

**7 (Optional) Preserve the IPMP group configuration across reboots.**

- For IPv4, add the following line to the `/etc/hostname.`*interface* file:

  *interface-address* `<parameters> group` *group-name* `up \`
  `addif` *logical-interface* `-failover deprecated <parameters> up`

  In this instance, the test IPv4 address is configured only on the next reboot. If you want the configuration to be invoked in the current session, do steps 1, 2, and, optionally 3.

- For IPv6, add the following line to the `/etc/hostname6.`*interface* file:

  `-failover group` *group-name* `up`

  This test IPv6 address is configured only on the next reboot. If you want the configuration to be invoked in the current session, do steps 1, 2, and, optionally, 5.

**8 (Optional) Add more interfaces to the IPMP group by repeating steps 1 through 6.**

You can add new interfaces to an existing group on a live system. However, changes are lost across reboots.

**Example 31–1** Configuring an IPMP Group With Two Interfaces

Suppose you want to do the following:

- Have the netmask and broadcast address set to the default value.
- Configure the interface with a test address `192.168.85.21`.

You would type the following command:

```
# ifconfig hme0 addif 192.168.85.21 netmask + broadcast + -failover deprecated up
```

You must mark an IPv4 test address as `deprecated` to prevent applications from using the test address. See "How to Configure an IPMP Group With Multiple Interfaces" on page 737.

To turn on the failover attribute of the address, you would use the `failover` option without the dash

All test IP addresses in an IPMP group must use the same network prefix. The test IP addresses must belong to a single IP subnet.

**Example 31–2** Preserving an IPv4 IPMP Group Configuration Across Reboots

Suppose you want to create an IPMP group called `testgroup1` with the following configuration:

- Physical interface `hme0` with the data address `192.168.85.19`
- A logical interface with the test address `192.168.85.21`

---

**Note –** In this example, physical interface and data address are paired together. Likewise for logical interface and test address. However, no inherent relationships exist between an interface "type" and the address type.

---

- `deprecated` and `-failover` options set
- Netmask and broadcast address set to the default value

You would add the following line to the `/etc/hostname.hme0` file:

```
192.168.85.19 netmask + broadcast + group testgroup1 up \
    addif 192.168.85.21 deprecated -failover netmask + broadcast + up
```

Similarly, to place the second interface `hme1` under the same group `testgroup1` and to configure a test address, you would add the following line:

```
192.168.85.20 netmask + broadcast + group testgroup1 up \
    addif 192.168.85.22 deprecated -failover netmask + broadcast + up
```

**Example 31–3** Preserving an IPv6 IPMP Group Configuration Across Reboots

To create a test group for interface `hme0` with an IPv6 address, you would add the following line to the `/etc/hostname6.hme0` file:

```
-failover group testgroup1 up
```

Similarly, to place the second interface `hme1` in group `testgroup1` and to configure a test address, you would add the following line to the `/etc/hostname6.hme1` file:

```
-failover group testgroup1 up
```

**Troubleshooting** During IPMP group configuration, `in.mpathd` outputs a number of messages to the system console or to the `syslog` file. These messages are informational in nature and indicate that the IPMP configuration functions correctly.

- This message indicates that interface `hme0` was added to IPMP group `testgroup1`. However, `hme0` does not have a test address configured. To enable probe-based failure detection, you need to assign a test address to the interface.

```
May 24 14:09:57 host1 in.mpathd[101180]: No test address configured on interface hme0;
disabling probe-based failure detection on it.
testgroup1
```

- This message appears for all interfaces with only IPv4 addresses that are added to an IPMP group.

  ```
  May 24 14:10:42 host4 in.mpathd[101180]: NIC qfe0 of group testgroup1 is not
  plumbed for IPv6 and may affect failover capability
  ```

- This message should appear when you have configured a test address for an interface.

```
Created new logical interface hme0:1
May 24 14:16:53 host1 in.mpathd[101180]: Test address now configured on interface hme0;
 enabling probe-based failure detection on it
```

**See Also** If you want the IPMP group to have an active-standby configuration, go on to "How to Configure a Standby Interface for an IPMP Group" on page 743.

## Configuring Target Systems

Probe-based failure detection involves the use of target systems, as explained in "Probe-Based Failure Detection" on page 726. For some IPMP groups, the default targets used by `in.mpathd` is sufficient. However, for some IPMP groups, you might want to configure specific targets for probe-based failure detection. You accomplish probe-based failure detection by setting up host routes in the routing table as probe targets. Any host routes that are configured in the routing table are listed before the default router. Therefore, IPMP uses the explicitly defined host routes for target selection. You can use either of two methods for directly specifying targets: manually setting host routes or creating a shell script that can become a startup script.

Consider the following criteria when evaluating which hosts on your network might make good targets.

- Make sure that the prospective targets are available and running. Make a list of their IP addresses.
- Ensure that the target interfaces are on the same network as the IPMP group that you are configuring.
- The netmask and broadcast address of the target systems must be the same as the addresses in the IPMP group.
- The target host must be able to answer ICMP requests from the interface that is using probe-based failure detection.

## How to Manually Specify Target Systems for Probe-Based Failure Detection

1. **Log in with your user account to the system where you are configuring probe-based failure detection.**

2. **Add a route to a particular host to be used as a target in probe-based failure detection.**

   ```
   $ route add -host destination-IP gateway-IP -static
   ```

   Replace the values of *destination-IP* and *gateway-IP* with the IPv4 address of the host to be used as a target. For example, you would type the following to specify the target system `192.168.85.137`, which is on the same subnet as the interfaces in IPMP group `testgroup1`.

   ```
   $ route add -host 192.168.85.137 192.168.85.137 -static
   ```

3. **Add routes to additional hosts on the network to be used as target systems.**

## How to Specify Target Systems in a Shell Script

1. **On the system where you have configured an IPMP group, assume the Primary Administrator role or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Create a shell script that sets up static routes to your proposed targets.**

   For example, you could create a shell script called `ipmp.targets` with the following contents:

   ```
   TARGETS="192.168.85.117 192.168.85.127 192.168.85.137"

   case "$1" in
           'start')
               /usr/bin/echo "Adding static routes for use as IPMP targets"
           for target in $TARGETS; do
         /usr/sbin/route add -host $target $target
           done
                     ;;
           'stop')
                 /usr/bin/echo "Removing static routes for use as IPMP targets"
            for target in $TARGETS; do
           /usr/sbin/route delete -host $target $target
            done
                     ;;
     esac
   ```

3 **Copy the shell script to the startup script directory.**

```
# cp ipmp.targets /etc/init.d
```

4 **Change the permissions on the new startup script.**

```
# chmod 744 /etc/init.d/ipmp.targets
```

5 **Change ownership of the new startup script.**

```
# chown root:sys /etc/init.d/ipmp.targets
```

6 **Create a link for the startup script in the `/etc/init.d` directory.**

```
# ln /etc/init.d/ipmp.targets /etc/rc2.d/S70ipmp.targets
```

The S70 prefix in the file name `S70ipmp.targets` orders the new script properly with respect to other startup scripts.

## Configuring Standby Interfaces

Use this procedure if you want the IPMP group to have an active-standby configuration. For more information on this type of configuration, refer to "IPMP Interface Configurations" on page 724.

### ▼ How to Configure a Standby Interface for an IPMP Group

**Before You Begin**

- You must have configured all interfaces as members of the IPMP group.
- You should not have configured a test address on the interface to become the standby interface.

For information on configuring an IPMP group and assigning test addresses, refer to "How to Configure an IPMP Group With Multiple Interfaces" on page 737.

1 **On the system with the standby interfaces to be configured, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Configure an interface as a standby and assign the test address.**

```
# ifconfig interface plumb ip-address <other-parameters> deprecated -failover standby up
```

A standby interface can have only one IP address, the test address. You must set the `-failover` option before you set the `standby up` option. For `<other-parameters>`, use the parameters that are required by your configuration, as described in the `ifconfig`(1M) man page.

- For example, to create an IPv4 test address, you would type the following command:

```
# ifconfig hme1 plumb 192.168.85.22 netmask + broadcast + deprecated -failover standby up
```

| | |
|---|---|
| `hme1` | Defines `hme1` as the physical interface to be configured as the standby interface. |
| `192.168.85.22` | Assigns this test address to the standby interface. |
| `deprecated` | Indicates that the test address is not used for outbound packets. |
| `-failover` | Indicates that the test address does not fail over if the interface fails. |
| `standby` | Marks the interface as a standby interface. |

- For example, to create an IPv6 test address, you would type the following command:

  ```
  # ifconfig hme1 plumb -failover standby up
  ```

**3 Check the results of the standby interface configuration.**

```
# ifconfig hme1
hme1: flags=69040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER,
      STANDBY,INACTIVE mtu 1500
         index 4 inet 192.168.85.22 netmask ffffff00 broadcast 19.16.85.255
         groupname test
```

The `INACTIVE` flag indicates that this interface is not used for any outbound packets. When a failover occurs on this standby interface, the `INACTIVE` flag is cleared.

---

**Note –** You can always view the current status of an interface by typing the `ifconfig` *interface* command. For more information on viewing interface status, refer to"How to Get Information About a Specific Interface" on page 205 .

---

**4 (Optional) Preserve the IPv4 standby interface across reboots.**

Assign the standby interface to the same IPMP group, and configure a test address for the standby interface.

For example, to configure `hme1` as the standby interface, you would add the following line to the `/etc/hostname.hme1` file:

```
192.168.85.22 netmask + broadcast + deprecated group test -failover standby up
```

**5 (Optional) Preserve the IPv6 standby interface across reboots.**

Assign the standby interface to the same IPMP group, and configure a test address for the standby interface.

For example, to configure `hme1` as the standby interface, add the following line to the `/etc/hostname6.hme1` file:

```
-failover group test standby up
```

**Example 31–4** Configuring a Standby Interface for an IPMP Group

Suppose you want to create a test address with the following configuration:

- Physical interface `hme2` as a standby interface
- Test address of `192.168.85.22`
- `deprecated` and `-failover` options set
- Netmask and broadcast address set to the default value

You would type the following:

```
# ifconfig hme2 plumb 192.168.85.22 netmask + broadcast + deprecated -failover standby up
```

The interface is marked as a standby interface only after the address is marked as a `NOFAILOVER` address.

You would remove the standby status of an interface by typing the following:

```
# ifconfig interface -standby
```

# Configuring IPMP Groups With a Single Physical Interface

When you have only one interface in an IPMP group, failover is not possible. However, you can enable failure detection on that interface by assigning the interface to an IPMP group. You do not have to configure a dedicated test IP address to establish failure detection for a single interface IPMP group. You can use a single IP address for sending data and detecting failure.

## ▼ How to Configure a Single Interface IPMP Group

**1 On the system with the prospective single interface IPMP group, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 For IPv4, create the single interface IPMP group.**

Use the following syntax to assign the single interface to an IPMP group.

```
# ifconfig interface group group-name
```

The following example assigns the interface `hme0` into the IPMP group `v4test`:

```
# ifconfig hme0 group v4test
```

After this step is performed, IPMP enables link-based failure detection on the interface.

In addition, you can also use the `-failover` subcommand of the `ifconfig` command to enable probe-based failure detection. The following example enables probe-based failure detection on `hme0` by using the IP address currently assigned to `hme0`:

```
# ifconfig hme0 -failover
```

Note that unlike multiple-interface groups, the same IP address can act as both a data address and a test address. To enable applications to use the test address as a data address, test addresses must never be marked `deprecated` on single-interface IPMP groups.

**3 For IPv6, create the single interface IPMP group.**

Use the following syntax to assign a single interface to an IPMP group:

**# ifconfig** *interface* **inet6 group** *group-name*

For example, to add the single interface `hme0` into the IPMP group `v6test`, type the following:

```
# ifconfig hme0 inet6 group v6test
```

After this step is performed, IPMP enables link-based failure detection on the interface.

In addition, you can also use the `-failover` subcommand of the `ifconfig` command to enable probe-based failure detection. The following example enables probe-based failure detection on `hme0` by using the IP address currently assigned to `hme0`:

```
# ifconfig hme0 inet6 -failover
```

Note that unlike multiple-interface groups, the same IP address can act as both a data address and a test address. To enable applications to use the test address as a data address, test addresses must never be marked `deprecated` on single-interface IPMP groups.

In a single physical interface configuration, you cannot verify whether the target system that is being probed has failed or whether the interface has failed. The target system can be probed through only one physical interface. If only one default router is on the subnet, turn off IPMP if a single physical interface is in the group. If a separate IPv4 and IPv6 default router exists, or multiple default routers exist, more than one target system needs to be probed. Hence, you can safely turn on IPMP.

# Maintaining IPMP Groups

This section contains tasks for maintaining existing IPMP groups and the interfaces that compose those groups. The tasks presume that you have already configured an IPMP group, as explained in "Configuring IPMP Groups" on page 735.

## ▼ How to Display the IPMP Group Membership of an Interface

1 **On the system with the IPMP group configuration, become superuser or assume an equivalent role.**

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" in *System Administration Guide: Security Services*.

2 **Display information about the interface, including the group to which the interface belongs.**

```
# ifconfig interface
```

3 **If applicable, display IPv6 information for the interface.**

```
# ifconfig interface inet6
```

**Example 31–5** Displaying Physical Interface Groups

To display the group name for `hme0`, you would type the following:

```
# ifconfig hme0
    hme0: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500
      index 2 inet 192.168.85.19 netmask ffffff00 broadcast 192.168.85.255
      groupname testgroup1
```

To display the group name for only the IPv6 information, you would type the following:

```
# ifconfig hme0 inet6
    hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
            inet6 fe80::a00:20ff:feb9:19fa/10
            groupname testgroup1
```

## ▼ How to Add an Interface to an IPMP Group

1 **On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Add the interface to the IPMP group.**

```
# ifconfig interface group group-name
```

The interface specified in *interface* becomes a member of IPMP group *group-name*.

**Example 31–6** Adding an Interface to an IPMP Group

To add `hme0` to the IPMP group `testgroup2`, you would type the following command:

```
# ifconfig hme0 group testgroup2
  hme0: flags=9000843<UP ,BROADCAST,RUNNING,MULTICAST,IPv4,NOFAILOVER> mtu 1500 index 2
  inet 192.168.85.19 netmask ff000000 broadcast 10.255.255.255
  groupname testgroup2
  ether 8:0:20:c1:8b:c3
```

## ▼ How to Remove an Interface From an IPMP Group

When you execute the `ifconfig` command's `group` parameter with a null string, the interface is removed from its current IPMP group. Be careful when removing interfaces from a group. If some other interface in the IPMP group has failed, a failover could have happened earlier. For example, if `hme0` failed previously, all addresses are failed over to `hme1`, if `hme1` is part of the same group. The removal of `hme1` from the group causes the `in.mpathd` daemon to return all the failover addresses to some other interface in the group. If no other interfaces are functioning in the group, failover might not restore all the network accesses.

Similarly, when an interface in a group needs to be unplumbed, you should first remove the interface from the group. Then, ensure that the interface has all the original IP addresses configured. The `in.mpathd` daemon tries to restore the original configuration of an interface that is removed from the group. You need to ensure that the configuration is restored before unplumbing the interface. Refer to "What Happens During Interface Failover" on page 727 to see how interfaces look before and after a failover.

1. **On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Remove the interface from the IPMP group.**

   ```
   # ifconfig interface group ""
   ```

   The quotation marks indicate a null string.

**Example 31–7** Removing an Interface From a Group

To remove `hme0` from the IPMP group `test`, you would type the following command:

```
# ifconfig hme0 group ""
    # ifconfig hme0
    hme0: flags=9000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500
    index 2 inet 192.168.85.19 netmask ffffff00 broadcast 192.168.85.255
    # ifconfig hme0 inet6
    hme0: flags=a000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
    inet6 fe80::a00:20ff:feb9:19fa/10
```

## ▼ How to Move an Interface From One IPMP Group to Another Group

You can place an interface in a new IPMP group when the interface belongs to an existing IPMP group. You do not need to remove the interface from the current IPMP group. When you place the interface in a new group, the interface is automatically removed from any existing IPMP group.

**1 On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Move the interface to a new IPMP group.**

`# ifconfig` *interface* `group` *group-name*

Placing the interface in a new group automatically removes the interface from any existing group.

**Example 31–8** Moving an Interface to a Different IPMP Group

To change the IPMP group of interface `hme0`, you would type the following:

```
# ifconfig hme0 group cs-link
```

This command removes the `hme0` interface from IPMP group `test` and then puts the interface in the group `cs-link`.

# Replacing a Failed Physical Interface on Systems That Support Dynamic Reconfiguration

This section contains procedures that relate to administering systems that support dynamic reconfiguration (DR).

---

**Note –** The tasks pertain only to IP layers that are configured by using the `ifconfig` command. Layers before or after the IP layer, such as ATM or other services, require specific manual steps if the layers are not automated. The steps in the next procedures are used to unconfigure interfaces during predetachment and configure interface after postattachment.

---

## ▼ How to Remove a Physical Interface That Has Failed (DR-Detach)

This procedure shows how to remove a physical interface on a system that supports DR. The procedure assumes that the following conditions already exist:

- Physical interfaces `hme0` and `hme1` are the example interfaces.
- Both interfaces are in the same IPMP group.
- `hme0` has failed.
- Logical interface `hme0:1` has the test address.
- You are replacing the failed interface with the same physical interface name, for example, `hme0` with `hme0`.

---

**Note –** You can skip Step 2 if the test address is plumbed by using the `/etc/hostname.hme0` file.

---

**1 On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Display the test address configuration.**

```
# ifconfig hme0:1

hme0:1:
flags=9040842<BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,NOFAILOVER>
mtu 1500 index 3
```

```
inet 192.168.233.250 netmask ffffff00 broadcast 192.168.233.255
```

You need this information to replumb the test address when replacing the physical interface.

**3 Remove the physical interface.**

Refer to the following sources for a complete description of how to remove the physical interface:

- `cfgadm`(1M) man page
- *Sun Enterprise 6x00, 5x00, 4x00, and 3x00 Systems Dynamic Reconfiguration User's Guide*
- *Sun Enterprise 10000 DR Configuration Guide*

## ▼ How to Replace a Physical Interface That Has Failed (DR-Attach)

This procedure shows how to replace a physical interface on a system that supports DR.

**1 On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Replace the physical interface.**

Refer to the instructions in the following sources:

- `cfgadm`(1M) man page
- *Sun Enterprise 6x00, 5x00, 4x00, and 3x00 Systems Dynamic Reconfiguration User's Guide*
- *Sun Enterprise 10000 DR Configuration Guide*, or *Sun Fire 880 Dynamic Reconfiguration User's Guide*

# Recovering a Physical Interface That Was Not Present at System Boot

**Note –** The following procedure pertains only to IP layers that are configured by using the `ifconfig` command. Layers before or after the IP layer, such as ATM or other services, require specific manual steps if the layers are not automated. The specific steps in the next procedure are used to unconfigure interfaces during predetachment and to configure interfaces after postattachment.

Recovery after dynamic reconfiguration is automatic for an interface that is part of the I/O board on a Sun Fire™ platform. If the NIC is a Sun Crypto Accelerator I - cPCI board, the recovery is also automatic. Consequently, the following steps are not required for an interface that is coming back as part of a DR operation. For more information on the Sun Fire x800 and Sun Fire 15000 systems, see the `cfgadm_sbd`(1M) man page. The physical interface fails back to the configuration that is specified in the `/etc/hostname.`*interface* file. See "Configuring IPMP Groups" on page 735 for details on how to configure interfaces to preserve the configuration across reboots.

**Note –** On Sun Fire legacy (Exx00) systems, DR detachments are still subject to manual procedures. However, DR attachments are automated.

## ▼ How to Recover a Physical Interface That Was Not Present at System Boot

You must complete the following procedure before you recover a physical interface that was not present at system boot. The example in this procedure has the following configuration:

- Physical interfaces `hme0` and `hme1` are the interfaces.
- Both interfaces are in the same IPMP group.
- `hme0` was not installed at system boot.

**Note –** The failback of IP addresses during the recovery of a failed physical interface takes up to three minutes. This time might vary, depending on network traffic. The time also depends on the stability of the incoming interface to fail back the failed-over interfaces by the `in.mpathd` daemon.

1. **On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Retrieve the failed network information from the failure error message of the console log.**

   See the `syslog`(3C)man page. The error message might be similar to the following:

   ```
   moving addresses from failed IPv4 interfaces:
   hme1 (moved to hme0)
   ```

   This message indicates that the IPv4 addresses on the failed interface `hme1` have failed over to the `hme0` interface.

   Alternatively, you might receive the following similar message:

   ```
   moving addresses from failed IPv4 interfaces:
   hme1 (couldn't move, no alternative interface)
   ```

   This message indicates that no active interface could be found in the same group as failed interface `hme1`. Therefore, the IPv4 addresses on `hme1` could not fail over.

3. **Attach the physical interface to the system.**

   Refer to the following for instructions on how to replace the physical interface:

   - `cfgadm`(1M) man page
   - *Sun Enterprise 10000 DR Configuration Guide*
   - *Sun Enterprise 6x00, 5x00, 4x00, and 3x00 Systems Dynamic Reconfiguration User's Guide*

4. **Refer to the message content from Step 2. If the addresses could not be moved, go to Step 6. If the addresses were moved, continue to Step 5.**

5. **Unplumb the logical interfaces that were configured as part of the failover process.**

   a. **Review the contents of the `/etc/hostname.`*moved-from-interface* file to determine what logical interfaces were configured as part of the failover process.**

   b. **Unplumb each failover IP address.**

      ```
      # ifconfig moved-to-interface removeif moved-ip-address
      ```

      **Note –** Failover addresses are marked with the `failover` parameter, or are not marked with the `-failover` parameter. You do not need to unplumb IP addresses that are marked `-failover`.

For example, assume that the contents of the `/etc/hostname.hme0` file contains the following lines:

```
inet 10.0.0.4 -failover up group one
addif 10.0.0.5 failover up
addif 10.0.0.6 failover up
```

To unplumb each failover IP address, you would type the following commands:

```
# ifconfig hme0 removeif 10.0.0.5
# ifconfig hme0 removeif 10.0.0.6
```

**6 Reconfigure the IPv4 information for the replaced physical interface by typing the following command for each interface that was removed:**

`# ifconfig` *removed-from-NIC* `<parameters>`

For example, you would type the following commands:

```
# ifconfig hme1 inet plumb
# ifconfig hme1 inet 10.0.0.4 -failover up group one
# ifconfig hme1 addif 10.0.0.5 failover up
# ifconfig hme1 addif 10.0.0.6 failover up
```

# Modifying IPMP Configurations

Use the IPMP configuration file `/etc/default/mpathd` to configure the following system-wide parameters for IPMP groups.

- `FAILURE_DETECTION_TIME`
- `TRACK_INTERFACES_ONLY_WITH_GROUPS`
- `FAILBACK`

## ▼ How to Configure the `/etc/default/mpathd` File

**1 On the system with the IPMP group configuration, assume the Primary Administrator role or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Edit the `/etc/default/mpathd` file.**

Change the default value of one or more of the three parameters.

**a. Type the new value for the `FAILURE_DETECTION_TIME` parameter.**

```
FAILURE_DETECTION_TIME=n
```

where *n* is the amount of time in seconds for ICMP probes to detect whether an interface failure has occurred. The default is 10 seconds.

**b. Type the new value for the `FAILBACK` parameter.**

```
FAILBACK=[yes | no]
```

- *yes*- The *yes* value is the default failback behavior of IPMP. When the repair of a failed interface is detected, network access fails back to the repaired interface, as described in "IPMP Failure Detection and Recovery Features" on page 725.
- *no* - The *no* indicates that data traffic does not move back to a repaired interface. When a failed interfaces is detected as repaired, the *INACTIVE* flag is set for that interface. This flag indicates that the interface is currently not to be used for data traffic. The interface can still be used for probe traffic.

  For example, suppose an IPMP group consists of two interfaces, ce0 and ce1. Then assume that the value *FAILBACK=no* is set in `/etc/default/mpathd`. If ce0 fails, its traffic fails over to ce1, as is the expected behavior of IPMP. However, when IPMP detects that ce0 is repaired, traffic does not fail back from ce1, due to the *FAILBACK=no* parameter in `/etc/default/mpathd`. The ce0 interface retains its `INACTIVE` status and is not used for traffic unless the ce1 interface fails. If the ce1 interface fails, the addresses on ce1 are migrated back to ce0, whose INACTIVE flag is then cleared. This migration occurs provided that ce0 is the only INACTIVE interface in the group. If other INACTIVE interfaces exist in the group, the addresses might be migrated to an INACTIVE interface other than ce0.

**c. Type the new value for the `TRACK_INTERFACES_ONLY_WITH_GROUPS` parameter.**

```
TRACK_INTERFACES_ONLY_WITH_GROUPS=[yes | no]
```

- *yes*- The *yes* value is the default behavior of IPMP. This parameter causes IPMP to ignore network interfaces that are not configured into an IPMP group.
- *no* - The *no* value sets failure and repair detection for *all* network interfaces, regardless of whether they are configured into an IPMP group. However, when a failure or repair is detected on an interface that is not configured into an IPMP group, no failover or failback occurs. Therefore, the*no* value is only useful for reporting failures and does not directly improve network availability.

**3 Restart the `in.mpathd` daemon.**

```
# pkill -HUP in.mpathd
```

PART VII

# IP Quality of Service (IPQoS)

This part contains tasks and information about IP Quality of Service (IPQoS), the Solaris operating system's implementation of differentiated services.

◆ ◆ ◆ CHAPTER 32

# Introducing IPQoS (Overview)

IP Quality of Service (IPQoS) enables you to prioritize, control, and gather accounting statistics. Using IPQoS, you can provide consistent levels of service to users of your network. You can also manage traffic to avoid network congestion.

The following is a list of topics in this chapter:


- "IPQoS Basics" on page 759
- "Providing Quality of Service With IPQoS" on page 762
- "Improving Network Efficiency With IPQoS" on page 763
- "Differentiated Services Model" on page 764
- "Traffic Forwarding on an IPQoS-Enabled Network" on page 769


## IPQoS Basics

IPQoS enables the Differentiated Services (Diffserv) architecture that is defined by the Differentiated Services Working Group of the Internet Engineering Task Force (IETF). In the Solaris OS, IPQoS is implemented at the IP level of the TCP/IP protocol stack.

### What Are Differentiated Services?

By enabling IPQoS, you can provide different levels of network service for selected customers and selected applications. The different levels of service are collectively referred to as *differentiated services*. The differentiated services that you provide to customers can be based on a structure of service levels that your company offers to its customers. You can also provide differentiated services based on the priorities that are set for applications or users on your network.

Providing quality of service involves the following activities:

- Delegating levels of service to different groups, such as customers or departments in an enterprise

- Prioritizing network services that are given to particular groups or applications
- Discovering and eliminating areas of network bottlenecks and other forms of congestion
- Monitoring network performance and providing performance statistics
- Regulating bandwidth to and from network resources

# IPQoS Features

IPQoS has the following features:

- `ipqosconf` Command-line tool for configuring the QoS policy
- Classifier that selects actions, which are based on filters that configure the QoS policy of your organization
- Metering module that measures network traffic, in compliance with the Diffserv model
- Service differentiation that is based on the ability to mark a packet's IP header with forwarding information
- Flow-accounting module that gathers statistics for traffic flows
- Statistics gathering for traffic classes, through the UNIX® `kstat` command
- Support for SPARC® and x86 architecture

**Note –** The x86 architecture does not support IPQoS on VLANs.

- Support for IPv4 and IPv6 addressing
- Interoperability with IP Security Architecture (IPsec)
- Support for 802.1D user-priority markings for virtual local area networks (VLANs)

# Where to Get More Information About Quality-of-Service Theory and Practice

You can find information on differentiated services and quality of service from print and online sources.

## Books About Quality of Service

For more information on quality-of-service theory and practice, refer to the following books:

- Ferguson, Paul and Geoff Huston. *Quality of Service*. John Wiley & Sons, Inc., 1998.
- Kilkki, Kalevi. *Differentiated Services for the Internet*. Macmillan Technical Publishing, 1999.

## Requests for Comments (RFCs) About Quality of Service

IPQoS conforms to the specifications that are described in the following RFCs and the following Internet drafts:

- RFC 2474, Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers (`http://www.ietf.org/rfc/rfc2474.txt?number=2474`) – Describes an enhancement to the type of service (ToS) field or DS fields of the IPv4 and IPv6 packet headers to support differentiated services
- RFC 2475, An Architecture for Differentiated Services (`http://www.ietf.org/rfc/rfc2475.txt?number=2475`) – Provides a detailed description of the organization and modules of the Diffserv architecture
- RFC 2597, Assured Forwarding PHB Group (`http://www.ietf.org/rfc/rfc2597.txt?number=2597`) – Describes how the assured forwarding (AF) per-hop behavior works.
- RFC 2598, An Expedited Forwarding PHB (`http://www.ietf.org/rfc/rfc2598.txt?number=2598`) – Describes how the expedited forwarding (EF) per-hop behavior works
- Internet-Draft, *An Informal Management Model for Diffserv Routers* – Presents a model for implementing the Diffserv architecture on routers.

## Web Sites With Quality-of-Service Information

The Differentiated Services Working Group of the IETF maintains a web site with links to Diffserv Internet drafts at `http://www.ietf.org/html.charters/diffserv-charter.html`.

Router manufacturers such as Cisco Systems and Juniper Networks provide information on their corporate web sites that describes how Differentiated Services are implemented in their products.

## IPQoS Man Pages

IPQoS documentation includes the following man pages:

- `ipqosconf`(1M) - Describes the command for setting up the IPQoS configuration file
- `ipqos`(7ipp) – Describes the IPQoS implementation of the Diffserv architectural model
- `ipgpc`(7ipp) – Describes the IPQoS implementation of a Diffserv classifier
- `tokenmt`(7ipp) – Describes the IPQoS `tokenmt` meter
- `tswtclmt`(7ipp) – Describes the IPQoS `tswtclmt` meter
- `dscpmk`(7ipp) – Describes the DSCP marker module
- `dlcosmk`(7ipp) – Describes the IPQoS 802.1D user-priority marker module
- `flowacct`(7ipp)– Describes the IPQoS flow-accounting module

- `acctadm`(1M) – Describes the command that configures the Solaris extended accounting facilities. The `acctadm` command includes IPQoS extensions.

# Providing Quality of Service With IPQoS

IPQoS features enable Internet service providers (ISPs) and application service providers (ASPs) to offer different levels of network service to customers. These features enable individual companies and educational institutions to prioritize services for internal organizations or for major applications.

## Implementing Service-Level Agreements

If your organization is an ISP or ASP, you can base your IPQoS configuration on the *service-level agreement* (SLA) that your company offers to its customers. In an SLA, a service provider guarantees to a customer a certain level of network service that is based on a price structure. For example, a premium-priced SLA might ensure that the customer receives highest priority for all types of network traffic 24 hours per day. Conversely, a medium-priced SLA might guarantee that the customer receives high priority for email only during business hours. All other traffic would receive medium priority 24 hours a day.

## Assuring Quality of Service for an Individual Organization

If your organization is an enterprise or an institution, you can also provide quality-of-service features for your network. You can guarantee that traffic from a particular group or from a certain application is assured a higher or lower degree of service.

## Introducing the Quality-of-Service Policy

You implement quality of service by defining a *quality-of-service (QoS) policy*. The QoS policy defines various network attributes, such as customers' or applications' priorities, and actions for handling different categories of traffic. You implement your organization's QoS policy in an IPQoS configuration file. This file configures the IPQoS modules that reside in the Solaris OS kernel. A host with an applied IPQoS policy is considered an *IPQoS-enabled system*.

Your QoS policy typically defines the following:

- Discrete groups of network traffic that are called *classes of service*.
- Metrics for regulating the amount of network traffic for each class. These metrics govern the traffic-measuring process that is called *metering*.

- An action that an IPQoS system and a Diffserv router must apply to a packet flow. This type of action is called a *per-hop behavior* (PHB).
- Any statistics gathering that your organization requires for a class of service. An example is traffic that is generated by a customer or particular application.

When packets pass to your network, the IPQoS-enabled system evaluates the packet headers. The action that the IPQoS system takes is determined by your QoS policy.

Tasks for designing the QoS policy are described in "Planning the Quality-of-Service Policy" on page 777.

# Improving Network Efficiency With IPQoS

IPQoS contains features that can help you make network performance more efficient as you implement quality of service. When computer networks expand, the need also increases for managing network traffic that is generated by increasing numbers of users and more powerful processors. Some symptoms of an overused network include lost data and traffic congestion. Both symptoms result in slow response times.

In the past, system administrators handled network traffic problems by adding more bandwidth. Often, the level of traffic on the links varied widely. With IPQoS, you can manage traffic on the existing network and help assess where, and whether, expansion is necessary.

For example, for an enterprise or institution, you must maintain an efficient network to avoid traffic bottlenecks. You must also ensure that a group or application does not consume more than its allotted bandwidth. For an ISP or ASP, you must manage network performance to ensure that customers receive their paid-for level of network service.

## How Bandwidth Affects Network Traffic

You can use IPQoS to regulate network *bandwidth*, the maximum amount of data that a fully used network link or device can transfer. Your QoS policy should prioritize the use of bandwidth to provide quality of service to customers or users. The IPQoS metering modules enable you to measure and control bandwidth allocation among the various traffic classes on an IPQoS-enabled host.

Before you can effectively manage traffic on your network, you must answer these questions about bandwidth usage:

- What are the traffic problem areas for your local network?
- What must you do to achieve optimum use of available bandwidth?
- What are your site's critical applications, which must be given highest priority?
- Which applications are sensitive to congestion?

- What are your less critical applications, which can be given a lower priority?

## Using Classes of Service to Prioritize Traffic

To implement quality of service, you analyze network traffic to determine any broad groupings into which the traffic can be divided. Then, you organize the various groupings into classes of service with individual characteristics and individual priorities. These classes form the basic categories on which you base the QoS policy for your organization. The classes of service represent the traffic groups that you want to control.

For example, a provider might offer platinum, gold, silver, and bronze levels of service, available at a sliding price structure. A platinum SLA might guarantee top priority to incoming traffic that is destined for a web site that the ISP hosts for the customer. Thus, incoming traffic to the customer's web site could be one traffic class.

For an enterprise, you could create classes of service that are based on department requirements. Or, you could create classes that are based on the preponderance of a particular application in the network traffic. Here are a few examples of traffic classes for an enterprise:

- Popular applications such as email and outgoing FTP to a particular server, either of which could constitute a class. Because employees constantly use these applications, your QoS policy might guarantee email and outgoing FTP a small amount of bandwidth and a lower priority.
- An order-entry database that needs to run 24 hours a day. Depending on the importance of the database application to the enterprise, you might give the database a large amount of bandwidth and a high priority.
- A department that performs critical work or sensitive work, such as the payroll department. The importance of the department to the organization would determine the priority and amount of bandwidth you would give to such a department.
- Incoming calls to a company's external web site. You might give this class a moderate amount of bandwidth that runs at low priority.

# Differentiated Services Model

IPQoS includes the following modules, which are part of the *Differentiated Services (Diffserv)* architecture that is defined in RFC 2475:

- Classifier
- Meter
- Marker

IPQoS adds the following enhancements to the Diffserv model:

- Flow-accounting module

- 802.1D datagram marker

This section introduces the Diffserv modules as they are used by IPQoS. You need to know about these modules, their names, and their uses to set up the QoS policy. For detailed information about each module, refer to "IPQoS Architecture and the Diffserv Model" on page 835.

# Classifier (ipgpc) Overview

In the Diffserv model, the *classifier* selects packets from a network traffic flow. A *traffic flow* consists of a group of packets with identical information in the following IP header fields:

- Source address
- Destination address
- Source port
- Destination port
- Protocol number

In IPQoS, these fields are referred to as the *5-tuple*.

The IPQoS classifier module is named `ipgpc`. The `ipgpc` classifier arranges traffic flows into classes that are based on characteristics you configure in the IPQoS configuration file.

For detailed information about `ipgpc`, refer to "Classifier Module" on page 835.

## IPQoS Classes

A *class* is a group of network flows that share similar characteristics. For example, an ISP might define classes to represent the different service levels that are offered to customers. An ASP might define SLAs that give different levels of service to various applications. For an ASP's QoS policy, a class might include outgoing FTP traffic that is bound for a particular destination IP address. Outgoing traffic from a company's external web site might also be defined as a class.

Grouping traffic into classes is a major part of planning your QoS policy. When you create classes by using the `ipqosconf` utility, you are actually configuring the `ipgpc` classifier.

For information on how to define classes, see "How to Define the Classes for Your QoS Policy" on page 780.

## IPQoS Filters

*Filters* are sets of rules that contain parameters called *selectors*. Each filter must point to a class. IPQoS matches packets against the selectors of each filter to determine if the packet belongs to the filter's class. You can filter on a packet by using a variety of selectors, for example, the IPQoS 5-tuple and other common parameters:

- Source address and destination addresses

- Source port and destination port
- Protocol numbers
- User IDs
- Project IDs
- Differentiated Services Codepoint (DSCP)
- Interface index

For example, a simple filter might include the destination port with the value of 80. The `ipgpc` classifier then selects all packets that are bound for destination port 80 (HTTP) and handles the packets as directed in the QoS policy.

For information on creating filters, see "How to Define Filters in the QoS Policy" on page 783.

## Meter (`tokenmt` and `tswtclmt`) Overview

In the Diffserv model, the *meter* tracks the transmission rate of traffic flows on a per-class basis. The meter evaluates how much the actual rate of the flow conforms to the configured rates to determine the appropriate outcome. Based on the traffic flow's outcome, the meter selects a subsequent action. Subsequent actions might include sending the packet to another action or returning the packet to the network without further processing.

The IPQoS meters determine whether a network flow conforms to the transmission rate that is defined for its class in the QoS policy. IPQoS includes two metering modules:

- `tokenmt` – Uses a two-token bucket metering scheme
- `tswtclmt` – Uses a time-sliding window metering scheme

Both metering modules recognize three outcomes: red, yellow, and green. You define the actions to be taken for each outcome in the parameters `red_action_name`, `yellow_action_name`, and `green_action_name`.

In addition, you can configure `tokenmt` to be color aware. A color-aware metering instance uses the packet's size, DSCP, traffic rate, and configured parameters to determine the outcome. The meter uses the DSCP to map the packet's outcome to a green, yellow, or red.

For information on defining parameters for the IPQoS meters, refer to "How to Plan Flow Control" on page 784.

## Marker (`dscpmk` and `dlcosmk`) Overview

In the Diffserv model, the *marker* marks a packet with a value that reflects a forwarding behavior. *Marking* is the process of placing a value in the packet's header to indicate how to forward the packet to the network. IPQoS contains two marker modules:

- `dscpmk` – Marks the DS field in an IP packet header with a numeric value that is called the *Differentiated Services codepoint*, or *DSCP*. A Diffserv-aware router can then use the DS codepoint to apply the appropriate forwarding behavior to the packet.
- `dlcosmk` – Marks the virtual local area network (VLAN) tag of an Ethernet frame header with a numeric value that is called the *user priority*. The user priority indicates the *class of service (CoS)*, which defines the appropriate forwarding behavior to be applied to the datagram.

  `dlcosmk` is an IPQoS addition that is not part of the Diffserv model, as designed by the IETF.

For information on implementing a marker strategy for the QoS policy, see "How to Plan Forwarding Behavior" on page 786.

## Flow Accounting (`flowacct`) Overview

IPQoS adds the `flowacct` accounting module to the Diffserv model. You can use `flowacct` to gather statistics on traffic flows, and bill customers in agreement with their SLAs. Flow accounting is also useful for capacity planning and system monitoring.

The `flowacct` module works with the `acctadm` command to create an accounting log file. A basic log includes the IPQoS 5-tuple and two additional attributes, as shown in the following list:

- Source address
- Source port
- Destination address
- Destination port
- Protocol number
- Number of packets
- Number of bytes

You can also gather statistics on other attributes, as described in "Recording Information About Traffic Flows" on page 829, and in the `flowacct`(7ipp) and `acctadm`(1M) man pages.

For information on planning a flow-accounting strategy, see "How to Plan for Flow Accounting" on page 789.

## How Traffic Flows Through the IPQoS Modules

The next figure shows a path that incoming traffic might take through some of the IPQoS modules.

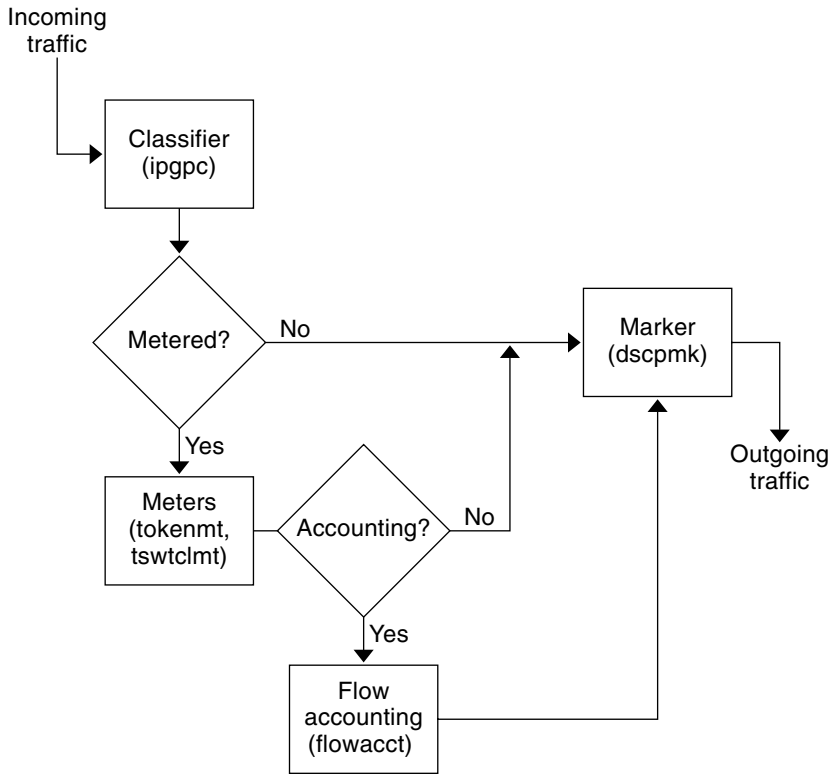



**FIGURE 32–1** Traffic Flow Through the IPQoS Implementation of the Diffserv Model

This figure illustrates a common traffic flow sequence on an IPQoS-enabled machine:

1. The classifier selects from the packet stream all packets that match the filtering criteria in the system's QoS policy.
2. The selected packets are then evaluated for the next action to be taken.
3. The classifier sends to the marker any traffic that does not require flow control.
4. Traffic to be flow-controlled is sent to the meter.
5. The meter enforces the configured rate. Then, the meter assigns a traffic conformance value to the flow-controlled packets.
6. The flow-controlled packets are then evaluated to determine if any packets require accounting.
7. The meter sends to the marker any traffic that does not require flow accounting.
8. The flow-accounting module gathers statistics on received packets. The module then sends the packets to the marker.

9. The marker assigns a DS codepoint to the packet header. This DSCP indicates the per-hop behavior that a Diffserv-aware system must apply to the packet.

# Traffic Forwarding on an IPQoS-Enabled Network

This section introduces the elements that are involved in forwarding packets on an IPQoS-enabled network. An IPQoS-enabled system handles any packets on the network stream with the system's IP address as the destination. The IPQoS system then applies its QoS policy to the packet to establish differentiated services.

## DS Codepoint

The DS codepoint (DSCP) defines in the packet header the action that any Diffserv-aware system should take on a marked packet. The diffserv architecture defines a set of DS codepoints for the IPQoS-enabled system and diffserv router to use. The Diffserv architecture also defines a set of actions that are called *forwarding behaviors*, which correspond to the DSCPs. The IPQoS-enabled system marks the precedence bits of the DS field in the packet header with the DSCP. When a router receives a packet with a DSCP value, the router applies the forwarding behavior that is associated with that DSCP. The packet is then released onto the network.

---

**Note –** The `dlcosmk` marker does not use the DSCP. Rather, `dlcosmk` marks Ethernet frame headers with a CoS value. If you plan to configure IPQoS on a network that uses VLAN devices, refer to "Marker Module" on page 840.

---

## Per-Hop Behaviors

In Diffserv terminology, the forwarding behavior that is assigned to a DSCP is called the *per-hop behavior (PHB)*. The PHB defines the forwarding precedence that a marked packet receives in relation to other traffic on the Diffserv-aware system. This precedence ultimately determines whether the IPQoS-enabled system or Diffserv router forwards or drops the marked packet. For a forwarded packet, each Diffserv router that the packet encounters en route to its destination applies the same PHB. The exception is if another Diffserv system changes the DSCP. For more information on PHBs, refer to "Using the `dscpmk` Marker for Forwarding Packets" on page 841.

The goal of a PHB is to provide a specified amount of network resources to a class of traffic on the contiguous network. You can achieve this goal in the QoS policy. Define DSCPs that indicate the precedence levels for traffic classes when the traffic flows leave the IPQoS-enabled system. Precedences can range from high-precedence/low-drop probability to low-precedence/high-drop probability.

For example, your QoS policy can assign to one class of traffic a DSCP that guarantees a low-drop PHB. This traffic class then receives a low-drop precedence PHB from any Diffserv-aware router, which guarantees bandwidth to packets of this class. You can add to the QoS policy other DSCPs that assign varying levels of precedence to other traffic classes. The lower-precedence packets are given bandwidth by Diffserv systems in agreement with the priorities that are indicated in the packets' DSCPs.

IPQoS supports two types of forwarding behaviors, which are defined in the Diffserv architecture, expedited forwarding and assured forwarding.

## Expedited Forwarding

The *expedited forwarding (EF)* per-hop behavior assures that any traffic class with EFs related DSCP is given highest priority. Traffic with an EF DSCP is not queued. EF provides low loss, latency, and jitter. The recommended DSCP for EF is 101110. A packet that is marked with 101110 receives guaranteed low-drop precedence as the packet traverses Diffserv-aware networks en route to its destination. Use the EF DSCP when assigning priority to customers or applications with a premium SLA.

## Assured Forwarding

The *assured forwarding (AF)* per-hop behavior provides four different forwarding classes that you can assign to a packet. Every forwarding class provides three drop precedences, as shown in Table 37–2.

The various AF codepoints provide the ability to assign different levels of service to customers and applications. In the QoS policy, you can prioritize traffic and services on your network when you plan the QoS policy. You can then assign different AF levels to the prioritized traffic.

## Packet Forwarding in a Diffserv Environment

The following figure shows part of an intranet at a company with a partially Diffserv-enabled environment. In this scenario, all hosts on networks `10.10.0.0` and `10.14.0.0` are IPQoS enabled, and the local routers on both networks are Diffserv aware. However, the interim networks are not configured for Diffserv.

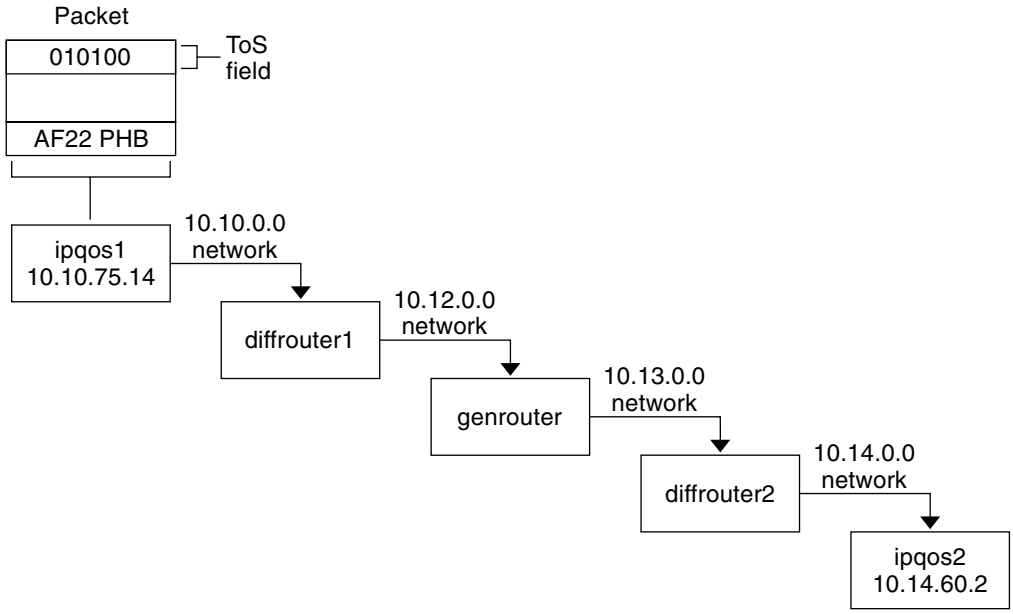



**FIGURE 32–2** Packet Forwarding Across Diffserv-Aware Network Hops

The next steps trace the flow of the packet that is shown in this figure. The steps begin with the progress of a packet that originates at host `ipqos1`. The steps then continue through several hops to host `ipqos2`.

1. The user on `ipqos1` runs the `ftp` command to access host `ipqos2`, which is three hops away.
2. `ipqos1` applies its QoS policy to the resulting packet flow. `ipqos1` then successfully classifies the `ftp` traffic.

   The system administrator has created a class for all outgoing `ftp` traffic that originates on the local network 10.10.0.0. Traffic for the `ftp` class is assigned the AF22 per-hop behavior: class two, medium-drop precedence. A traffic flow rate of 2Mb/sec is configured for the `ftp` class.
3. `ipqos-1` meters the `ftp` flow to determine if the flow exceeds the committed rate of 2 Mbit/sec.
4. The marker on `ipqos1` marks the DS fields in the outgoing `ftp` packets with the 010100 DSCP, corresponding to the AF22 PHB.
5. The router `diffrouter1` receives the `ftp` packets. `diffrouter1` then checks the DSCP. If `diffrouter1` is congested, packets that are marked with AF22 are dropped.
6. `ftp` traffic is forwarded to the next hop in agreement with the per-hop behavior that is configured for AF22 in `diffrouter1`'s files.

7. The `ftp` traffic traverses network `10.12.0.0` to `genrouter`, which is not Diffserv aware. As a result, the traffic receives "best-effort" forwarding behavior.
8. `genrouter` passes the `ftp` traffic to network `10.13.0.0`, where the traffic is received by `diffrouter2`.
9. `diffrouter2` is Diffserv aware. Therefore, the router forwards the `ftp` packets to the network in agreement with the PHB that is defined in the router policy for AF22 packets.
10. `ipqos2` receives the `ftp` traffic. `ipqos2` then prompts the user on `ipqos1` for a user name and password.

◆ ◆ ◆ CHAPTER 33

# Planning for an IPQoS-Enabled Network (Tasks)

You can configure IPQoS on any system that runs the Solaris OS. The IPQoS system then works with Diffserv-aware routers to provide differentiated services and traffic management on an intranet.

This chapter contains planning tasks for adding IPQoS-enabled systems onto a Diffserv-aware network. The following topics are covered.


- "General IPQoS Configuration Planning (Task Map)" on page 773
- "Planning the Diffserv Network Topology" on page 774
- "Planning the Quality-of-Service Policy" on page 777
- "QoS Policy Planning (Task Map)" on page 778
- "Introducing the IPQoS Configuration Example" on page 789


## General IPQoS Configuration Planning (Task Map)

Implementing differentiated services, including IPQoS, on a network requires extensive planning. You must consider not only the position and function of each IPQoS-enabled system, but also each system's relationship to the router on the local network. The following task map lists the major planning tasks for implementing IPQoS on your network.

| Task | Description | For Instructions |
|---|---|---|
| 1. Plan a Diffserv network topology that incorporates IPQoS-enabled systems. | Learn about the various Diffserv network topologies to determine the best solution for your site. | "Planning the Diffserv Network Topology" on page 774. |
| 2. Plan the different types of services to be offered by the IPQoS systems. | Organize the types of services that the network provides into service-level agreements (SLAs). | "Planning the Quality-of-Service Policy" on page 777. |

| Task | Description | For Instructions |
|---|---|---|
| 3. Plan the QoS policy for each IPQoS system. | Decide on the classes, metering, and accounting features that are needed to implement each SLA. | "Planning the Quality-of-Service Policy" on page 777. |
| 4. If applicable, plan the policy for the Diffserv router. | Decide any scheduling and queuing policies for the Diffserv router that is used with the IPQoS systems. | Refer to router documentation for queuing and scheduling policies. |

# Planning the Diffserv Network Topology

To provide differentiated services for your network, you need at least one IPQoS-enabled system and a Diffserv-aware router. You can expand this basic scenario in a variety of ways, as explained in this section.

## Hardware Strategies for the Diffserv Network

Typically, customers run IPQoS on servers and server consolidations, such as the Sun Enterprise™ 0000 server. Conversely, you can also run IPQoS on desktop systems such as UltraSPARC® systems, depending on the needs of your network. The following list describes possible systems for an IPQoS configuration:

- Solaris systems that offer various services, such as web servers and database servers
- Application servers that offer email, FTP, or other popular network applications
- Web cache servers or proxy servers
- Network of IPQoS-enabled server farms that are managed by Diffserv-aware load balancers
- Firewalls that manage traffic for a single heterogeneous network
- IPQoS systems that are part of a virtual local area network (LAN)

You might introduce IPQoS systems into a network topology with already functioning Diffserv-aware routers. If your router does not currently offer Diffserv, consider the Diffserv solutions that are offered by Cisco Systems, Juniper Networks, and other router manufacturers. If the local router does not implement Diffserv, then the router passes marked packets on to the next hop without evaluating the marks.

## IPQoS Network Topologies

This section illustrates IPQoS strategies for various network needs.

## IPQoS on Individual Hosts

The following figure shows a single network of IPQoS-enabled systems.

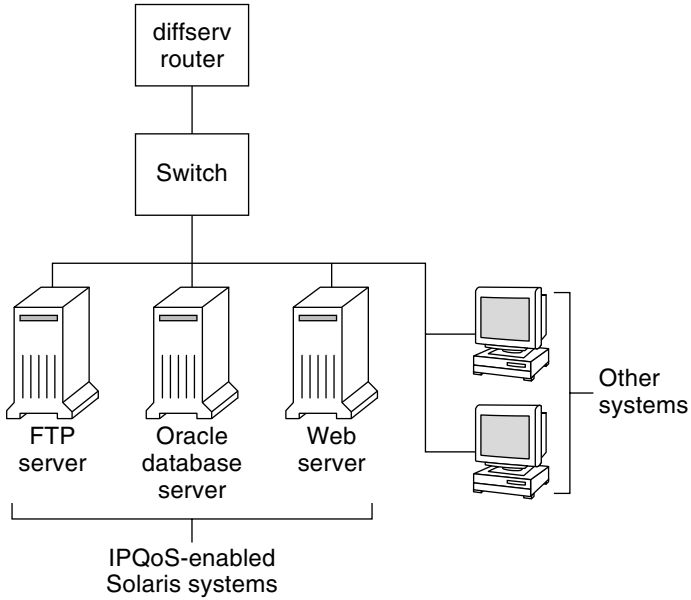



**FIGURE 33–1** IPQoS Systems on a Network Segment

This network is but one segment of a corporate intranet. By enabling IPQoS on the application servers and web servers, you can control the rate at which each IPQoS system releases outgoing traffic. If you make the router Diffserv aware, you can further control incoming and outgoing traffic.

The examples in this guide use the "IPQoS on an individual host" scenario. For the example topology that is used throughout the guide, see Figure 33–4.

## IPQoS on a Network of Server Farms

The following figure shows a network with several heterogeneous server farms.

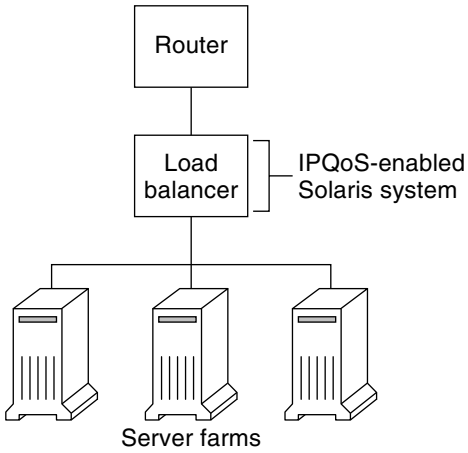



**FIGURE 33–2** Network of IPQoS-Enabled Server Farms

In such a topology, the router is Diffserv aware, and therefore able to queue and rate both incoming and outgoing traffic. The load balancer is also Diffserv-aware, and the server farms are IPQoS enabled. The load balancer can provide additional filtering beyond the router by using selectors such as user ID and project ID. These selectors are included in the application data.

This scenario provides flow control and traffic forwarding to manage congestion on the local network. This scenario also prevents outgoing traffic from the server farms from overloading other portions of the intranet.

## IPQoS on a Firewall

The following figure shows a segment of a corporate network that is secured from other segments by a firewall.

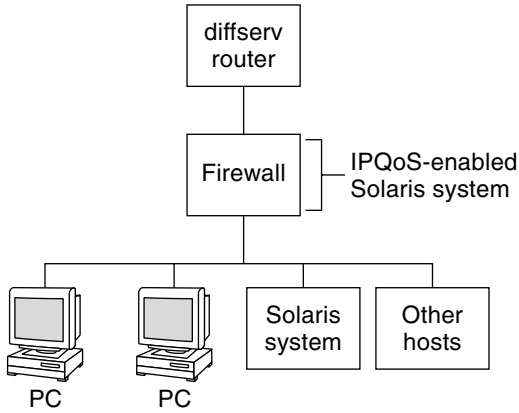



**FIGURE 33–3** Network Protected by an IPQoS-Enabled Firewall

In this scenario, traffic flows into a Diffserv-aware router where the packets are filtered and queued. All incoming traffic that is forwarded by the router then travels into the IPQoS-enabled firewall. To use IPQoS, the firewall must not bypass the IP forwarding stack.

The firewall's security policy determines whether incoming traffic is permitted to enter or depart the internal network. The QoS policy controls the service levels for incoming traffic that has passed the firewall. Depending on the QoS policy, outgoing traffic can also be marked with a forwarding behavior.

# Planning the Quality-of-Service Policy

When you plan the quality-of-service (QoS) policy, you must review, classify, and then prioritize the services that your network provides. You must also assess the amount of available bandwidth to determine the rate at which each traffic class is released onto the network.

## QoS Policy Planning Aids

Gather information for planning the QoS policy in a format that includes the information needed for the IPQoS configuration file. For example, you can use the following template to list the major categories of information to be used in the IPQoS configuration file.

**TABLE 33–1** QoS Planning Template

| Class | Priority | Filter | Selector | Rate | Forwarding? | Accounting? |
|---|---|---|---|---|---|---|
| Class 1 | 1 | Filter 1<br>Filter 3 | Selector 1<br>Selector 2 | Meter rates, depending on meter type | Marker drop precedence | Requires flow-accounting statistics |
| Class 1 | 1 | Filter 2 | Selector 1<br>Selector 2 | N/A | N/A | N/A |
| Class 2 | 2 | Filter 1 | Selector 1<br>Selector 2 | Meter rates, depending on meter type | Marker drop precedence | Requires flow-accounting statistics |
| Class 2 | 2 | Filter 2 | Selector 1<br>Selector 2 | N/A | N/A | N/A |

You can divide each major category to further define the QoS policy. Subsequent sections explain how to obtain information for the categories that are shown in the template.

# QoS Policy Planning (Task Map)

This task map lists the major tasks for planning a QoS policy.

| Task | Description | For Instructions |
|---|---|---|
| 1. Design your network topology to support IPQoS. | Identify the hosts and routers on your network to provide differentiated services. | "How to Prepare a Network for IPQoS" on page 779 |
| 2. Define the classes into which services on your network must be divided. | Examine the types of services and SLAs that are offered by your site, and determine the discrete traffic classes into which these services fall. | "How to Define the Classes for Your QoS Policy" on page 780 |
| 3. Define filters for the classes. | Determine the best ways of separating traffic of a particular class from the network traffic flow. | "How to Define Filters in the QoS Policy" on page 783 |
| 4. Define flow-control rates for measuring traffic as packets leave the IPQoS system. | Determine acceptable flow rates for each class of traffic. | "How to Plan Flow Control" on page 784 |

| Task | Description | For Instructions |
|---|---|---|
| 5. Define DSCPs or user-priority values to be used in the QoS policy. | Plan a scheme to determine the forwarding behavior that is assigned to a traffic flow when the flow is handled by the router or switch. | "How to Plan Forwarding Behavior" on page 786 |
| 6. If applicable, set up a statistics-monitoring plan for traffic flows on the network. | Evaluate the traffic classes to determine which traffic flows must be monitored for accounting or statistical purposes. | "How to Plan for Flow Accounting" on page 789 |

**Note –** The rest of this section explains how to plan the QoS policy of an IPQoS-enabled system. To plan the QoS policy for the Diffserv router, refer to the router documentation and the router manufacturer's web site.

## ▼ How to Prepare a Network for IPQoS

The following procedure lists general planning tasks to do before you create the QoS policy.

1 **Review your network topology. Then, plan a strategy that uses IPQoS systems and Diffserv routers.**

For topology examples, see "Planning the Diffserv Network Topology" on page 774.

2 **Identify the hosts in the topology that require IPQoS or that might become good candidates for IPQoS service.**

3 **Determine which IPQoS-enabled systems could use the same QoS policy.**

For example, if you plan to enable IPQoS on all hosts on the network, identify any hosts that could use the same QoS policy. Each IPQoS-enabled system must have a local QoS policy, which is implemented in its IPQoS configuration file. However, you can create one IPQoS configuration file to be used by a range of systems. You can then copy the configuration file to every system with the same QoS policy requirements.

4 **Review and perform any planning tasks that are required by the Diffserv router on your network.**

Refer to the router documentation and the router manufacturer's web site for details.

## How to Define the Classes for Your QoS Policy

The first step in defining the QoS policy is organizing traffic flows into classes. You do not need to create classes for every type of traffic on a Diffserv network. Moreover, depending on your network topology, you might have to create a different QoS policy for each IPQoS-enabled system.

**Note –** For an overview of classes, see "IPQoS Classes" on page 765.

The next procedure assumes that you have determined which systems on your network are to be IPQoS-enabled, as identified in "How to Prepare a Network for IPQoS" on page 779.

1. **Create a QoS planning table for organizing the QoS policy information.**

   For suggestions, refer to Table 33–1.

2. **Perform the remaining steps for every QoS policy that is on your network.**

3. **Define the classes to be used in the QoS policy.**

   The following questions are a guideline for analyzing network traffic for possible class definitions.

   - **Does your company offer service-level agreements to customers?**

     If yes, then evaluate the relative priority levels of the SLAs that your company offers to customers. The same applications might be offered to customers who are guaranteed different priority levels.

     For example, your company might offer web site hosting to each customer, which indicates that you need to define a class for each customer web site. One SLA might provide a premium web site as one service level. Another SLA might offer a "best-effort" personal web site to discount customers. This factor indicates not only different web site classes but also potentially different per-hop behaviors that are assigned to the web site classes.

   - **Does the IPQoS system offer popular applications that might need flow control?**

     You can improve network performance by enabling IPQoS on servers offering popular applications that generate excessive traffic. Common examples are electronic mail, network news, and FTP. Consider creating separate classes for incoming and outgoing traffic for each service type, where applicable. For example, you might create a `mail-in` class and a `mail-out` class for the QoS policy for a mail server.

   - **Does your network run certain applications that require highest-priority forwarding behaviors?**

     Any critical applications that require highest-priority forwarding behaviors must receive highest priority in the router's queue. Typical examples are streaming video and streaming audio.

Define incoming classes and outgoing classes for these high-priority applications. Then, add the classes to the QoS policies of both the IPQoS-enabled system that serves the applications and the Diffserv router.

- **Does your network experience traffic flows that must be controlled because the flows consume large amounts of bandwidth?**

  Use `netstat`, `snoop`, and other network monitoring utilities to discover the types of traffic that are causing problems on the network. Review the classes that you have created thus far, and then create new classes for any undefined problem traffic category. If you have already defined classes for a category of problem traffic, then define rates for the meter to control the problem traffic.

  Create classes for the problem traffic on every IPQoS-enabled system on the network. Each IPQoS system can then handle any problem traffic by limiting the rate at which the traffic flow is released onto the network. Be sure also to define these problem classes in the QoS policy on the Diffserv router. The router can then queue and schedule the problem flows as configured in its QoS policy.

- **Do you need to obtain statistics on certain types of traffic?**

  A quick review of an SLA can indicate which types of customer traffic require accounting. If your site does offer SLAs, you probably have already created classes for traffic that requires accounting. You might also define classes to enable statistics gathering on traffic flows that you are monitoring. You could also create classes for traffic to which you restrict access for security reasons.

4 **List the classes that you have defined in the QoS planning table you created in Step 1.**

5 **Assign a priority level to each class.**

For example, have priority level 1 represent the highest-priority class, and assign descending-level priorities to the remaining classes. The priority level that you assign is for organizational purposes only. Priority levels that you set in the QoS policy template are not actually used by IPQoS. Moreover, you can assign the same priority to more than one class, if appropriate for your QoS policy.

6 **When you finish defining classes, you next define filters for each class, as explained in "How to Define Filters in the QoS Policy" on page 783.**

**More Information**

## Prioritizing the Classes

As you create classes, you quickly realize which classes have highest priority, medium priority, and best-effort priority. A good scheme for prioritizing classes becomes particularly important when you assign per-hop behaviors to outgoing traffic, as explained in "How to Plan Forwarding Behavior" on page 786.

In addition to assigning a PHB to a class, you can also define a priority selector in a filter for the class. The priority selector is active on the IPQoS-enabled host only. Suppose several classes

with equal rates and identical DSCPs sometimes compete for bandwidth as they leave the IPQoS system. The priority selector in each class can further order the level of service that is given to the otherwise identically valued classes.

## Defining Filters

You create filters to identify packet flows as members of a particular class. Each filter contains selectors, which define the criteria for evaluating a packet flow. The IPQoS-enabled system then uses the criteria in the selectors to extract packets from a traffic flow. The IPQoS system then associates the packets with a class. For an introduction to filters, see "IPQoS Filters" on page 765.

The following table lists the most commonly used selectors. The first five selectors represent the IPQoS 5-tuple, which the IPQoS system uses to identify packets as members of a flow. For a complete list of selectors, see Table 37–1.

**TABLE 33–2** Common IPQoS Selectors

| Name | Definition |
|---|---|
| `saddr` | Source address. |
| `daddr` | Destination address. |
| `sport` | Source port number. You can use a well-known port number, as defined in `/etc/services`, or a user-defined port number. |
| `dport` | Destination port number. |
| `protocol` | IP protocol number or protocol name that is assigned to the traffic flow type in `/etc/protocols`. |
| `ip_version` | Addressing style to use. Use either IPv4 or IPv6. IPv4 is the default. |
| `dsfield` | Contents of the DS field, that is, the DSCP. Use this selector for extracting incoming packets that are already marked with a particular DSCP. |
| `priority` | Priority level that is assigned to the class. For more information, see "How to Define the Classes for Your QoS Policy" on page 780. |
| `user` | Either the UNIX user ID or user name that is used when the upper-level application is executed. |
| `projid` | Project ID that is used when the upper-level application is executed. |
| `direction` | Direction of traffic flow. Value is either `LOCAL_IN`, `LOCAL_OUT`, `FWD_IN`, or `FWD_OUT`. |

**Note –** Be judicious in your choice of selectors. Use only as many selectors as you need to extract packets for a class. The more selectors that you define, the greater the impact on IPQoS performance.

## ▼ How to Define Filters in the QoS Policy

**Before You Begin** Before you can perform the next steps, you should have completed the procedure "How to Define the Classes for Your QoS Policy" on page 780.

1 **Create at least one filter for each class in the QoS planning table that you created in "How to Define the Classes for Your QoS Policy" on page 780.**

Consider creating separate filters for incoming and outgoing traffic for each class, where applicable. For example, add an `ftp-in` filter and an `ftp-out` filter to the QoS policy of an IPQoS-enabled FTP server. You then can define an appropriate `direction` selector in addition to the basic selectors.

2 **Define at least one selector for each filter in a class.**

Use the QoS planning table that was introduced in Table 33–1 to fill in filters for the classes you defined.

**Example 33–1** Defining Filters for FTP Traffic

The next table shows how you would define a filter for outgoing FTP traffic.

| Class | Priority | Filters | Selectors |
|---|---|---|---|
| `ftp-traffic` | 4 | `ftp-out` | `saddr 10.190.17.44` |
| | | | `daddr 10.100.10.53` |
| | | | `sport 21` |
| | | | `direction LOCAL_OUT` |

**See Also**

- To define a flow-control scheme, refer to "How to Plan Flow Control" on page 784.
- To define forwarding behaviors for flows as the flows return to the network stream, refer to "How to Plan Forwarding Behavior" on page 786.
- To plan for flow accounting of certain types of traffic, refer to "How to Plan for Flow Accounting" on page 789.
- To add more classes to the QoS policy, refer to "How to Define the Classes for Your QoS Policy" on page 780.

- To add more filters to the QoS policy, refer to "How to Define Filters in the QoS Policy" on page 783.

## How to Plan Flow Control

Flow control involves measuring traffic flow for a class and then releasing packets onto the network at a defined rate. When you plan flow control, you define parameters to be used by the IPQoS metering modules. The meters determine the rate at which traffic is released onto the network. For an introduction to the metering modules, see "Meter (`tokenmt` and `tswtclmt`) Overview" on page 766.

The next procedure assumes that you have defined filters and selectors, as described in "How to Define Filters in the QoS Policy" on page 783.

1. **Determine the maximum bandwidth for your network.**

2. **Review any SLAs that are supported on your network. Identify customers and the type of service that is guaranteed to each customer.**

   To guarantee a certain level of service, you might need to meter certain traffic classes that are generated by the customer.

3. **Review the list of classes that you created in "How to Define the Classes for Your QoS Policy" on page 780.**

   Determine if any classes other than those classes that are associated with SLAs need to be metered.

   Suppose the IPQoS system runs an application that generates a high level of traffic. After you classify the application's traffic, meter the flows to control the rate at which the packets of the flow return to the network.

   ---

   **Note –** Not all classes need to be metered. Remember this guideline as you review your list of classes.

   ---

4. **Determine which filters in each class select traffic that needs flow control. Then, refine your list of classes that require metering.**

   Classes that have more than one filter might require metering for only one filter. Suppose that you define filters for incoming and outgoing traffic of a certain class. You might conclude that only traffic in one direction requires flow control.

5. **Choose a meter module for each class to be flow controlled.**

   Add the module name to the meter column in your QoS planning table.

**6 Add the rates for each class to be metered to the organizational table.**

If you use the `tokenmt` module, you need to define the following rates in bits per second:

- Committed rate
- Peak rate

If these rates are sufficient to meter a particular class, you can define only the committed rate and the committed burst for `tokenmt`.

If needed, you can also define the following rates:

- Committed burst
- Peak burst

For a complete definition of `tokenmt` rates, refer to "Configuring `tokenmt` as a Two-Rate Meter" on page 839. You can also find more detailed information in the `tokenmt`(7ipp) man page.

If you use the `tswtclmt` module, you need to define the following rates in bits per second.

- Committed rate
- Peak rate

You can also define the window size in milliseconds. These rates are defined in "`tswtclmt` Metering Module" on page 840 and in the `twstclmt`(7ipp) man page.

**7 Add traffic conformance outcomes for the metered traffic.**

The outcomes for both metering modules are green, red, and yellow. Add to your QoS organizational table the traffic conformance outcomes that apply to the rates you define. Outcomes for the meters are fully explained in "Meter Module" on page 838.

You need to determine what action should be taken on traffic that conforms, or does not conform, to the committed rate. Often, but not always, this action is to mark the packet header with a per-hop behavior. One acceptable action for green-level traffic could be to continue processing while traffic flows do not exceed the committed rate. Another action could be to drop packets of the class if flows exceed peak rate.

**Example 33–2** Defining Meters

The next table shows meter entries for a class of email traffic. The network on which the IPQoS system is located has a total bandwidth of 100 Mbits/sec, or 10000000 bits per second. The QoS policy assigns a low priority to the email class. This class also receives best-effort forwarding behavior.

| Class | Priority | Filter | Selector | Rate |
|---|---|---|---|---|
| email | 8 | mail_in | daddr10.50.50.5<br>dport imap<br>direction LOCAL_IN | |
| email | 8 | mail_out | saddr10.50.50.5<br>sport imap<br>direction LOCAL_OUT | meter=tokenmt<br>committed rate=5000000<br>committed burst =5000000<br>peak rate =10000000<br>peak burst=1000000<br>green precedence=continue processing<br>yellow precedence=mark yellow PHB<br>red precedence=drop |

**See Also**

- To define forwarding behaviors for flows as the packets return to the network stream, refer to "How to Plan Forwarding Behavior" on page 786.
- To plan for flow accounting of certain types of traffic, refer to "How to Plan for Flow Accounting" on page 789.
- To add more classes to the QoS policy, refer to "How to Define the Classes for Your QoS Policy" on page 780.
- To add more filters to the QoS policy, refer to "How to Define Filters in the QoS Policy" on page 783.
- To define another flow-control scheme, refer to "How to Plan Flow Control" on page 784.
- To create an IPQoS configuration file, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 798.

## ▼ How to Plan Forwarding Behavior

Forwarding behavior determines the priority and drop precedence of traffic flows that are about to be forwarded to the network. You can choose two major forwarding behaviors: prioritize the flows of a class in relationship to other traffic classes or drop the flows entirely.

The Diffserv model uses the marker to assign the chosen forwarding behavior to traffic flows. IPQoS offers the following marker modules.

- `dscpmk` – Used to mark the DS field of an IP packet with a DSCP

- `dlcosmk` – Used to mark the VLAN tag of a datagram with a class-of-service (CoS) value

---

**Note –** The suggestions in this section refer specifically to IP packets. If your IPQoS system includes a VLAN device, you can use the `dlcosmk` marker to mark forwarding behaviors for datagrams. For more information, refer to "Using the `dlcosmk` Marker With VLAN Devices" on page 843.

---

To prioritize IP traffic, you need to assign a DSCP to each packet. The `dscpmk` marker marks the DS field of the packet with the DSCP. You choose the DSCP for a class from a group of well-known codepoints that are associated with the forwarding behavior type. These well-known codepoints are 46 (101110) for the EF PHB and a range of codepoints for the AF PHB. For overview information on DSCP and forwarding, refer to "Traffic Forwarding on an IPQoS-Enabled Network" on page 769.

**Before You Begin** The next steps assume that you have defined classes and filters for the QoS policy. Though you often use the meter with the marker to control traffic, you can use the marker alone to define a forwarding behavior.

1. **Review the classes that you have created thus far and the priorities that you have assigned to each class.**

   Not all traffic classes need to be marked.

2. **Assign the EF per-hop behavior to the class with the highest priority.**

   The EF PHB guarantees that packets with the EF DSCP 46 (101110) are released onto the network before packets with any AF PHBs. Use the EF PHB for your highest-priority traffic. For more information about EF, refer to "Expedited Forwarding (EF) PHB" on page 841.

3. **Assign forwarding behaviors to classes that have traffic to be metered.**

4. **Assign DS codepoints to the remaining classes in agreement with the priorities that you have assigned to the classes.**

**Example 33–3** QoS Policy for a Games Application

Traffic is generally metered for the following reasons:

- An SLA guarantees packets of this class greater service or lesser service when the network is heavily used.
- A class with a lower priority might have a tendency to flood the network.

You use the marker with the meter to provide differentiated services and bandwidth management to these classes. For example, the following table shows a portion of a QoS policy. This policy defines a class for a popular games application that generates a high level of traffic.

| Class | Priority | Filter | Selector | Rate | Forwarding? |
|---|---|---|---|---|---|
| games_app | 9 | games_in | sport 6080 | N/A | N/A |
| games_app | 9 | games_out | dport 6081 | meter=tokenmt<br>committed rate=5000000<br>committed burst =5000000<br>peak rate =10000000<br>peak burst=15000000<br>green precedence=continue processing<br>yellow precedence=mark yellow PHB<br>red precedence=drop | green =AF31<br>yellow=AF42<br>red=drop |

The forwarding behaviors assign low-priority DSCPs to games_app traffic that conforms to its committed rate or is under the peak rate. When games_app traffic exceeds peak rate, the QoS policy indicates that packets from games_app are to be dropped. All AF codepoints are listed in Table 37–2.

**See Also**

- To plan for flow accounting of certain types of traffic, refer to "How to Plan for Flow Accounting" on page 789.
- To add more classes to the QoS policy, refer to "How to Define the Classes for Your QoS Policy" on page 780.
- To add more filters to the QoS policy, refer to "How to Define Filters in the QoS Policy" on page 783.
- To define a flow-control scheme, refer to "How to Plan Flow Control" on page 784.
- To define additional forwarding behaviors for flows as the packets return to the network stream, refer to "How to Plan Forwarding Behavior" on page 786.
- To create an IPQoS configuration file, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 798.

## ▼ How to Plan for Flow Accounting

You use the IPQoS `flowacct` module to track traffic flows for billing or network management purposes. Use the following procedure to determine if your QoS policy should include flow accounting.

1 **Does your company offer SLAs to customers?**

If the answer is yes, then you should use flow accounting. Review the SLAs to determine what types of network traffic your company wants to bill customers for. Then, review your QoS policy to determine which classes select traffic to be billed.

2 **Are there applications that might need monitoring or testing to avoid network problems?**

If the answer is yes, consider using flow accounting to observe the behavior of these applications. Review your QoS policy to determine the classes that you have assigned to traffic that requires monitoring.

3 **Mark Y in the flow-accounting column for each class that requires flow accounting in your QoS planning table.**

**See Also**

- To add more classes to the QoS policy, refer to "How to Define the Classes for Your QoS Policy" on page 780.
- To add more filters to the QoS policy, refer to "How to Define Filters in the QoS Policy" on page 783.
- To define a flow-control scheme, refer to "How to Plan Flow Control" on page 784.
- To define forwarding behaviors for flows as the packets return to the network stream, refer to "How to Plan Forwarding Behavior" on page 786.
- To plan for additional flow accounting of certain types of traffic, refer to "How to Plan for Flow Accounting" on page 789.
- To create the IPQoS configuration file, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 798.

# Introducing the IPQoS Configuration Example

Tasks in the remaining chapters of the guide use the example IPQoS configuration that is introduced in this section. The example shows the differentiated services solution on the public intranet of BigISP, a fictitious service provider. BigISP offers services to large companies that reach BigISP through leased lines. Individuals who dial in from modems can also buy services from BigISP.

# IPQoS Topology

The following figure shows the network topology that is used for BigISP's public intranet.

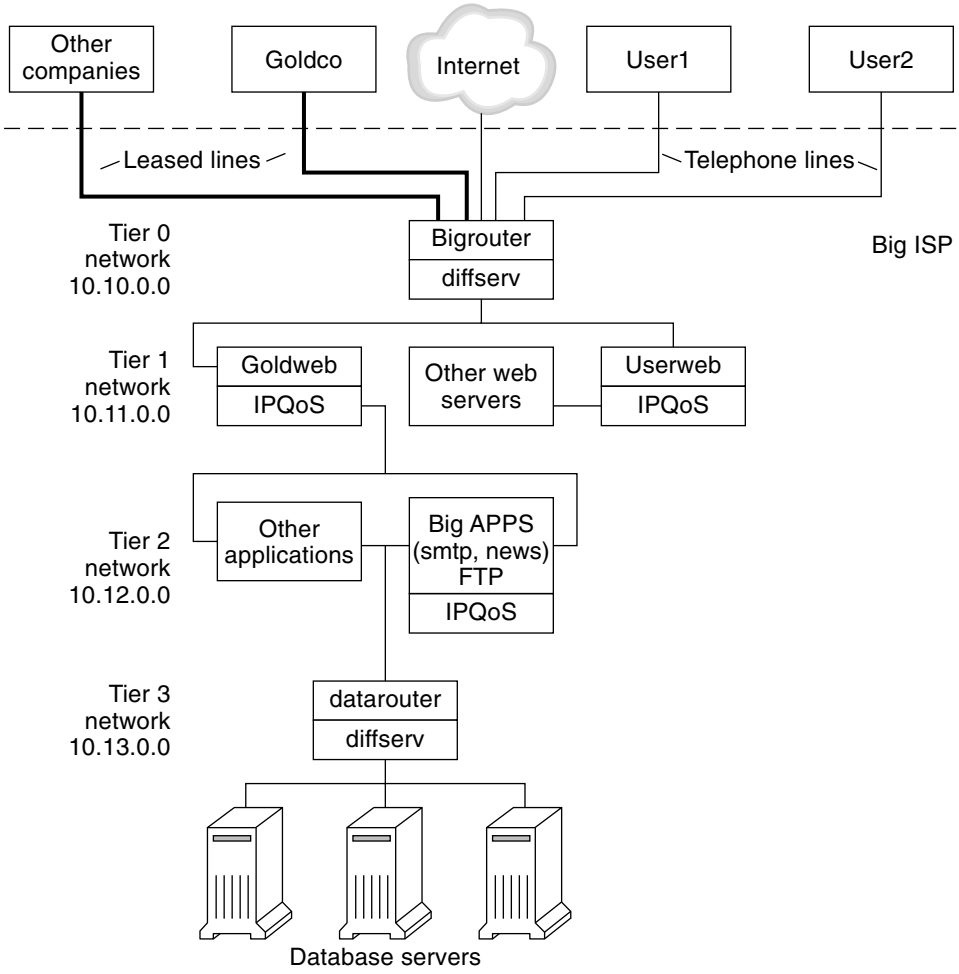



**FIGURE 33–4** IPQoS Example Topology

BigISP has implemented these four tiers in its public intranet:

- **Tier 0** – Network `10.10.0.0` includes a large Diffserv router that is called `Bigrouter`, which has both external and internal interfaces. Several companies, including a large organization that is called Goldco, have rented leased-line services that terminate at `Bigrouter`. Tier 0 also handles individual customers who call over telephone lines or ISDN.

- **Tier 1** – Network `10.11.0.0` provides web services. The `Goldweb` server hosts the web site which was purchased by Goldco as part of the premium service that Goldco has purchased from BigISP. The server `Userweb` hosts small web sites that were purchased by individual customers. Both `Goldweb` and `Userweb` are IPQoS enabled.
- **Tier 2** – Network `10.12.0.0` provides applications for all customers to use. `BigAPPS`, one of the application servers, is IPQoS-enabled. `BigAPPS` provides SMTP, News, and FTP services.
- **Tier 3** – Network `10.13.0.0` houses large database servers. Access to Tier 3 is controlled by `datarouter`, a Diffserv router.

◆ ◆ ◆ CHAPTER 34

# Creating the IPQoS Configuration File (Tasks)

This chapter shows how to create IPQoS configuration files. Topics that are covered in the chapter include the following.


- "Defining a QoS Policy in the IPQoS Configuration File (Task Map)" on page 793
- "Tools for Creating a QoS Policy" on page 794
- "Creating IPQoS Configuration Files for Web Servers" on page 795
- "Creating an IPQoS Configuration File for an Application Server" on page 808
- "Providing Differentiated Services on a Router" on page 818


This chapter assumes that you have defined a complete QoS policy, and you are ready to use this policy as the basis for the IPQoS configuration file. For instructions on QoS policy planning, refer to "Planning the Quality-of-Service Policy" on page 777.

## Defining a QoS Policy in the IPQoS Configuration File (Task Map)

This task map lists the general tasks for creating an IPQoS configuration file.

| Task | Description | For Instructions |
|---|---|---|
| 1. Plan your IPQoS-enabled network configuration. | Decide which systems on the local network should become IPQoS enabled. | "How to Prepare a Network for IPQoS" on page 779 |
| 2. Plan the QoS policy for IPQoS systems on your network. | Identify traffic flows as distinct classes of service. Then, determine which flows require traffic management. | "Planning the Quality-of-Service Policy" on page 777 |

| Task | Description | For Instructions |
|---|---|---|
| 3. Create the IPQoS configuration file and define its first action. | Create the IPQoS file, invoke the IP classifier, and define a class for processing. | "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 798 |
| 4. Create filters for a class. | Add the filters that govern which traffic is selected and organized into a class. | "How to Define Filters in the IPQoS Configuration File" on page 800 |
| 5. Add more classes and filters to the IPQoS configuration file. | Create more classes and filters to be processed by the IP classifier. | "How to Create an IPQoS Configuration File for a Best-Effort Web Server" on page 806 |
| 6. Add an `action` statement with parameters that configure the metering modules. | If the QoS policy calls for flow control, assign flow-control rates and conformance levels to the meter. | "How to Configure Flow Control in the IPQoS Configuration File" on page 815 |
| 7. Add an `action` statement with parameters that configure the marker. | If the QoS policy calls for differentiated forwarding behaviors, define how traffic classes are to be forwarded. | "How to Define Traffic Forwarding in the IPQoS Configuration File" on page 801 |
| 8. Add an `action` statement with parameters that configure the flow-accounting module. | If the QoS policy calls for statistics gathering on traffic flows, define how accounting statistics are to be gathered. | "How to Enable Accounting for a Class in the IPQoS Configuration File" on page 804 |
| 9. Apply the IPQoS configuration file. | Add the content of a specified IPQoS configuration file into the appropriate kernel modules. | "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 822 |
| 10. Configure forwarding behaviors in the router files. | If any IPQoS configuration files on the network define forwarding behaviors, add the resulting DSCPs to the appropriate scheduling files on the router. | "How to Configure a Router on an IPQoS-Enabled Network" on page 819 |

# Tools for Creating a QoS Policy

The QoS policy for your network resides in the IPQoS configuration file. You create this configuration file with a text editor. Then, you provide the file as an argument to `ipqosconf`, the IPQoS configuration utility. When you instruct `ipqosconf` to apply the policy that is defined in your configuration file, the policy is written into the kernel IPQoS system. For detailed information about the `ipqosconf` command, refer to the `ipqosconf`(1M) man page. For instructions on the use of `ipqosconf`, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 822.

## Basic IPQoS Configuration File

An IPQoS configuration file consists of a tree of `action` statements that implement the QoS policy that you defined in "Planning the Quality-of-Service Policy" on page 777. The IPQoS configuration file configures the IPQoS modules. Each action statement contains a set of *classes*, *filters*, or *parameters* to be processed by the module that is called in the action statement.

For the complete syntax of the IPQoS configuration file, refer to Example 37–3 and the `ipqosconf`(1M) man page.

### Configuring the IPQoS Example Topology

The tasks in this chapter explain how to create IPQoS configuration files for three IPQoS-enabled systems. These systems are part of the network topology of the company BigISP, which was introduced in Figure 33–4.

- `Goldweb` – A web server that hosts web sites for customers who have purchased premium-level SLAs
- `Userweb` – A less-powerful web server that hosts personal web sites for home users who have purchased "best-effort" SLAs
- `BigAPPS` – An application server that serves mail, network news, and FTP to both gold-level and best-effort customers

These three configuration files illustrate the most common IPQoS configurations. You might use the sample files that are shown in the next section as templates for your own IPQoS implementation.

# Creating IPQoS Configuration Files for Web Servers

This section introduces the IPQoS configuration file by showing how to create a configuration for a premium web server. The section then shows how to configure a completely different level of service in another configuration file for a server that hosts personal web sites. Both servers are part of the network example that is shown in Figure 33–4.

The following configuration file defines IPQoS activities for the `Goldweb` server. This server hosts the web site for Goldco, the company that has purchased a premium SLA.

**EXAMPLE 34–1** Sample IPQoS Configuration File for a Premium Web Server

```
fmt_version 1.0

action {
    module ipgpc
    name ipgpc.classify
```

**EXAMPLE 34–1** Sample IPQoS Configuration File for a Premium Web Server *(Continued)*

```
    params {
        global_stats TRUE
    }
    class {
        name goldweb
        next_action markAF11
        enable_stats FALSE
    }
    class {
        name video
        next_action markEF
        enable_stats FALSE
    }
    filter {
        name webout
        sport 80
        direction LOCAL_OUT
        class goldweb
    }
    filter {
        name videoout
        sport videosrv
        direction LOCAL_OUT
        class video
    }
}
action {
    module dscpmk
    name markAF11
    params {
        global_stats FALSE
        dscp_map{0-63:10}
        next_action continue
    }
}
action {
    module dscpmk
    name markEF
    params {
        global_stats TRUE
        dscp_map{0-63:46}
        next_action acct
    }
}
action {
    module flowacct
```

**EXAMPLE 34–1** Sample IPQoS Configuration File for a Premium Web Server *(Continued)*

```
    name acct
    params {
        enable_stats TRUE
        timer 10000
        timeout 10000
        max_limit 2048
    }
}
```

The following configuration file defines IPQoS activities on `Userweb`. This server hosts web sites for individuals with low-priced, or *best-effort*, SLAs. This level of service guarantees the best service that can be delivered to best-effort customers after the IPQoS system handles traffic from customers with more expensive SLAs.

**EXAMPLE 34–2** Sample Configuration for a Best-Effort Web Server

```
fmt_version 1.0

action {
    module ipgpc
    name ipgpc.classify
    params {
        global_stats TRUE
    }
    class {
        name Userweb
        next_action markAF12
        enable_stats FALSE
    }
    filter {
        name webout
        sport 80
        direction LOCAL_OUT
        class Userweb
   }
}
action {
    module dscpmk
    name markAF12
    params {
        global_stats FALSE
        dscp_map{0-63:12}
        next_action continue
    }
}
```

## How to Create the IPQoS Configuration File and Define Traffic Classes

You can create your first IPQoS configuration file in whatever directory is easiest for you to maintain. The tasks in this chapter use the directory `/var/ipqos` as the location for IPQoS configuration files. The next procedure builds the initial segment of the IPQoS configuration file that is introduced in Example 34–1.

---

**Note –** As you create the IPQoS configuration file, be very careful to start and end each `action` statement and clause with curly braces ({ }). For an example of the use of braces, see Example 34–1.

---

1 **Log in to the premium web server, and create a new IPQoS configuration file with a `.qos` extension.**

Every IPQoS configuration file must start with the version number `fmt_version 1.0` as its first uncommented line.

2 **Follow the opening parameter with the initial `action` statement, which configures the generic IP classifier `ipgpc`.**

This initial action begins the tree of `action` statements that compose the IPQoS configuration file. For example, the `/var/ipqos/Goldweb.qos` file begins with the initial `action` statement to call the `ipgpc` classifier.

```
fmt_version 1.0

action {
    module ipgpc
    name ipgpc.classify
```

| | |
|---|---|
| `fmt_version 1.0` | Begins the IPQoS configuration file. |
| `action {` | Begins the action statement. |
| `module ipgpc` | Configures the `ipgpc` classifier as the first action in the configuration file. |
| `name ipgpc.classify` | Defines the name of the classifier `action` statement, which must always be `ipgpc.classify`. |

For detailed syntactical information about `action` statements, refer to "`action` Statement" on page 849 and the `ipqosconf`(1M) man page.

3 **Add a `params` clause with the statistics parameter `global_stats`.**

```
params {
         global_stats TRUE
   }
```

The parameter `global_stats TRUE` in the`ipgpc.classify` statement enables statistics gathering for that action. `global_stats TRUE` also enables per-class statistics gathering wherever a class clause definition specifies `enable_stats TRUE`.

Turning on statistics impacts performance. You might want to gather statistics on a new IPQoS configuration file to verify that IPQoS works properly. Later, you can turn off statistics collection by changing the argument to `global_stats` to `FALSE`.

Global statistics are but one type of parameter you can define in a `params` clause. For syntactical and other details about `params` clauses, refer to "`params` Clause" on page 851 and the `ipqosconf`(1M) man page.

**4 Define a class that identifies traffic that is bound for the premium server.**

```
class {
        name goldweb
        next_action markAF11
        enable_stats FALSE
    }
```

This statement is called a *class clause*. A `class` clause has the following contents.

| | |
|---|---|
| name goldweb | Creates the class `goldweb` to identify traffic that is bound for the `Goldweb` server. |
| next_action markAF11 | Instructs the `ipgpc` module to pass packets of the `goldweb` class to the `markAF11` action statement. The `markAF11` action statement calls the `dscpmk` marker. |
| enable_stats FALSE | Enables statistics taking for the `goldweb` class. However, because the value of `enable_stats` is `FALSE`, statistics for this class are not turned on. |

For detailed information about the syntax of the `class` clause, see "`class` Clause" on page 850 and the `ipqosconf`(1M) man page.

**5 Define a class that identifies an application that must have highest-priority forwarding.**

```
class {
        name video
        next_action markEF
        enable_stats FALSE
    }
```

| | |
|---|---|
| `name video` | Creates the class `video` to identify streaming video traffic that is outgoing from the `Goldweb` server. |
| `next_action markEF` | Instructs the `ipgpc` module to pass packets of the `video` class to the `markEF` statement after `ipgpc` completes processing. The `markEF` statement calls the `dscpmk` marker. |

`enable_stats FALSE` Enables statistics collection for the `video` class. However, because the value of `enable_stats` is `FALSE`, statistics collection for this class is not turned on.

**See Also**
- To define filters for the class you just created, refer to "How to Define Filters in the IPQoS Configuration File" on page 800.
- To create another class clause for the configuration file, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 798.

## ▼ How to Define Filters in the IPQoS Configuration File

The next procedure shows how to define filters for a class in the IPQoS configuration file.

**Before You Begin** The procedure assumes that you have already started file creation and have defined classes. The steps continue building the `/var/ipqos/Goldweb.qos` file that is created in "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 798.

---

**Note –** As you create the IPQoS configuration file, be very careful to start and end each `class` clause and each `filter` clause with curly braces ({ }). For an example of the use of braces, use Example 34–1.

---

**1 Open the IPQoS configuration file, and locate the end of the last class that you defined.**

For example, on the IPQoS-enabled server `Goldweb`, you would start after the following `class` clause in `/var/ipqos/Goldweb.qos`:

```
class {
        name video
        next_action markEF
        enable_stats FALSE
    }
```

**2 Define a `filter` clause to select outgoing traffic from the IPQoS system.**

```
    filter {
        name webout
        sport 80
        direction LOCAL_OUT
        class goldweb
    }
```

`name webout` Gives the name `webout` to the filter.

`sport 80` Selects traffic with a source port of 80, the well-known port for HTTP (web) traffic.

| | |
|---|---|
| `direction LOCAL_OUT` | Further selects traffic that is outgoing from the local system. |
| `class goldweb` | Identifies the class to which the filter belongs, in this instance, class `goldweb`. |

For syntactical and detailed information about the `filter` clause in the IPQoS configuration file, refer to "`filter` Clause" on page 850.

**3 Define a `filter` clause to select streaming video traffic on the IPQoS system.**

```
filter {
    name videoout
    sport videosrv
    direction LOCAL_OUT
    class video
}
```

| | |
|---|---|
| name videoout | Gives the name `videoout` to the filter. |
| `sport videosrv` | Selects traffic with a source port of `videosrv`, a previously defined port for the streaming video application on this system. |
| `direction LOCAL_OUT` | Further selects traffic that is outgoing from the local system. |
| `class video` | Identifies the class to which the filter belongs, in this instance, class `video`. |

**See Also**

- To define forwarding behaviors for the marker modules, refer to "How to Define Traffic Forwarding in the IPQoS Configuration File" on page 801.
- To define flow-control parameters for the metering modules, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 815.
- To activate the IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 822.
- To define additional filters, refer to "How to Define Filters in the IPQoS Configuration File" on page 800.
- To create classes for traffic flows from applications, refer to "How to Configure the IPQoS Configuration File for an Application Server" on page 811.

## ▼ How to Define Traffic Forwarding in the IPQoS Configuration File

The next procedure shows how to define traffic forwarding by adding per-hop behaviors for a class into the IPQoS configuration file.

**Before You Begin** The procedure assumes that you have an existing IPQoS configuration file with already defined classes and already defined filters. The steps continue building the `/var/ipqos/Goldweb.qos` file from Example 34–1.

---

**Note –** The procedure shows how to configure traffic forwarding by using the `dscpmk` marker module. For information about traffic forwarding on VLAN systems by using the `dlclosmk` marker, refer to "Using the `dlcosmk` Marker With VLAN Devices" on page 843.

---

**1 Open the IPQoS configuration file, and locate the end of the last filter you defined.**

For example, on the IPQoS-enabled server `Goldweb`, you would start after the following `filter` clause in `/var/ipqos/Goldweb.qos`:

```
filter {
        name videoout
        sport videosrv
        direction LOCAL_OUT
        class video
    }
}
```

Note that this `filter` clause is at the end of the `ipgpc` classifier `action` statement. Therefore, you need a closing brace to terminate the filter and a second closing brace to terminate the `action` statement.

**2 Invoke the marker with the following `action` statement.**

```
action {
    module dscpmk
    name markAF11
```

module dscpmk — Calls the marker module `dscpmk`.

name markAF11 — Gives the name `markAF11` to the `action` statement.

The previously defined class `goldweb` includes a `next_action markAF11` statement. This statement sends traffic flows to the `markAF11` action statement after the classifier concludes processing.

**3 Define actions for the marker to take on the traffic flow.**

```
    params {
        global_stats FALSE
        dscp_map{0-63:10}
        next_action continue
    }
}
```

| | |
|---|---|
| global_stats FALSE | Enables statistics collection for the `markAF11` marker `action` statement. However, because the value of `enable_stats` is `FALSE`, statistics are not collected. |
| dscp_map{0–63:10} | Assigns a DSCP of `10` to the packet headers of the traffic class `goldweb`, which is currently being processed by the marker. |
| next_action continue | Indicates that no further processing is required on packets of the traffic class `goldweb`, and that these packets can return to the network stream. |

The DSCP of `10` instructs the marker to set all entries in the `dscp` map to the decimal value 10 (binary 001010). This codepoint indicates that packets of the `goldweb` traffic class are subject to the AF11 per-hop behavior. AF11 guarantees that all packets with the DSCP of `10` receive a low-drop, high-priority service. Thus, outgoing traffic for premium customers on `Goldweb` is given the highest priority that is available for the Assured Forwarding (AF) PHB. For a table of possible DSCPs for AF, refer to Table 37–2.

**4 Start another marker `action` statement.**

```
action {
    module dscpmk
    name markEF
```

| | |
|---|---|
| module dscpmk | Calls the marker module `dscpmk`. |
| `name markEF` | Gives the name `markEF` to the `action` statement. |

**5 Define actions for the marker to take on the traffic flow.**

```
    params {
        global_stats TRUE
        dscp_map{0-63:46}
        next_action acct
    }
}
```

| | |
|---|---|
| `global_stats TRUE` | Enables statistics collection on class `video`, which selects streaming video packets. |
| `dscp_map{0–63:46}` | Assigns a DSCP of `46` to the packet headers of the traffic class `video`, which is currently being processed by the marker. |
| `next_action acct` | Instructs the `dscpmk` module to pass packets of the class `video` to the `acct action` statement after `dscpmk` completes processing. The `acct action` statement invokes the `flowacct` module. |

The DSCP of `46` instructs the `dscpmk` module to set all entries in the `dscp` map to the decimal value 46 (binary 101110) in the DS field. This codepoint indicates that packets of the `video` traffic class are subject to the Expedited Forwarding (EF) per-hop behavior.

**Note –** The recommended codepoint for EF is 46 (binary 101110). Other DSCPs assign AF PHBs to a packet.

The EF PHB guarantees that packets with the DSCP of 46 are given the highest precedence by IPQoS and Diffserv-aware systems. Streaming applications require highest-priority service, which is the rationale behind assigning to streaming applications the EF PHBs in the QoS policy. For more details about the expedited forwarding PHB, refer to "Expedited Forwarding (EF) PHB" on page 841.

6 **Add the DSCPs that you have just created to the appropriate files on the Diffserv router.**

For more information, refer to "How to Configure a Router on an IPQoS-Enabled Network" on page 819.

**See Also**

- To start gathering flow-accounting statistics on traffic flows, refer to "How to Enable Accounting for a Class in the IPQoS Configuration File" on page 804.
- To define forwarding behaviors for the marker modules, refer to "How to Define Traffic Forwarding in the IPQoS Configuration File" on page 801.
- To define flow-control parameters for the metering modules, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 815.
- To activate the IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 822.
- To define additional filters, refer to "How to Define Filters in the IPQoS Configuration File" on page 800.
- To create classes for traffic flows from applications, refer to "How to Configure the IPQoS Configuration File for an Application Server" on page 811.

## ▼ How to Enable Accounting for a Class in the IPQoS Configuration File

The next procedure shows how to enable accounting on a traffic class in the IPQoS configuration file. The procedure shows how to define flow accounting for the `video` class, which is introduced in "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 798. This class selects streaming video traffic, which must be billed as part of a premium customer's SLA.

**Before You Begin** The procedure assumes that you have an existing IPQoS configuration file with already defined classes, filters, metering actions, if appropriate, and marking actions, if appropriate. The steps continue building the `/var/ipqos/Goldweb.qos` file from Example 34–1.

1 **Open the IPQoS configuration file, and locate the end of the last `action` statement you defined.**

For example, on the IPQoS-enabled server `Goldweb`, you would start after the following `markEF` `action` statement in `/var/ipqos/Goldweb.qos`.

```
action {
    module dscpmk
    name markEF
    params {
        global_stats TRUE
        dscp_map{0-63:46}
        next_action acct
    }
}
```

2 **Begin an `action` statement that calls flow accounting.**

```
action {
    module flowacct
    name acct
```

| | |
|---|---|
| `module flowacct` | Invokes the flow-accounting module `flowacct`. |
| `name acct` | Gives the name `acct` to the `action` statement |

3 **Define a `params` clause to control accounting on the traffic class.**

```
params {
        global_stats TRUE
        timer 10000
        timeout 10000
        max_limit 2048
        next_action continue
    }
}
```

| | |
|---|---|
| `global_stats TRUE` | Enables statistics collection on the class `video`, which selects streaming video packets. |
| `timer 10000` | Specifies the duration of the interval, in milliseconds, when the flow table is scanned for timed-out flows. In this parameter, that interval is 10000 milliseconds. |
| `timeout 10000` | Specifies the minimum interval time out value. A flow "times out" when packets for the flow are not seen during a time out interval. In this parameter, packets time out after 10000 milliseconds. |
| `max_limit 2048` | Sets the maximum number of active flow records in the flow table for this action instance. |
| `next_action continue` | Indicates that no further processing is required on packets of the traffic class `video`, and that these packets can return to the network |

stream.

The `flowacct` module gathers statistical information on packet flows of a particular class until a specified `timeout` value is reached.

**See Also**
- To configure per-hop behaviors on a router, refer to "How to Configure a Router on an IPQoS-Enabled Network" on page 819.
- To activate the IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 822.
- To create classes for traffic flows from applications, refer to "How to Configure the IPQoS Configuration File for an Application Server" on page 811.

## ▼ How to Create an IPQoS Configuration File for a Best-Effort Web Server

The IPQoS configuration file for a best-effort web server differs slightly from an IPQoS configuration file for a premium web server. As an example, the procedure uses the configuration file from Example 34–2.

**1 Log in to the best-effort web server.**

**2 Create a new IPQoS configuration file with a `.qos` extension.**

```
fmt_vesion 1.0
action {
    module ipgpc
    name ipgpc.classify
    params {
        global_stats TRUE
   }
```

The `/var/ipqos/userweb.qos` file must begin with the partial `action` statement to invoke the `ipgpc` classifier. In addition, the `action` statement also has a params clause to turn on statistics collection. For an explanation of this `action` statement, see "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 798.

**3 Define a class that identifies traffic that is bound for the best-effort web server.**

```
class {
        name userweb
        next_action markAF12
        enable_stats FALSE
    }
```

| | |
|---|---|
| `name userweb` | Creates a class that is called `userweb` for forwarding web traffic from users. |
| `next_action markAF1` | Instructs the `ipgpc` module to pass packets of the `userweb` class to the `markAF12` `action` statement after `ipgpc` completes processing. The `markAF12` `action` statement invokes the `dscpmk` marker. |
| `enable_stats FALSE` | Enables statistics collection for the `userweb` class. However, because the value of `enable_stats` is `FALSE`, statistics collection for this class does not occur. |

For an explanation of the `class` clause task, see "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 798.

**4 Define a `filter` clause to select traffic flows for the `userweb` class.**

```
    filter {
        name webout
        sport 80
        direction LOCAL_OUT
        class userweb
    }
}
```

| | |
|---|---|
| `name webout` | Gives the name `webout` to the filter. |
| `sport 80` | Selects traffic with a source port of 80, the well-known port for HTTP (web) traffic. |
| `direction LOCAL_OUT` | Further selects traffic that is outgoing from the local system. |
| class userweb | Identifies the class to which the filter belongs, in this instance, class `userweb`. |

For an explanation of the `filter` clause task, see "How to Define Filters in the IPQoS Configuration File" on page 800.

**5 Begin the `action` statement to invoke the `dscpmk` marker.**

```
action {
    module dscpmk
    name markAF12
```

| | |
|---|---|
| `module dscpmk` | Invokes the marker module `dscpmk`. |
| `name markAF12` | Gives the name `markAF12` to the `action` statement. |

The previously defined class `userweb` includes a `next_action` `markAF12` statement. This statement sends traffic flows to the `markAF12` `action` statement after the classifier concludes processing.

6 **Define parameters for the marker to use for processing the traffic flow.**

```
    params {
        global_stats FALSE
        dscp_map{0-63:12}
        next_action continue
    }
}
```

| | |
|---|---|
| `global_stats FALSE` | Enables statistics collection for the `markAF12` marker `action` statement. However, because the value of `enable_stats` is `FALSE`, statistics collection does not occur. |
| `dscp_map{0–63:12}` | Assigns a DSCP of `12` to the packet headers of the traffic class `userweb`, which is currently being processed by the marker. |
| next_action continue | Indicates that no further processing is required on packets of the traffic class `userweb`, and that these packets can return to the network stream. |

The DSCP of `12` instructs the marker to set all entries in the `dscp` map to the decimal value 12 (binary 001100). This codepoint indicates that packets of the `userweb` traffic class are subject to the AF12 per-hop behavior. AF12 guarantees that all packets with the DSCP of `12` in the DS field receive a medium-drop, high-priority service.

7 **When you complete the IPQoS configuration file, apply the configuration.**

**See Also**
- To add classes and other configuration for traffic flows from applications, refer to "How to Configure the IPQoS Configuration File for an Application Server" on page 811.
- To configure per-hop behaviors on a router, refer to "How to Configure a Router on an IPQoS-Enabled Network" on page 819.
- To activate your IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 822.

# Creating an IPQoS Configuration File for an Application Server

This section explains how to create a configuration file for an application server that provides major applications to customers. The procedure uses as its example the `BigAPPS` server from Figure 33–4.

The following configuration file defines IPQoS activities for the `BigAPPS` server. This server hosts FTP, electronic mail (SMTP), and network news (NNTP) for customers.

**EXAMPLE 34–3** Sample IPQoS Configuration File for an Application Server

```
fmt_version 1.0

action {
    module ipgpc
    name ipgpc.classify
    params {
        global_stats TRUE
    }
    class {
        name smtp
        enable_stats FALSE
        next_action markAF13
    }
    class {
        name news
        next_action markAF21
    }
    class {
        name ftp
        next_action meterftp
    }
    filter {
        name smtpout
        sport smtp
        class smtp
    }
    filter {
        name newsout
        sport nntp
        class news
    }
    filter {
        name ftpout
        sport ftp
        class ftp
    }
   filter {
        name ftpdata
        sport ftp-data
        class ftp
    }
}
action {
    module dscpmk
    name markAF13
```

**EXAMPLE 34–3** Sample IPQoS Configuration File for an Application Server *(Continued)*

```
    params {
        global_stats FALSE
        dscp_map{0-63:14}
        next_action continue
    }
}
action {
    module dscpmk
    name markAF21
    params {
        global_stats FALSE
        dscp_map{0-63:18}
        next_action continue
    }
}
action {
    module tokenmt
    name meterftp
    params {
        committed_rate 50000000
        committed_burst 50000000
        red_action_name AF31
        green_action_name markAF22
        global_stats TRUE
    }
}
action {
    module dscpmk
    name markAF31
    params {
        global_stats TRUE
        dscp_map{0-63:26}
        next_action continue
    }
}
action {
    module dscpmk
    name markAF22
    params {
        global_stats TRUE
        dscp_map{0-63:20}
        next_action continue
    }
}
```

## ▼ How to Configure the IPQoS Configuration File for an Application Server

**1 Log in to the IPQoS-enabled application server, and create a new IPQoS configuration file with a `.qos` extension.**

For example, you would create the `/var/ipqos/BigAPPS.qos` file for the application server. Begin with the following required phrases to start the `action` statement that invokes the `ipgpc` classifier:

```
fmt_version 1.0

action {
    module ipgpc
    name ipgpc.classify
    params {
        global_stats TRUE
    }
```

For an explanation of the opening `action` statement, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 798.

**2 Create classes to select traffic from three applications on the BigAPPS server.**

Add the class definitions after the opening `action` statement.

```
    class {
        name smtp
        enable_stats FALSE
        next_action markAF13
    }
    class {
        name news
        next_action markAF21
    }
    class {
        name ftp
        enable_stats TRUE
        next_action meterftp
    }
```

| | |
|---|---|
| `name smtp` | Creates a class that is called `smtp`, which includes email traffic flows to be handled by the SMTP application |
| `enable_stats FALSE` | Enables statistics collection for the `smtp` class. However, because the value of `enable_stats` is `FALSE`, statistics for this class are not taken. |

| | |
|---|---|
| `next_action markAF13` | Instructs the `ipgpc` module to pass packets of the `smtp` class to the `markAF13` `action` statement after `ipgpc` completes processing. |
| `name news` | Creates a class that is called `news`, which includes network news traffic flows to be handled by the NNTP application. |
| `next_action markAF21` | Instructs the `ipgpc` module to pass packets of the `news` class to the `markAF21` action statement after `ipgpc` completes processing. |
| `name ftp` | Creates a class that is called `ftp`, which handles outgoing traffic that is handled by the FTP application. |
| `enable_stats TRUE` | Enables statistics collection for the `ftp` class. |
| `next_action meterftp` | Instructs the `ipgpc` module to pass packets of the `ftp` class to the `meterftp` `action` statement after `ipgpc` completes processing. |

For more information about defining classes, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 798.

**3 Define `filter` clauses to select traffic of the classes defined in Step 2.**

```
    filter {
        name smtpout
        sport smtp
        class smtp
    }
    filter {
        name newsout
        sport nntp
        class news
    }
        filter {
        name ftpout
        sport ftp
        class ftp
    }
        filter {
        name ftpdata
        sport ftp-data
        class ftp
    }
}
```

| | |
|---|---|
| `name smtpout` | Gives the name `smtpout` to the filter. |
| `sport smtp` | Selects traffic with a source port of 25, the well-known port for the `sendmail` (SMTP) application. |
| `class smtp` | Identifies the class to which the filter belongs, in this instance, class `smtp`. |

| | |
|---|---|
| `name newsout` | Gives the name `newsout` to the filter. |
| `sport nntp` | Selects traffic with a source port name of `nntp`, the well-known port name for the network news (NNTP) application. |
| `class news` | Identifies the class to which the filter belongs, in this instance, class `news`. |
| `name ftpout` | Gives the name `ftpout` to the filter. |
| `sport ftp` | Selects control data with a source port of 21, the well-known port number for FTP traffic. |
| `name ftpdata` | Gives the name `ftpdata` to the filter. |
| `sport ftp-data` | Selects traffic with a source port of 20, the well-known port number for FTP data traffic. |
| `class ftp` | Identifies the class to which the `ftpout` and `ftpdata` filters belong, in this instance `ftp`. |

**See Also**

- To define filters, refer to "How to Define Filters in the IPQoS Configuration File" on page 800.
- To define forwarding behaviors for application traffic, refer to "How to Configure Forwarding for Application Traffic in the IPQoS Configuration File" on page 813.
- To configure flow control by using the metering modules, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 815.
- To configure flow accounting, refer to "How to Enable Accounting for a Class in the IPQoS Configuration File" on page 804.

## ▼ How to Configure Forwarding for Application Traffic in the IPQoS Configuration File

The next procedure shows how to configure forwarding for application traffic. In the procedure, you define per-hop behaviors for application traffic classes that might have lower precedence than other traffic on a network. The steps continue building the `/var/ipqos/BigAPPS.qos` file in Example 34–3.

**Before You Begin** The procedure assumes that you have an existing IPQoS configuration file with already-defined classes and already-defined filters for the applications to be marked.

**1 Open the IPQoS configuration file that you have created for the application server, and locate the end of the last `filter` clause.**

In the `/var/ipqos/BigAPPS.qos` file, the last filter is the following:

```
 filter {
        name ftpdata
        sport ftp-data
        class ftp
    }
}
```

**2 Invoke the marker as follows:**

```
action {
    module dscpmk
    name markAF13
```

| | |
|---|---|
| `module dscpmk` | Invokes the marker module `dscpmk`. |
| `name markAF13` | Gives the name `markAF13` to the `action` statement. |

**3 Define the per-hop behavior to be marked on electronic mail traffic flows.**

```
    params {
        global_stats FALSE
        dscp_map{0-63:14}
        next_action continue
    }
}
```

| | |
|---|---|
| `global_stats FALSE` | Enables statistics collection for the `markAF13` marker `action` statement. However, because the value of `enable_stats` is `FALSE`, statistics are not collected. |
| `dscp_map{0–63:14}` | Assigns a DSCP of `14` to the packet headers of the traffic class `smtp`, which is currently being processed by the marker. |
| `next_action continue` | Indicates that no further processing is required on packets of the traffic class `smtp`. These packets can then return to the network stream. |

The DSCP of `14` tells the marker to set all entries in the `dscp` map to the decimal value 14 (binary 001110). The DSCP of `14` sets the AF13 per-hop behavior. The marker marks packets of the `smtp` traffic class with the DSCP of `14` in the DS field.

AF13 assigns all packets with a DSCP of `14` to a high-drop precedence. However, because AF13 also assures a Class 1 priority, the router still guarantees outgoing email traffic a high priority in its queue. For a table of possible AF codepoints, refer to Table 37–2.

4 **Add a marker `action` statement to define a per-hop behavior for network news traffic:**

```
action {
    module dscpmk
    name markAF21
    params {
        global_stats FALSE
        dscp_map{0-63:18}
        next_action continue
    }
}
```

`name markAF21` Gives the name `markAF21` to the `action` statement.

`dscp_map{0–63:18}` Assigns a DSCP of `18` to the packet headers of the traffic class `nntp`, which is currently being processed by the marker.

The DSCP of `18` tells the marker to set all entries in the `dscp` map to the decimal value 18 (binary 010010). The DSCP of `18` sets the AF21 per-hop behavior. The marker marks packets of the `news` traffic class with the DSCP of `18` in the DS field.

AF21 assures that all packets with a DSCP of `18` receive a low-drop precedence, but with only Class 2 priority. Thus, the possibility of network news traffic being dropped is low.

**See Also**
- To add configuration information for web servers, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 798.
- To configure flow control by using the metering modules, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 815.
- To configure flow accounting, refer to "How to Enable Accounting for a Class in the IPQoS Configuration File" on page 804.
- To configure forwarding behaviors on a router, refer to "How to Configure a Router on an IPQoS-Enabled Network" on page 819.
- To activate the IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 822.

## ▼ How to Configure Flow Control in the IPQoS Configuration File

To control the rate at which a particular traffic flow is released onto the network, you must define parameters for the meter. You can use either of the two meter modules, `tokenmt` or `tswtclmt`, in the IPQoS configuration file.

The next procedure continues to build the IPQoS configuration file for the application server in Example 34–3. In the procedure, you configure not only the meter but also two marker actions that are called within the meter `action` statement.

**Before You Begin** The steps assume that you have already defined a class and a filter for the application to be flow-controlled.

**1 Open the IPQoS configuration file that you have created for the applications server.**

In the `/var/ipqos/BigAPPS.qos` file, you begin after the following marker action:

```
action {
    module dscpmk
    name markAF21
    params {
        global_stats FALSE
        dscp_map{0-63:18}
        next_action continue
    }
}
```

**2 Create a meter `action` statement to flow-control traffic of the `ftp` class.**

```
action {
    module tokenmt
    name meterftp
```

| | |
|---|---|
| `module tokenmt` | Invokes the `tokenmt` meter. |
| `name meterftp` | Gives the name `meterftp` to the `action` statement. |

**3 Add parameters to configure the meter's rate.**

```
params {
       committed_rate 50000000
       committed_burst 50000000
```

| | |
|---|---|
| `committed_rate 50000000` | Assigns a transmission rate of 50,000,000 bps to traffic of the `ftp` class. |
| `committed_burst 50000000` | Commits a burst size of 50,000,000 bits to traffic of the `ftp` class. |

For an explanation of `tokenmt` parameters, refer to "Configuring `tokenmt` as a Two-Rate Meter" on page 839.

**4 Add parameters to configure traffic conformance precedences:**

```
    red_action markAF31
    green_action_name markAF22
    global_stats TRUE
    }
}
```

| | |
|---|---|
| `red_action_name markAF31` | Indicates that when the traffic flow of the `ftp` class exceeds the committed rate, packets are sent to the `markAF31` marker `action` statement. |
| `green_action_name markAF22` | Indicates that when traffic flows of class `ftp` conform to the committed rate, packets are sent to the `markAF22` action statement. |
| `global_stats TRUE` | Enables metering statistics for the `ftp` class. |

For more information about traffic conformance, see "Meter Module" on page 838.

**5 Add a marker `action` statement to assign a per-hop behavior to nonconformant traffic flows of class `ftp`.**

```
action {
    module dscpmk
    name markAF31
    params {
        global_stats TRUE
        dscp_map{0-63:26}
        next_action continue
    }
}
```

| | |
|---|---|
| `module dscpmk` | Invokes the marker module `dscpmk`. |
| `name markAF31` | Gives the name `markAF31` to the `action` statement. |
| `global_stats TRUE` | Enables statistics for the `ftp` class. |
| `dscp_map{0–63:26}` | Assigns a DSCP of 26 to the packet headers of the traffic class `ftp` whenever this traffic exceeds the committed rate. |
| `next_action continue` | Indicates that no further processing is required on packets of the traffic class `ftp`. Then these packets can return to the network stream. |

The DSCP of 26 instructs the marker to set all entries in the `dscp` map to the decimal value 26 (binary 011010). The DSCP of 26 sets the AF31 per-hop behavior. The marker marks packets of the `ftp` traffic class with the DSCP of 26 in the DS field.

AF31 assures that all packets with a DSCP of 26 receive a low-drop precedence, but with only Class 3 priority. Therefore, the possibility of nonconformant FTP traffic being dropped is low. For a table of possible AF codepoints, refer to Table 37–2.

**6 Add a marker `action` statement to assign a per-hop behavior to `ftp` traffic flows that conform to the committed rate.**

```
action {
    module dscpmk
```

```
    name markAF22
    params {
        global_stats TRUE
        dscp_map{0-63:20}
        next_action continue
    }
}
```

`name markAF22` Gives the name `markAF22` to the `marker` action.

`dscp_map{0–63:20}` Assigns a DSCP of `20` to the packet headers of the traffic class `ftp` whenever `ftp` traffic conforms to its configured rate.

The DSCP of 20 tells the marker to set all entries in the `dscp` map to the decimal value 20 (binary 010100). The DSCP of `20` sets the AF22 per-hop behavior. The marker marks packets of the `ftp` traffic class with the DSCP of `20` in the DS field.

AF22 assures that all packets with a DSCP of `20` receive a medium-drop precedence with Class 2 priority. Therefore, conformant FTP traffic is assured a medium-drop precedence among flows that are simultaneously released by the IPQoS system. However, the router gives a higher forwarding priority to traffic classes with a Class 1 medium-drop precedence mark or higher. For a table of possible AF codepoints, refer to Table 37–2.

**7 Add the DSCPs that you have created for the application server to the appropriate files on the Diffserv router.**

**See Also**
- To activate the IPQoS configuration file, refer to "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 822.
- To add configuration information for web servers, refer to "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 798.
- To configure flow accounting, refer to "How to Enable Accounting for a Class in the IPQoS Configuration File" on page 804.
- To configure forwarding behaviors on a router, refer to "How to Configure a Router on an IPQoS-Enabled Network" on page 819.

# Providing Differentiated Services on a Router

To provide true differentiated services, you must include a Diffserv-aware router in your network topology, as described in "Hardware Strategies for the Diffserv Network" on page 774. The actual steps for configuring Diffserv on a router and updating that router's files are outside the scope of this guide.

This section gives general steps for coordinating the forwarding information among various IPQoS-enabled systems on the network and the Diffserv router.

## ▼ How to Configure a Router on an IPQoS-Enabled Network

The next procedure uses as its example the topology in Figure 33–4.

**Before You Begin** The next procedure assumes that you have already configured the IPQoS systems on your network by performing the previous tasks in this chapter.

**1 Review the configuration files for all IPQoS-enabled systems on your network.**

**2 Identify each codepoint that is used in the QoS various policies.**

List the codepoints, and the systems and classes, to which the codepoints apply. The next table can illustrate areas where you might have used the same codepoint. This practice is acceptable. However, you should provide other criteria in the IPQoS configuration file, such as a `precedence` selector, to determine the precedence of identically marked classes.

For example, for the sample network that is used in the procedures throughout this chapter, you might construct the following codepoint table.

| System | Class | PHB | DS Codepoint |
|---|---|---|---|
| `Goldweb` | `video` | EF | 46 (101110) |
| `Goldweb` | `goldweb` | AF11 | 10 (001010) |
| `Userweb` | `webout` | AF12 | 12 ( 001100) |
| `BigAPPS` | `smtp` | AF13 | 14 ( 001110) |
| `BigAPPS` | `news` | AF18 | 18 ( 010010) |
| `BigAPPS` | `ftp` conformant traffic | AF22 | 20 ( 010100) |
| `BigAPPS` | `ftp` nonconformant traffic | AF31 | 26 ( 011010) |

**3 Add the codepoints from your network's IPQoS configuration files to the appropriate files on the Diffserv router.**

The codepoints that you supply should help to configure the router's Diffserv scheduling mechanism. Refer to the router manufacturer's documentation and web sites for instructions.

◆ ◆ ◆ CHAPTER 35

# Starting and Maintaining IPQoS (Tasks)

This chapter contains tasks for activating an IPQoS configuration file and for logging IPQoS-related events. The following topics are covered:


- "Administering IPQoS (Task Map)" on page 821
- "Applying an IPQoS Configuration" on page 822
- "Enabling `syslog` Logging for IPQoS Messages" on page 823
- "Troubleshooting with IPQoS Error Messages" on page 824


## Administering IPQoS (Task Map)

This section lists the set of tasks for starting and maintaining IPQoS on a Solaris system. Before you use the tasks, you must have a completed IPQoS configuration file, as described in "Defining a QoS Policy in the IPQoS Configuration File (Task Map)" on page 793.

| Task | Description | For Instructions |
|---|---|---|
| 1. Configure IPQoS on a system. | Use the `ipqosconf` command to activate the IPQoS configuration file on a system. | "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 822 |
| 2. Make the Solaris startup scripts apply the debugged IPQoS configuration file after each system boot. | Ensure that the IPQoS configuration is applied each time the system reboots. | "How to Ensure That the IPQoS Configuration Is Applied After Each Reboot" on page 823. |
| 3. Enable `syslog` logging for IPQoS. | Add an entry to enable `syslog` logging of IPQoS messages. | "How to Enable Logging of IPQoS Messages During Booting" on page 823. |
| 4. Fix any IPQoS problems that arise. | Troubleshoot IPQoS problems by using error messages. | Refer to the error messages in Table 35–1. |

# Applying an IPQoS Configuration

You activate and otherwise manipulate the IPQoS configuration by using the `ipqosconf` command.

## ▼ How to Apply a New Configuration to the IPQoS Kernel Modules

You use the `ipqosconf` command to read the IPQoS configuration file and to configure the IPQoS modules in the UNIX kernel. The next procedure uses as an example the file `/var/ipqos/Goldweb.qos`, which is created in "Creating IPQoS Configuration Files for Web Servers" on page 795. For detailed information, refer to the `ipqosconf`(1M) man page.

**1 Assume the Primary Administrator role, or become superuser, on the IPQoS-enabled system.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Apply the new configuration.**

```
# /usr/sbin/ipqosconf -a/var/ipqos/Goldweb.qos
```

`ipqosconf` writes the information in the specified IPQoS configuration file into the IPQoS modules in the Solaris kernel. In this example, the contents of `/var/ipqos/Goldweb.qos` are applied to the current Solaris kernel.

---

**Note –** When you apply an IPQoS configuration file with the `-a` option, the actions in the file are active for the current session only.

---

**3 Test and debug the new IPQoS configuration.**

Use UNIX utilities to track IPQoS behavior and to gather statistics on your IPQoS implementation. This information can help you determine if the configuration operates as expected.

**See Also**

- To view statistics on how IPQoS modules are working, refer to "Gathering Statistical Information" on page 832.
- To log `ipqosconf` messages, refer to "Enabling `syslog` Logging for IPQoS Messages" on page 823.
- To ensure that the current IPQoS configuration is applied after each boot, refer to "How to Ensure That the IPQoS Configuration Is Applied After Each Reboot" on page 823.

### ▼ How to Ensure That the IPQoS Configuration Is Applied After Each Reboot

You must explicitly make an IPQoS configuration persistent across reboots. Otherwise, the current configuration applies only until the system reboots. When IPQoS works correctly on a system, do the following to make the configuration persistent across reboots.

1. **Assume the Primary Administrator role, or become superuser, on the IPQoS-enabled system.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Test for the existence of an IPQoS configuration in the kernel modules.**

   ```
   # ipqosconf -l
   ```

   If a configuration already exists, `ipqosconf` displays the configuration on the screen. If you do not receive output, apply the configuration, as explained in "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 822.

3. **Ensure that the existing IPQoS configuration is applied every time the IPQoS system reboots.**

   ```
   # /usr/sbin/ipqosconf -c
   ```

   The `-c` option causes the current IPQoS configuration to be represented in the boot-time configuration file `/etc/inet/ipqosinit.conf`.

## Enabling `syslog` Logging for IPQoS Messages

To record IPQoS boot-time messages, you need to modify the `/etc/syslog.conf` file as shown in the next procedure.

### ▼ How to Enable Logging of IPQoS Messages During Booting

1. **Assume the Primary Administrator role, or become superuser, on the IPQoS-enabled system.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Open the `/etc/syslog.conf` file.**

**3 Add the following text as the final entry in the file.**

```
user.info                       /var/adm/messages
```

Use tabs rather than spaces between the columns.

The entry logs all boot-time messages that are generated by IPQoS into the `/var/adm/messages` file.

**4 Reboot the system to apply the messages.**

**Example 35–1** IPQoS Output From `/var/adm/messages`

When you view `/var/adm/messages` after system reboot, your output might contain IPQoS logging messages that are similar to the following.

```
May 14 10:44:33 ipqos-14 ipqosconf: [ID 815575 user.info]
 New configuration applied.
May 14 10:44:46 ipqos-14 ipqosconf: [ID 469457 user.info]
Current configuration saved to init file.
May 14 10:44:55 ipqos-14 ipqosconf: [ID 435810 user.info]
Configuration flushed.
```

You might also see IPQoS error messages that are similar to the following in your IPQoS system's `/var/adm/messages` file.

```
May 14 10:56:47 ipqos-14 ipqosconf: [ID 123217 user.error]
 Missing/Invalid config file fmt_version.
May 14 10:58:19 ipqos-14 ipqosconf: [ID 671991 user.error]
No ipgpc action defined.
```

For a description of these error messages, see Table 35–1.

# Troubleshooting with IPQoS Error Messages

This section contains a table of error messages that are generated by IPQoS and their possible solutions.

**TABLE 35–1** IPQoS Error Messages

| Error Message | Description | Solution |
|---|---|---|
| `Undefined action in parameter` *parameter-name's* `action` *action-name* | In the IPQoS configuration file, the action name that you specified in *parameter-name* does not exist in the configuration file. | Create the action. Or, refer to a different, existing action in the parameter. |

**TABLE 35–1** IPQoS Error Messages *(Continued)*

| Error Message | Description | Solution |
|---|---|---|
| `action` *action-name* `involved in cycle` | In the IPQoS configuration file, *action-name* is part of a cycle of actions, which is not allowed by IPQoS. | Determine the action cycle. Then remove one of the cyclical references from the IPQoS configuration file. |
| `Action` *action-name* `isn't referenced by any other actions` | A non-`ipgpc` action definition is not referenced by any other defined actions in the IPQoS configuration, which is not allowed by IPQoS. | Remove the unreferenced action. Alternatively, make another action reference the currently unreferenced action. |
| `Missing/Invalid config file fmt_version` | The format of the configuration file is not specified as the first entry of the file, which is required by IPQoS. | Add the format version, as explained in "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 798. |
| `Unsupported config file format version` | The format version that is specified in the configuration file is not supported by IPQoS. | Change the format version to `fmt_version 1.0`, which is required to run the Solaris 9 9/02 and later versions of IPQoS. |
| `No ipgpc action defined.` | You did not define an action for the `ipgpc` classifier in the configuration file, which is an IPQoS requirement. | Define an action for `ipgpc`, as shown in "How to Create the IPQoS Configuration File and Define Traffic Classes" on page 798. |
| `Can't commit a null configuration` | When you ran `ipqosconf -c` to commit a configuration, that configuration was empty, which IPQoS does not allow. | Be sure to apply a configuration file before you attempt to commit a configuration. For instructions, see "How to Apply a New Configuration to the IPQoS Kernel Modules" on page 822. |
| `Invalid CIDR mask on line` *line-number* | In the configuration file, you used a CIDR mask as part of the IP address that is out of the valid range for IP addresses. | Change the mask value to be in the range of 1–32 for IPv4 and 1–128 for IPv6. |
| `Address masks aren't allowed for host names line` *line-number* | In the configuration file, you defined a CIDR mask for a host name, which is not allowed in IPQoS. | Remove the mask or change the host name to an IP address. |
| `Invalid module name line` *line-number* | In the configuration file, the module name that you specified in an action statement is invalid. | Check the spelling of the module name. For a list of IPQoS modules, refer to Table 37–5. |
| `ipgpc action has incorrect name line` *line-number* | The name that you gave to the `ipgpc` action in the configuration file is not the required `ipgpc.classify`. | Rename the action `ipgpc.classify`. |
| `Second parameter clause not supported line` *line-number* | In the configuration file, you specified two parameter clauses for a single action, which IPQoS does not allow. | Combine all parameters for the action into a single parameters clause. |
| `Duplicate named action` | In the configuration file, you gave the same name to two actions. | Rename or remove one of the actions. |

**TABLE 35–1** IPQoS Error Messages *(Continued)*

| Error Message | Description | Solution |
|---|---|---|
| `Duplicate named filter/class in action` *action-name* | You gave the same name to two filters or two classes in the same action, which is not allowed in the IPQoS configuration file. | Rename or remove one of the filters or classes. |
| `Undefined class in filter` *filter-name* `in action` *action-name* | In the configuration file, the filter references a class that is not defined in the action. | Create the class, or change the filter reference to an already existing class. |
| `Undefined action in class` *class-name* `action` *action-name* | The class refers to an action that is not defined in the configuration file. | Create the action, or change the reference to an already existing action. |
| `Invalid parameters for action` *action-name* | In the configuration file, one of the parameters is invalid. | For the module that is called by the named action, refer to the module entry in "IPQoS Architecture and the Diffserv Model" on page 835. Alternatively, you can refer to the `ipqosconf`(1M) man page. |
| `Mandatory parameter missing for action` *action-name* | You have not defined a required parameter for an action in the configuration file. | For the module that is called by the named action, refer to the module entry in "IPQoS Architecture and the Diffserv Model" on page 835. Alternatively, you can refer to the `ipqosconf`(1M) man page. |
| `Max number of classes reached in ipgpc` | You specified more classes than are allowed in the `ipgpc` action of the IPQoS configuration file. The maximum number is 10007. | Review the configuration file, and remove unneeded classes. Alternatively, you can raise the maximum number of classes by adding to the `/etc/system` file the entry `ipgpc_max_classes`*class-number*. |
| `Max number of filters reached in action ipgpc` | You specified more filters than are allowed in the `ipgpc` action of the IPQoS configuration file. The maximum number is 10007. | Review the configuration file, and remove unneeded filters. Alternatively, you can raise the maximum number of filters by adding to the `/etc/system` file the entry `ipgpc_max_filters`*filter-number*. |
| `Invalid/missing parameters for filter` *filter-name* `in action ipgpc` | In the configuration file, filter *filter-name* has an invalid or missing parameter. | Refer to the `ipqosconf`(1M) man page for the list of valid parameters. |
| `Name not allowed to start with '!', line` *line-number* | You began an action, filter, or class name with an exclamation mark (!), which is not allowed in the IPQoS file. | Remove the exclamation mark, or rename the action, class, or filter. |
| `Name exceeds the maximum name length line` *line-number* | You defined a name for an action, class, or filter in the configuration file that exceeds the maximum length of 23 characters. | Give a shorter name to the action, class, or filter. |
| `Array declaration line` *line-number* `is invalid` | In the configuration file, the array declaration for the parameter on line *line-number* is invalid. | For the correct syntax of the array declaration that is called by the `action` statement with the invalid array, refer to "IPQoS Architecture and the Diffserv Model" on page 835. Alternatively, refer to the `ipqosconf`(1M) man page. |

**TABLE 35–1** IPQoS Error Messages *(Continued)*

| Error Message | Description | Solution |
|---|---|---|
| `Quoted string exceeds line,` *line-number* | The string does not have the terminating quotation marks on the same line, which is required in the configuration file. | Make sure that the quoted string begins and ends on the same line in the configuration file. |
| `Invalid value, line` *line-number* | The value that is given on *line-number* of the configuration file is not supported for the parameter. | For the acceptable values for the module that is called by the `action` statement, refer to the module description in "IPQoS Architecture and the Diffserv Model" on page 835. Alternatively, you can refer to the `ipqosconf`(1M) man page. |
| `Unrecognized value, line` *line-number* | The value on *line-number* of the configuration file is not a supported enumeration value for its parameter. | Check that the enumeration value is correct for the parameter. For a description of the module that is called by the `action` statement with the unrecognized line number, refer to "IPQoS Architecture and the Diffserv Model" on page 835. Alternatively, you can refer to the `ipqosconf`(1M) man page. |
| `Malformed value list line` *line-number* | The enumeration that is specified on *line-number* of the configuration file does not conform to the specification syntax. | For correct syntax for the module that is called by the `action` statement with the malformed value list, refer to the module description in "IPQoS Architecture and the Diffserv Model" on page 835. Alternatively, you can refer to the `ipqosconf`(1M) man page. |
| `Duplicate parameter line` *line-number* | A duplicate parameter was specified on *line-number*, which is not allowed in the configuration file. | Remove one of the duplicate parameters. |
| `Invalid action name line` *line-number* | You gave the action on *line-number* of the configuration file a name that uses the predefined name "continue" or "drop." | Rename the action so that the action does not use a predefined name. |
| `Failed to resolve src/dst host name for filter at line` *line-number*`, ignoring filter` | `ipqosconf` could not resolve the source or destination address that was defined for the given filter in the configuration file. Therefore, the filter is ignored. | If the filter is important, try applying the configuration at a later time. |
| `Incompatible address version line` *line-number* | The IP version of the address on *line-number* is incompatible with the version of a previously specified IP address or `ip_version` parameter. | Change the two conflicting entries to be compatible. |
| `Action at line` *line-number* `has the same name as currently installed action, but is for a different module` | You tried to change the module of an action that already exists in the system's IPQoS configuration, which is not allowed. | Flush the current configuration before you apply the new configuration. |

◆ ◆ ◆ CHAPTER 36

# Using Flow Accounting and Statistics Gathering (Tasks)

This chapter explains how to obtain accounting and statistical information on traffic that is handled by an IPQoS system. The following topics are discussed:


- "Setting Up Flow Accounting (Task Map)" on page 829
- "Recording Information About Traffic Flows" on page 829
- "Gathering Statistical Information" on page 832


## Setting Up Flow Accounting (Task Map)

The following task map lists the generic tasks for obtaining information about traffic flows by using the `flowacct` module.

| Task | Description | For Instructions |
|---|---|---|
| 1. Create a file to contain accounting information for traffic flows. | Use the `acctadm` command to create a file that holds the results of processing by `flowacct`. | "How to Create a File for Flow-Accounting Data" on page 830 |
| 2. Define `flowacct` parameters in the IPQoS configuration file. | Define values for the `timer`, `timeout`, and `max_limit` parameters. | "How to Enable Accounting for a Class in the IPQoS Configuration File" on page 804 |

## Recording Information About Traffic Flows

You use the IPQoS `flowacct` module to collect information about traffic flows. For example, you can collect source and destination addresses, number of packets in a flow, and similar data. The process of accumulating and recording information about flows is called *flow accounting*.

The results of flow accounting on traffic of a particular class are recorded in a table of *flow records*. Each flow record consists of a series of attributes. These attributes contain data about traffic flows of a particular class over an interval of time. For a list of the `flowacct` attributes, refer to Table 37–4.

Flow accounting is particularly useful for billing clients as is defined in their service-level agreements (SLAs). You can also use flow accounting to obtain flow statistics for critical applications. This section contains tasks for using `flowacct` with the Solaris extended accounting facility to obtain data on traffic flows.

The following information is contained in sources outside this chapter:

- For instructions on creating an action statement for `flowacct` in the IPQoS configuration file, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 815.
- To learn how `flowacct` works, refer to "Classifier Module" on page 835.
- For technical information, refer to the `flowacct`(7ipp) man page.

## How to Create a File for Flow-Accounting Data

Before you add a `flowacct` action to the IPQoS configuration file, you must create a file for flow records from the `flowacct` module. You use the `acctadm` command for this purpose. `acctadm` can record either basic attributes or extended attributes in the file. All `flowacct` attributes are listed in Table 37–4. For detailed information about `acctadm`, refer to the `acctadm`(1M) man page.

**1 Assume the Primary Administrator role, or become superuser, on the IPQoS-enabled system.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

**2 Create a basic flow-accounting file.**

The following example shows how to create a basic flow-accounting file for the premium web server that is configured in Example 34–1.

```
# /usr/sbin/acctadm -e basic -f /var/ipqos/goldweb/account.info flow
```

| | |
|---|---|
| `acctadm -e` | Invokes `acctadm` with the `-e` option. The `-e` option enables the arguments that follow. |
| `basic` | States that only data for the eight basic `flowacct` attributes is to be recorded in the file. |
| `/var/ipqos/goldweb/account.info` | Specifies the fully qualified path name of the file to hold the flow records from `flowacct`. |
| `flow` | Instructs `acctadm` to enable flow accounting. |

**3 View information about flow accounting on the IPQoS system by typing `acctadm` without arguments.**

`acctadm` generates the following output:

```
Task accounting: inactive
       Task accounting file: none
     Tracked task resources: none
   Untracked task resources: extended
         Process accounting: inactive
    Process accounting file: none
  Tracked process resources: none
Untracked process resources: extended,host,mstate
           Flow accounting: active
       Flow accounting file: /var/ipqos/goldweb/account.info
     Tracked flow resources: basic
   Untracked flow resources: dsfield,ctime,lseen,projid,uid
```

All entries but the last four are for use with the Solaris Resource Manager feature. The next table explains the entries that are specific to IPQoS.

| Entry | Description |
|---|---|
| `Flow accounting: active` | Indicates that flow accounting is turned on. |
| `Flow accounting file: /var/ipqos/goldweb/account.info` | Gives the name of the current flow-accounting file. |
| `Tracked flow resources: basic` | Indicates that only the basic flow attributes are tracked. |
| `Untracked flow resources: dsfield,ctime,lseen,projid,uid` | Lists the `flowacct` attributes that are not tracked in the file. |

**4 (Optional) Add the extended attributes to the accounting file.**

```
# acctadm -e extended -f /var/ipqos/goldweb/account.info flow
```

**5 (Optional) Return to recording only the basic attributes in the accounting file.**

```
# acctadm -d extended -e basic -f /var/ipqos/goldweb/account.info
```

The `-d` option disables extended accounting.

**6 View the contents of a flow-accounting file.**

Instructions for viewing the contents of a flow-accounting file are in "Perl Interface to libexacct" in *System Administration Guide: Virtualization Using the Solaris Operating System*.

**See Also**

- For detailed information on the extended accounting feature, refer to Chapter 4, "Extended Accounting (Overview)," in *System Administration Guide: Virtualization Using the Solaris Operating System*.
- To define `flowacct` parameters in the IPQoS configuration file, refer to "How to Enable Accounting for a Class in the IPQoS Configuration File" on page 804.
- To print the data in the file that was created with `acctadm`, refer to "Perl Interface to libexacct" in *System Administration Guide: Virtualization Using the Solaris Operating System*.

# Gathering Statistical Information

You can use the `kstat` command to generate statistical information from the IPQoS modules. Use the following syntax:

**`/bin/kstat -m`** *ipqos-module-name*

You can specify any valid IPQoS module name, as shown in Table 37–5. For example, to view statistics that are generated by the `dscpmk` marker, you use the following form of `kstat`:

```
/bin/kstat -m dscpmk
```

For technical details, refer to the `kstat`(1M) man page.

**EXAMPLE 36–1** `kstat` Statistics for IPQoS

Here is an example of possible results from running `kstat` to obtain statistics about the `flowacct` module.

```
# kstat -m flowacct
module: flowacct                        instance: 3
name:   Flowacct statistics             class:    flacct
        bytes_in_tbl                    84
        crtime                          345728.504106363
        epackets                        0
        flows_in_tbl                    1
        nbytes                          84
        npackets                        1
        snaptime                        345774.031843301
        usedmem                         256
```

`class: flacct` Gives the name of the class to which the traffic flows belong, in this example `flacct`.

**EXAMPLE 36–1** `kstat` Statistics for IPQoS *(Continued)*

| | |
|---|---|
| `bytes_in_tbl` | Total number of bytes in the flow table. The total number of bytes is the sum in bytes of all the flow records that currently reside in the flow table. The total number of bytes for this flow table is 84. If no flows are in the table, the value for `bytes_in_tbl` is 0. |
| `crtime` | The last time that this `kstat` output was created. |
| `epackets` | Number of packets that resulted in an error during processing, in this example 0. |
| `flows_in_tbl` | Number of flow records in the flow table, which in this example is 1. When no records are in the table, the value for `flows_in_tbl` is 0. |
| `nbytes` | Total number of bytes that are seen by this `flowacct` action instance, which is 84 in the example. The value includes bytes that are currently in the flow table. The value also includes bytes that have timed out and are no longer in the flow table. |
| `npackets` | Total number of packets that are seen by this `flowacct` action instance, which is 1 in the example. `npackets` includes packets that are currently in the flow table. `npackets` also includes packets that have timed out—are no longer in the flow table. |
| `usedmem` | Memory in bytes in use by the flow table that is maintained by this `flowacct` instance. The `usedmem` value is 256 in the example. The value for `usedmem` is 0 when the flow table does not have any flow records. |

◆ ◆ ◆ CHAPTER 37

# IPQoS in Detail (Reference)

This chapter contains reference materials that provide in-depth details about the following IPQoS topics:


- "IPQoS Architecture and the Diffserv Model" on page 835
- "IPQoS Configuration File" on page 847
- "`ipqosconf` Configuration Utility" on page 851


For an overview, refer to Chapter 32, "Introducing IPQoS (Overview)." For planning information, refer to Chapter 33, "Planning for an IPQoS-Enabled Network (Tasks)." For procedures for configuring IPQoS, refer to Chapter 34, "Creating the IPQoS Configuration File (Tasks)."

## IPQoS Architecture and the Diffserv Model

This section describes the IPQoS architecture and how IPQoS implements the differentiated services (Diffserv) model that is defined inRFC 2475, An Architecture for Differentiated Services (`http://www.ietf.org/rfc/rfc2475.txt?number=2475`). The following elements of the Diffserv model are included in IPQoS:

- Classifier
- Meter
- Marker

In addition, IPQoS includes the flow-accounting module and the `dlcosmk` marker for use with virtual local area network (VLAN) devices.

### Classifier Module

In the Diffserv model, the *classifier* is responsible for organizing selected traffic flows into groups on which to apply different service levels. The classifiers that are defined in RFC 2475 were originally designed for boundary routers. In contrast, the IPQoS classifier `ipgpc` is

designed to handle traffic flows on hosts that are internal to the local network. Therefore, a network with both IPQoS systems and a Diffserv router can provide a greater degree of differentiated services. For a technical description of `ipgpc`, refer to the `ipgpc`(7ipp) man page.

The `ipgpc` classifier does the following:

1. Selects traffic flows that meet the criteria specified in the IPQoS configuration file on the IPQoS-enabled system

   The QoS policy defines various criteria that must be present in packet headers. These criteria are called *selectors*. The `ipgpc` classifier compares these selectors against the headers of packets that are received by the IPQoS system. `ipgpc` then selects all matching packets.
2. Separates the packet flows into *classes*, network traffic with the same characteristics, as defined in the IPQoS configuration file
3. Examines the value in the packet's differentiated service (DS) field for the presence of a differentiated services codepoint (DSCP)

   The presence of the DSCP indicates whether the incoming traffic has been marked by the sender with a forwarding behavior.
4. Determines what further action is specified in the IPQoS configuration file for packets of a particular class
5. Passes the packets to the next IPQoS module specified in the IPQoS configuration file, or returns the packets to the network stream

For an overview of the classifier, refer to "Classifier (`ipgpc`) Overview" on page 765. For information on invoking the classifier in the IPQoS configuration file, refer to "IPQoS Configuration File" on page 847.

## IPQoS Selectors

The `ipgpc` classifier supports a variety of selectors that you can use in the `filter` clause of the IPQoS configuration file. When you define a filter, always use the minimum number of selectors that are needed to successfully retrieve traffic of a particular class. The number of filters you define can impact IPQoS performance.

The next table lists the selectors that are available for `ipgpc`.

**TABLE 37–1** Filter Selectors for the IPQoS Classifier

| Selector | Argument | Information Selected |
|---|---|---|
| `saddr` | IP address number. | Source address. |
| `daddr` | IP address number. | Destination address. |

**TABLE 37–1** Filter Selectors for the IPQoS Classifier *(Continued)*

| Selector | Argument | Information Selected |
|---|---|---|
| `sport` | Either a port number or service name, as defined in `/etc/services`. | Source port from which a traffic class originated. |
| `dport` | Either a port number or service name, as defined in `/etc/services`. | Destination port to which a traffic class is bound. |
| `protocol` | Either a protocol number or protocol name, as defined in `/etc/protocols`. | Protocol to be used by this traffic class. |
| `dsfield` | DS codepoint (DSCP) with a value of 0–63. | DSCP, which defines any forwarding behavior to be applied to the packet. If this parameter is specified, the `dsfield_mask` parameter must also be specified. |
| `dsfield_mask` | Bit mask with a value of 0–255. | Used in tandem with the `dsfield` selector. `dsfield_mask` is applied to the `dsfield` selector to determine which of its bits to match against. |
| `if_name` | Interface name. | Interface to be used for either incoming or outgoing traffic of a particular class. |
| `if_groupname` | Interface group name. | Interface group to be used for either incoming or outgoing traffic of a particular class. |
| `user` | Number of the UNIX user ID or user name to be selected. If no user ID or user name is on the packet, the default –1 is used. | User ID that is supplied to an application. |
| `projid` | Number of the project ID to be selected. | Project ID that is supplied to an application. |
| `priority` | Priority number. Lowest priority is 0. | Priority that is given to packets of this class. Priority is used to order the importance of filters for the same class. |
| `direction` | Argument can be one of the following: | Direction of packet flow on the IPQoS machine. |
| | `LOCAL_IN` | Input traffic local to the IPQoS system. |
| | `LOCAL_OUT` | Output traffic local to the IPQoS system. |
| | `FWD_IN` | Input traffic to be forwarded. |
| | `FWD_OUT` | Output traffic to be forwarded. |
| `precedence` | Precedence value. Highest precedence is 0. | Precedence is used to order filters with the same priority. |
| `ip_version` | `V4` or `V6` | Addressing scheme that is used by the packets, either IPv4 or IPv6. |

# Meter Module

The *meter* tracks the transmission rate of flows on a per-packet basis. The meter then determines whether the packet conforms to the configured parameters. The meter module determines the next action for a packet from a set of actions that depend on packet size, configured parameters, and flow rate.

The meter consists of two metering modules, `tokenmt` and `tswtclmt`, which you configure in the IPQoS configuration file. You can configure either module or both modules for a class.

When you configure a metering module, you can define two parameters for rate:

- `committed-rate` – Defines the acceptable transmission rate in bits per second for packets of a particular class
- `peak-rate` – Defines the maximum transmission rate in bits per second that is allowable for packets of a particular class

A metering action on a packet can result in one of three outcomes:

- `green` – The packet causes the flow to remain within its committed rate.
- `yellow` – The packet causes the flow to exceed its committed rate but not its peak rate.
- `red` – The packet causes the flow to exceed its peak rate.

You can configure each outcome with different actions in the IPQoS configuration file. Committed rate and peak rate are explained in the next section.

## `tokenmt` Metering Module

The `tokenmt` module uses *token buckets* to measure the transmission rate of a flow. You can configure `tokenmt` to operate as a single-rate or two-rate meter. A `tokenmt` action instance maintains two token buckets that determine whether the traffic flow conforms to configured parameters.

The `tokenmt`(7ipp) man page explains how IPQoS implements the token meter paradigm. You can find more general information about token buckets in Kalevi Kilkki's *Differentiated Services for the Internet* and on a number of web sites.

Configuration parameters for `tokenmt` are as follows:

- `committed_rate` – Specifies the committed rate of the flow in bits per second.
- `committed_burst` – Specifies the committed burst size in bits. The `committed_burst` parameter defines how many outgoing packets of a particular class can pass onto the network at the committed rate.
- `peak_rate` – Specifies the peak rate in bits per second.
- `peak_burst` – Specifies the peak or excess burst size in bits. The `peak_burst` parameter grants to a traffic class a peak-burst size that exceeds the committed rate.

- `color_aware` – Turns on awareness mode for `tokenmt`.
- `color_map` – Defines an integer array that maps DSCP values to green, yellow, or red.

## Configuring `tokenmt` as a Single-Rate Meter

To configure `tokenmt` as a single-rate meter, do not specify a `peak_rate` parameter for `tokenmt` in the IPQoS configuration file. To configure a single-rate `tokenmt` instance to have a red, green, or a yellow outcome, you must specify the `peak_burst` parameter. If you do not use the `peak_burst` parameter, you can configure `tokenmt` to have only a red outcome or green outcome. For an example of a single-rate `tokenmt` with two outcomes, see Example 34–3.

When `tokenmt` operates as a single-rate meter, the `peak_burst` parameter is actually the excess burst size. `committed_rate`, and either `committed_burst` or `peak_burst`, must be nonzero positive integers.

## Configuring `tokenmt` as a Two-Rate Meter

To configure `tokenmt` as a two-rate meter, specify a `peak_rate` parameter for the `tokenmt` action in the IPQoS configuration file. A two-rate `tokenmt` always has the three outcomes, red, yellow, and green. The `committed_rate`, `committed_burst`, and `peak_burst` parameters must be nonzero positive integers.

## Configuring `tokenmt` to Be Color Aware

To configure a two-rate `tokenmt` to be color aware, you must add parameters to specifically add "color awareness." The following is an example action statement that configures `tokenmt` to be color aware.

**EXAMPLE 37–1** Color-Aware `tokenmt` Action for the IPQoS Configuration File

```
action {
    module tokenmt
    name meter1
    params {
          committed_rate 4000000
          peak_rate 8000000
          committed_burst 4000000
          peak_burst 8000000
          global_stats true
          red_action_name continue
          yellow_action_name continue
          green_action_name continue
          color_aware true
          color_map {0-20,22:GREEN;21,23-42:RED;43-63:YELLOW}
    }
}
```

You turn on color awareness by setting the `color_aware` parameter to `true`. As a color-aware meter, `tokenmt` assumes that the packet has already been marked as red, yellow, or green by a previous `tokenmt` action. Color-aware `tokenmt` evaluates a packet by using the DSCP in the packet header in addition to the parameters for a two-rate meter.

The `color_map` parameter contains an array into which the DSCP in the packet header is mapped. Consider the following `color_map` array:

```
color_map {0-20,22:GREEN;21,23-42:RED;43-63:YELLOW}
```

Packets with a DSCP of 0–20 and 22 are mapped to green. Packets with a DSCP of 21 and 23–42 are mapped to red. Packets with a DSCP of 43–63 are mapped to yellow. `tokenmt` maintains a default color map. However, you can change the default as needed by using the `color_map` parameters.

In the *color*`_action_name` parameters, you can specify `continue` to complete processing of the packet. Or, you can add an argument to send the packet to a marker action, for example, `yellow_action_name mark22`.

### `tswtclmt` Metering Module

The `tswtclmt` metering module estimates average bandwidth for a traffic class by using a time-based *rate estimator*. `tswtclmt` always operates as a three-outcome meter. The rate estimator provides an estimate of the flow's arrival rate. This rate should approximate the running average bandwidth of the traffic stream over a specific period or time, its *time window*. The rate estimation algorithm is taken from RFC 2859, *A Time Sliding Window Three Colour Marker*.

You use the following parameters to configure `tswtclmt`:

- `committed_rate` – Specifies the committed rate in bits per second
- `peak_rate` – Specifies the peak rate in bits per second
- `window` – Defines the time window, in milliseconds over which history of average bandwidth is kept

For technical details on `tswtclmt`, refer to the`tswtclmt`(7ipp) man page. For general information on rate shapers that are similar to `tswtclmt`, see RFC 2963, A Rate Adaptive Shaper for Differentiated Services (`http://www.ietf.org/rfc/rfc2963.txt?number=2963`).

## Marker Module

IPQoS includes two marker modules, `dscpmk` and `dlcosmk`. This section contains information for using both markers. Normally, you should use `dscpmk` because `dlcosmk` is only available for IPQoS systems with VLAN devices.

For technical information about `dscpmk`, refer to the `dscpmk`(7ipp) man page. For technical information about `dlcosmk`, refer to the `dlcosmk`(7ipp) man page.

## Using the `dscpmk` Marker for Forwarding Packets

The marker receives traffic flows after the flows are processed by the classifier or by the metering modules. The marker marks the traffic with a forwarding behavior. This forwarding behavior is the action to be taken on the flows after the flows leaving the IPQoS system. Forwarding behavior to be taken on a traffic class is defined in the *per-hop behavior (PHB)*. The PHB assigns a priority to a traffic class, which indicates the precedence flows of that class in relation to other traffic classes. PHBs only govern forwarding behaviors on the IPQoS system's contiguous network. For more information on PHBs, refer to "Per-Hop Behaviors" on page 769.

*Packet forwarding* is the process of sending traffic of a particular class to its next destination on a network. For a host such as an IPQoS system, a packet is forwarded from the host to the local network stream. For a Diffserv router, a packet is forwarded from the local network to the router's next hop.

The marker marks the DS field in the packet header with a well-known forwarding behavior that is defined in the IPQoS configuration file. Thereafter, the IPQoS system and subsequent Diffserv-aware systems forward the traffic as indicated in the DS field until the mark changes. To assign a PHB, the IPQoS system marks a value in the DS field of the packet header. This value is called the differentiated services codepoint (DSCP). The Diffserv architecture defines two types of forwarding behaviors, EF and AF, which use different DSCPs. For overview information about DSCPs, refer to "DS Codepoint" on page 769.

The IPQoS system reads the DSCP for the traffic flow and evaluates the flow's precedence in relation to other outgoing traffic flows. The IPQoS system then prioritizes all concurrent traffic flows and releases each flow onto the network by its priority.

The Diffserv router receives the outgoing traffic flows and reads the DS field in the packet headers. The DSCP enables the router to prioritize and schedule the concurrent traffic flows. The router forwards each flow by the priority that is indicated by the PHB. Note that the PHB cannot apply beyond the boundary router of the network unless Diffserv-aware systems on subsequent hops also recognize the same PHB.

### Expedited Forwarding (EF) PHB

*Expedited forwarding* (EF) guarantees that packets with the recommended EF codepoint 46 (101110) receive the best treatment that is available on release to the network. Expedited forwarding is often compared to a leased line. Packets with the 46 (101110) codepoint are guaranteed preferential treatment by all Diffserv routers en route to the packets' destination. For technical information about EF, refer to RFC 2598, *An Expedited Forwarding PHB*.

## Assured Forwarding (AF) PHB

*Assured forwarding* (AF) provides four different classes of forwarding behaviors that you can specify to the marker. The next table shows the classes, the three drop precedences that are provided with each class, and the recommended DSCPs that are associated with each precedence. Each DSCP is represented by its AF value, its value in decimal, and its value in binary.

**TABLE 37–2** Assured Forwarding Codepoints

| | Class 1 | Class 2 | Class 3 | Class 4 |
|---|---|---|---|---|
| **Low-Drop Precedence** | AF11 = 10 (001010) | AF21 = 18 (010010) | AF31 = 26 (011010) | AF41 = 34 (100010) |
| **Medium-Drop Precedence** | AF12 = 12 (001100) | AF22 = 20 (010100) | AF32 = 28 (011100) | AF42 = 36 (100100) |
| **High-Drop Precedence** | AF13 = 14 (001110) | AF23 = 22 (010110) | AF33 = 30 (011110) | AF43 = 38 (100110) |

Any Diffserv-aware system can use the AF codepoint as a guide for providing differentiated forwarding behaviors to different classes of traffic.

When these packets reach a Diffserv router, the router evaluates the packets' codepoints along with DSCPs of other traffic in the queue. The router then forwards or drops packets, depending on the available bandwidth and the priorities that are assigned by the packets' DSCPs. Note that packets that are marked with the EF PHB are guaranteed bandwidth over packets that are marked with the various AF PHBs.

Coordinate packet marking between any IPQoS systems on your network and the Diffserv router to ensure that packets are forwarded as expected. For example, suppose IPQoS systems on your network mark packets with AF21 (010010), AF13 (001110), AF43 (100110), and EF (101110) codepoints. You then need to add the AF21, AF13, AF43, and EF DSCPs to the appropriate file on the Diffserv router.

For a technical explanation of the AF codepoint table, refer to RFC 2597. Router manufacturers Cisco Systems and Juniper Networks have detailed information about setting the AF PHB on their web sites. You can use this information to define AF PHBs for IPQoS systems as well as routers. Additionally, router manufacturers' documentation contains instructions for setting DS codepoints on their equipment.

## Supplying a DSCP to the Marker

The DSCP is 6 bits in length. The DS field is 1 byte long. When you define a DSCP, the marker marks the first 6 significant bits of the packet header with the DS codepoint. The remaining 2 least-significant bits are unused.

To define a DSCP, you use the following parameter within a marker action statement:

```
dscp_map{0-63:DS_codepoint}
```

The `dscp_map` parameter is a 64-element array, which you populate with the (DSCP) value. `dscp_map` is used to map incoming DSCPs to outgoing DSCPs that are applied by the `dscpmk` marker.

You must specify the DSCP value to `dscp_map` in decimal notation. For example, you must translate the EF codepoint of 101110 into the decimal value 46, which results in `dscp_map{0-63:46}`. For AF codepoints, you must translate the various codepoints that are shown in Table 37–2 to decimal notation for use with `dscp_map`.

## Using the `dlcosmk` Marker With VLAN Devices

The `dlcosmk` marker module marks a forwarding behavior in the MAC header of a datagram. You can use `dlcosmk` only on an IPQoS system with a VLAN interface.

`dlcosmk` adds four bytes, which are known as the *VLAN tag*, to the MAC header. The VLAN tag includes a 3-bit user-priority value, which is defined by the IEEE 801.D standard. Diffserv-aware switches that understand VLAN can read the user-priority field in a datagram. The 801.D user priority values implement the class-of-service (CoS) marks, which are well known and understood by commercial switches.

You can use the user-priority values in the `dlcosmk` marker action by defining the class of service marks that are listed in the next table.

**TABLE 37–3** 801.D User-Priority Values

| Class of Service | Definition |
|---|---|
| 0 | Best effort |
| 1 | Background |
| 2 | Spare |
| 3 | Excellent effort |
| 4 | Controlled load |
| 5 | Video less than 100ms latency |
| 6 | Video less than 10ms latency |
| 7 | Network control |

For more information on `dlcosmk`, refer to the `dlcosmk`(7ipp) man page.

## IPQoS Configuration for Systems With VLAN Devices

This section introduces a simple network scenario that shows how to implement IPQoS on systems with VLAN devices. The scenario includes two IPQoS systems, `machine1` and `machine2`, that are connected by a switch. The VLAN device on `machine1` has the IP address `10.10.8.1`. The VLAN device on `machine2` has the IP address `10.10.8.3`.

The following IPQoS configuration file for `machine1` shows a simple solution for marking traffic through the switch to `machine2`.

**EXAMPLE 37–2** IPQoS Configuration File for a System With a VLAN Device

```
fmt_version 1.0
action {
        module ipgpc
          name ipgpc.classify

        filter {
                name myfilter2
                daddr 10.10.8.3
                class myclass
        }

        class {
                name myclass
                next_action mark4
        }
}

action {
        name mark4
        module dlcosmk
        params {
                cos 4
                next_action continue
        global_stats true
        }
}
```

In this configuration, all traffic from `machine1` that is destined for the VLAN device on `machine2` is passed to the `dlcosmk` marker. The `mark4` marker action instructs `dlcosmk` to add a VLAN mark to datagrams of class `myclass` with a CoS of 4. The user-priority value of 4 indicates that the switch between the two machines should give controlled load forwarding to `myclass` traffic flows from `machine1`.

# `flowacct` Module

The IPQoS `flowacct` module records information about traffic flows, a process that is referred to as *flow accounting*. Flow accounting produces data that can be used for billing customers or for evaluating the amount of traffic to a particular class.

Flow accounting is optional. `flowacct` is typically the final module that metered or marked traffic flows might encounter before release onto the network stream. For an illustration of `flowacct`'s position in the Diffserv model, see Figure 32–1. For detailed technical information about `flowacct`, refer to the `flowacct`(7ipp) man page.

To enable flow accounting, you need to use the Solaris `exacct` accounting facility and the `acctadm` command, as well as `flowacct`. For the overall steps in setting up flow accounting, refer to "Setting Up Flow Accounting (Task Map)" on page 829.

## `flowacct` Parameters

The `flowacct` module gathers information about flows in a *flow table* that is composed of *flow records*. Each entry in the table contains one flow record. You cannot display a flow table.

In the IPQoS configuration file, you define the following `flowacct` parameters to measure flow records and to write the records to the flow table:

- `timer` – Defines an interval, in milliseconds, when timed-out flows are removed from the flow table and written to the file that is created by `acctadm`
- `timeout` – Defines an interval, in milliseconds, which specifies how long a packet flow must be inactive before the flow times out

  **Note –** You can configure `timer` and `timeout` to have different values.

- `max_limit` – Places an upper limit on the number of flow records that can be stored in the flow table

For an example of how `flowacct` parameters are used in the IPQoS configuration file, refer to "How to Configure Flow Control in the IPQoS Configuration File" on page 815.

## Flow Table

The `flowacct` module maintains a flow table that records all packet flows that are seen by a `flowacct` instance. A flow is identified by the following parameters, which include the `flowacct` 8–tuple:

- Source address
- Destination address
- Source port

- Destination port
- DSCP
- User ID
- Project ID
- Protocol Number

If all the parameters of the 8–tuple for a flow remain the same, the flow table contains only one entry. The `max_limit` parameter determines the number of entries that a flow table can contain.

The flow table is scanned at the interval that is specified in the IPQoS configuration file for the `timer` parameter. The default is 15 seconds. A flow "times out" when its packets are not seen by the IPQoS system for at least the `timeout` interval in the IPQoS configuration file. The default time out interval is 60 seconds. Entries that have timed out are then written to the accounting file that is created with the `acctadm` command.

## `flowacct` Records

A `flowacct` record contains the attributes described in the following table.

**TABLE 37–4** Attributes of a `flowacct` Record

| Attribute Name | Attribute Contents | Type |
|---|---|---|
| `src-addr-`*address-type* | Source address of the originator. *address-type* is either `v4` for IPv4 or `v6` for IPv6, as specified in the IPQoS configuration file. | Basic |
| `dest-addr-`*address-type* | Destination address for the packets. *address-type* is either `v4` for IPv4 or `v6` for IPv6, as specified in the IPQoS configuration file. | Basic |
| `src-port` | Source port from which the flow originated. | Basic |
| `dest-port` | Destination port number to which this flow is bound. | Basic |
| `protocol` | Protocol number for the flow. | Basic |
| `total-packets` | Number of packets in the flow. | Basic |
| `total-bytes` | Number of bytes in the flow. | Basic |
| *action-name* | Name of the `flowacct` action that recorded this flow. | Basic |
| `creation-time` | First time that a packet is seen for the flow by `flowacct`. | Extended only |
| `last-seen` | Last time that a packet of the flow was seen. | Extended only |
| `diffserv-field` | DSCP in the outgoing packet headers of the flow. | Extended only |

**TABLE 37–4** Attributes of a `flowacct` Record *(Continued)*

| Attribute Name | Attribute Contents | Type |
|---|---|---|
| `user` | Either a UNIX User ID or user name, which is obtained from the application. | Extended only |
| `projid` | Project ID, which is obtained from the application. | Extended only |

### Using `acctadm` with the `flowacct` Module

You use the `acctadm` command to create a file in which to store the various flow records that are generated by `flowacct`. `acctadm` works in conjunction with the extended accounting facility. For technical information about `acctadm`, refer to the `acctadm`(1M) man page.

The `flowacct` module observes flows and fills the flow table with flow records. `flowacct` then evaluates its parameters and attributes in the interval that is specified by `timer`. When a packet is not seen for at least the `last_seen` plus `timeout` values, the packet times out. All timed-out entries are deleted from the flow table. These entries are then written to the accounting file each time the interval that is specified in the `timer` parameter elapses.

To invoke `acctadm` for use with the `flowacct` module, use the following syntax:

`acctadm -e` *file-type* `-f` *filename* `flow`

| | |
|---|---|
| `acctadm -e` | Invokes `acctadm` with the `-e` option. The -e indicates that a resource list follows. |
| *file-type* | Specifies the attributes to be gathered. *file-type* must be replaced by either `basic` or `extended`. For a list of attributes in each file type, refer to Table 37–4. |
| `-f`*file-name* | Creates the file*file-name* to hold the flow records. |
| `flow` | Indicates that `acctadm` is to be run with IPQoS. |

# IPQoS Configuration File

This section contains full details about the parts of the IPQoS configuration file. The IPQoS boot-time activated policy is stored in the file `/etc/inet/ipqosinit.conf`. Although you can edit this file, the best practice for a new IPQoS system is to create a configuration file with a different name. Tasks for applying and debugging an IPQoS configuration are in Chapter 34, "Creating the IPQoS Configuration File (Tasks)."

The syntax of the IPQoS configuration file is shown in Example 37–3. The example uses the following conventions:

- `computer-style type` – Syntactical information that is provided to explain the parts of the configuration file. You do not type any text that appears in computer-style type.

- **bold type** – Literal text that you must type in the IPQoS configuration file. For example, you must always begin the IPQoS configuration file with **fmt_version**.
- *italic type* – Variable text that you replace with descriptive information about your configuration. For example, you must always replace *action-name* or *module-name* with information that pertains to your configuration.

**EXAMPLE 37–3** Syntax of the IPQoS Configuration File

```
file_format_version ::= fmt_version version

action_clause ::= action {
     name action-name
     module module-name
     params-clause | ""
     cf-clauses
}
action_name ::= string
module_name ::= ipgpc | dlcosmk | dscpmk | tswtclmt | tokenmt | flowacct

params_clause ::= params {
     parameters
     params-stats | ""
     }
parameters ::=    prm-name-value parameters | ""
prm_name_value ::= param-name param-value

params_stats ::= global-stats boolean

cf_clauses ::= class-clause cf-clauses |
               filter-clause cf-clauses | ""

class_clause ::= class {
     name class-name
     next_action next-action-name
     class-stats | ""
                  }
class_name  ::= string
next_action_name  ::= string
class_stats ::= enable_stats boolean
boolean ::= TRUE | FALSE

filter_clause ::= filter {
                name filter-name
                class class–name
                parameters
                }
filter_name ::= string
```

The remaining text describes each major part of the IPQoS configuration file.

## `action` Statement

You use `action` statements to invoke the various IPQoS modules that are described in "IPQoS Architecture and the Diffserv Model" on page 835.

When you create the IPQoS configuration file, you must always begin with the version number. Then, you must add the following `action` statement to invoke the classifier:

```
fmt_version 1.0

action {
    module ipgpc
    name ipgpc.classify
}
```

Follow the classifier `action` statement with a `params` clause or a `class` clause.

Use the following syntax for all other `action` statements:

```
action {
name action-name
module module-name
params-clause | ""
cf-clauses
}
```

| | |
|---|---|
| `name` *action_name* | Assigns a name to the action. |
| `module` *module_name* | Identifies the IPQoS module to be invoked, which must be one of the modules in Table 37–5. |
| *params_clause* | Can be parameters for the classifier to process, such as global statistics or the next action to process. |
| *cf_clauses* | A set of zero or more `class` clauses or `filter` clauses |

# Module Definitions

The module definition indicates which module is to process the parameters in the `action` statement. The IPQoS configuration file can include the following modules.

**TABLE 37–5** IPQoS Modules

| Module Name | Definition |
|---|---|
| `ipgpc` | IP classifier |
| `dscpmk` | Marker to be used to create DSCPs in IP packets |
| `dlcosmk` | Marker to be used with VLAN devices |
| `tokenmt` | Token bucket meter |
| `tswtclmt` | Time-sliding window meter |
| `flowacct` | Flow-accounting module |

## `class` Clause

You define a `class` clause for each class of traffic.

Use this syntax to define the remaining classes in the IPQoS configuration:

```
class {

      name class-name
      next_action next-action-name
}
```

To enable statistics collection on a particular class, you must first enable global statistics in the `ipgpc.classify` `action` statement. For more information, refer to "`action` Statement" on page 849.

Use the `enable_stats` `TRUE` statement whenever you want to turn on statistics collection for a class. If you do not need to gather statistics for a class, you can specify `enable_stats` `FALSE`. Alternatively, you can eliminate the `enable_stats` statement.

Traffic on an IPQoS-enabled network that you do not specifically define is relegated to the *default class*.

## `filter` Clause

*Filters* are made up of selectors that group traffic flows into classes. These selectors specifically define the criteria to be applied to traffic of the class that was created in the class clause. If a packet matches all selectors of the highest-priority filter, the packet is considered to be a member of the filter's class. For a complete list of selectors that you can use with the `ipgpc` classifier, refer to Table 37–1.

You define filters in the IPQoS configuration file by using a *filter clause*, which has the following syntax:

```
filter {
        name filter-name
        class class-name
        parameters (selectors)
        }
```

### params Clause

The params clause contains processing instructions for the module that is defined in the action statement. Use the following syntax for the params clause:

```
params {
            parameters
            params-stats | ""
        }
```

In the params clause, you use parameters that are applicable to the module.

The *params-stats* value in the params clause is either global_stats TRUE or global_stats FALSE. The global_stats TRUE instruction turns on UNIX style statistics for the action statement where global statistics is invoked. You can view the statistics by using the kstat command. You must enable action statement statistics before you can enable per-class statistics.

## ipqosconf Configuration Utility

You use the ipqosconf utility to read the IPQoS configuration file and to configure IPQoS modules in the UNIX kernel. ipqosconf performs the following actions:

- Applies the configuration file to the IPQoS kernel modules (ipqosconf -a *filename*)
- Lists the IPQoS configuration file currently resident in the kernel (ipqosconf -l)
- Ensures that the current IPQoS configuration is read and applied each time the machine reboots (ipqosconf -c)
- Flushes the current IPQoS kernel modules (ipqosconf -f)

For technical information, refer to the ipqosconf(1M) man page.

# Glossary

This glossary contains definitions of new terms in this book that are not in the *Sun Global Glossary* available from the `docs.sun.com` web site.

**3DES** See Triple-DES.

**address migration** Refers to the process that moves an address from one network interface to another network interface. Address migration occurs as part of failover when an interface fails, or failback when an interface is repaired.

**address pool** In Mobile IP, a set of addresses that are designated by the home network administrator for use by mobile nodes that need a home address.

**AES** Advanced Encryption Standard. A symmetric 128-bit block data encryption technique. The U.S. government adopted the Rijndael variant of the algorithm as its encryption standard in October 2000. AES replaces DES encryption as the government standard.

**agent advertisement** In Mobile IP, a message that is periodically sent by home agents and foreign agents to advertise their presence on any attached link.

**agent discovery** In Mobile IP, the process by which a mobile node determines if it has moved, its current location, and its care-of address on a foreign network.

**anycast address** An IPv6 address that is assigned to a group of interfaces (typically belonging to different nodes). A packet that is sent to an anycast address is routed to the *nearest* interface having that address. The packet's route is in compliance with the routing protocol's measure of distance.

**anycast group** A group of interfaces with the same anycast IPv6 address. The Solaris OS implementation of IPv6 does not support the creation of anycast addresses and groups. However, Solaris IPv6 nodes can send traffic to anycast groups.

**asymmetric key cryptography** An encryption system in which the sender and receiver of a message use different keys to encrypt and decrypt the message. Asymmetric keys are used to establish a secure channel for symmetric key encryption. The Diffie-Hellman protocol is an example of an asymmetric key protocol. Contrast with symmetric key cryptography.

**authentication header** An extension header that provides authentication and integrity, without confidentiality, to IP datagrams.

**autoconfiguration** The process where a host automatically configures its IPv6 address from the site prefix and the local MAC address.

**bidirectional tunnel** A tunnel that can transmit datagrams in both directions.

**binding table** In Mobile IP, a home agent table that associates a home address with a care-of address, including remaining lifetime and time granted.

**Blowfish** A symmetric block cipher algorithm that takes a variable-length key from 32 bits to 448 bits. Its author, Bruce Schneier, claims that Blowfish is optimized for applications where the key does not change often.

**broadcast address** IPv4 network addresses with the host portion of the address having all zeroes (10.50.0.0) or all one bits (10.50.255.255). A packet that is sent to a broadcast address from a machine on the local network is delivered to all machines on that network.

**CA** See certificate authority (CA).

**care-of address** A mobile node's temporary address that is used as a tunnel exit point when the mobile node is connected to a foreign network.

**certificate authority (CA)** A trusted third-party organization or company that issues digital certificates used to create digital signatures and public-private key pairs. The CA guarantees the identity of the individual who is granted the unique certificate.

**certificate revocation list (CRL)** A list of public key certificates that have been revoked by a CA. CRLs are stored in the CRL database that is maintained through IKE.

**class** In IPQoS, a group of network flows that share similar characteristics. You define classes in the IPQoS configuration file.

**classless inter-domain routing (CIDR) address** An IPv4 address format that is not based on network classes (Class A, B, and C). CIDR addresses are 32 bits in length. They use the standard IPv4 dotted decimal notation format, with the addition of a network prefix. This prefix defines the network number and the network mask.

**data address** An IP address which can be used as the source or destination address for data. Data addresses are part of an IPMP group and can be used to send and receive traffic on any interface in the group. Moreover, the set of data addresses in an IPMP group can be used continuously provided that one interface in the group is functioning.

**datagram** See IP datagram.

**`DEPRECATED` address** An IP address that cannot be used as the source address for data in an IPMP group. Typically, IPMP test addresses are `DEPRECATED`. However, any address can be marked `DEPRECATED` to prevent the address from being used as a source address.

**DES** Data Encryption Standard. A symmetric-key encryption method developed in 1975 and standardized by ANSI in 1981 as ANSI X.3.92. DES uses a 56-bit key.

**Diffie-Hellman protocol** Also known as public key cryptography. An asymmetric cryptographic key agreement protocol that was developed by Diffie and Hellman in 1976. The protocol enables two users to exchange a secret key over an insecure medium without any prior secrets. Diffie-Hellman is used by the IKE protocol.

**diffserv model** Internet Engineering Task Force architectural standard for implementing differentiated services on IP networks. The major modules are classifier, meter, marker, scheduler, and dropper. IPQoS implements the classifier, meter, and marker modules. The diffserv model is described in RFC 2475, *An Architecture for Differentiated Services*.

**digital signature** A digital code that is attached to an electronically transmitted message that uniquely identifies the sender.

**domain of interpretation (DOI)** A DOI defines data formats, network traffic exchange types, and conventions for naming security-relevant information. Security policies, cryptographic algorithms, and cryptographic modes are examples of security-relevant information.

**DS codepoint (DSCP)** A 6-bit value that, when included in the DS field of an IP header, indicates how a packet must be forwarded.

**DSA** Digital Signature Algorithm. A public key algorithm with a variable key size from 512 to 4096 bits. The U.S. Government standard, DSS, goes up to 1024 bits. DSA relies on SHA-1 for input.

**dual stack** A TCP/IP protocol stack with both IPv4 and IPv6 at the network layer, with the rest of the stack being identical. When you enable IPv6 during Solaris OS installation, the host receives the dual-stack version of TCP/IP.

**dynamic packet filter** See stateful packet filter.

**dynamic reconfiguration (DR)** A feature that allows you to reconfigure a system while the system is running, with little or no impact on ongoing operations. Not all Sun platforms support DR. Some Sun platforms might only support DR of certain types of hardware such as NICs.

**encapsulating security payload (ESP)** An extension header that provides integrity and confidentiality to datagrams. ESP is one of the five components of the IP Security Architecture (IPsec).

**encapsulation** The process of a header and payload being placed in the first packet, which is subsequently placed in the second packet's payload.

**failback** The process of switching back network access to an interface that has its repair detected.

**failover** The process of switching network access from a failed interface to a good physical interface. Network access includes IPv4 unicast, multicast, and broadcast traffic, as well as IPv6 unicast and multicast traffic.

**failure detection** The process of detecting when an interface or the path from an interface to an Internet layer device no longer works. IP network multipathing (IPMP) includes two types of failure detection: link based (default) and probe based (optional).

**filter** A set of rules that define the characteristics of a class in the IPQoS configuration file. The IPQoS system selects for processing any traffic flows that conform to the filters in its IPQoS configuration file. See packet filter.

**firewall** Any device or software that isolates an organization's private network or intranet from the Internet, thus protecting it from external intrusions. A firewall can include packet filtering, proxy servers, and NAT (network address translation).

**flow accounting** In IPQoS, the process of accumulating and recording information about traffic flows. You establish flow accounting by defining parameters for the `flowacct` module in the IPQoS configuration file.

**foreign agent** A router or server on the foreign network that the mobile node visits.

**foreign network** Any network other than the mobile node's home network.

**forward tunnel** A tunnel that starts at the home agent and terminates at the mobile node's care-of address.

**Generic Routing Encapsulation (GRE)** An optional form of tunneling that can be supported by home agents, foreign agents, and mobile nodes. GRE enables a packet of any network-layer protocol to be encapsulated within a delivery packet of any other (or the same) network-layer protocol.

**hash value** A number that is generated from a string of text. Hash functions are used to ensure that transmitted messages have not been tampered with. MD5 and SHA-1 are examples of one-way hash functions.

**header** See IP header.

**HMAC** Keyed hashing method for message authentication. HMAC is a secret key authentication algorithm. HMAC is used with an iterative cryptographic hash function, such as MD5 or SHA-1, in combination with a secret shared key. The cryptographic strength of HMAC depends on the properties of the underlying hash function.

**home address** An IP address that is assigned for an extended period to a mobile node. The address remains unchanged when the node is attached elsewhere on the Internet or an organization's network.

**home agent** A router or server on the home network of a mobile node.

**home network** A network that has a network prefix that matches the network prefix of a mobile node's home address.

**hop** A measure that is used to identify the number of routers that separate two hosts. If three routers separate a source and destination, the hosts are four hops away from each other.

**host** A system that does not perform packet forwarding. Upon installation of the Solaris OS, a system becomes a host by default, that is, the system cannot forward packets. A host typically has one physical interface, although it can have multiple interfaces.

**ICMP** Internet Control Message Protocol. Used to handle errors and exchange control messages.

**ICMP echo request packet** A packet sent to a machine on the Internet to solicit a response. Such packets are commonly known as "ping" packets.

**IKE** Internet Key Exchange. IKE automates the provision of authenticated keying material for IPsec security association (SA)s.

**Internet Protocol (IP)** The method or protocol by which data is sent from one computer to another on the Internet.

**IP** See Internet Protocol (IP), IPv4, IPv6.

**IP datagram** A packet of information that is carried over IP. An IP datagram contains a header and data. The header includes the addresses of the source and the destination of the datagram. Other fields in the header help identify and recombine the data with accompanying datagrams at the destination.

**IP header** Twenty bytes of data that uniquely identify an Internet packet. The header includes source and destination addresses for the packet. An option exists within the header to allow further bytes to be added.

**IP in IP encapsulation** The mechanism for tunneling IP packets within IP packets.

**IP link** A communication facility or medium over which nodes can communicate at the link layer. The link layer is the layer immediately below IPv4/IPv6. Examples include Ethernets (simple or bridged) or ATM networks. One or more IPv4 subnet numbers or prefixes are assigned to an IP link. A subnet number or prefix cannot be assigned to more than one IP link. In ATM LANE, an IP link is a single emulated LAN. When you use ARP, the scope of the ARP protocol is a single IP link.

**IP stack** TCP/IP is frequently referred to as a "stack." This refers to the layers (TCP, IP, and sometimes others) through which all data passes at both client and server ends of a data exchange.

**IPMP group** IP multipathing group, composed of a set of network interfaces with a set of data addresses that are treated as interchangeable by the system to improve network availability and utilization. The IPMP group, including all its underlying IP interfaces and data addresses, is represented by an IPMP interface.

**IPQoS** A software feature that provides an implementation of the diffserv model standard, plus flow accounting and 802.1 D marking for virtual LANs. Using IPQoS, you can provide different levels of network services to customers and applications, as defined in the IPQoS configuration file.

**IPsec** IP security. The security architecture that provides protection for IP datagrams.

**IPv4** Internet Protocol, version 4. IPv4 is sometimes referred to as IP. This version supports a 32-bit address space.

**IPv6** Internet Protocol, version 6. IPv6 supports a 128-bit address space.

**key management** The way in which you manage security association (SA)s.

**keystore name** The name that an administrator gives to the storage area, or keystore, on a network interface card (NIC). The keystore name is also called the token or the token ID.

**link layer** The layer immediately below IPv4/IPv6.

**link-local address** In IPv6, a designation that is used for addressing on a single link for purposes such as automatic address configuration. By default, the link-local address is created from the system's MAC address.

**load spreading** The process of distributing inbound or outbound traffic over a set of interfaces. With load spreading, higher throughput is achieved. Load spreading occurs only when the network traffic is flowing to multiple destinations that use multiple connections. Two types of load spreading exists: inbound load spreading for inbound traffic and outbound load spreading for outbound traffic.

**local-use address** A unicast address that has only local routability scope (within the subnet or within a subscriber network). This address also can have a local or global uniqueness scope.

**marker** 1. A module in the diffserv architecture and IPQoS that marks the DS field of an IP packet with a value that indicates how the packet is to be forwarded. In the IPQoS implementation, the marker module is `dscpmk`.

| | |
|---|---|
| | 2. A module in the IPQoS implementation that marks the virtual LAN tag of an Ethernet datagram with a user priority value. The user priority value indicates how datagrams are to be forwarded on a network with VLAN devices. This module is called `dlcosmk`. |
| **MD5** | An iterative cryptographic hash function that is used for message authentication, including digital signatures. The function was developed in 1991 by Rivest. |
| **message authentication code (MAC)** | MAC provides assurance of data integrity and authenticates data origin. MAC does not protect against eavesdropping. |
| **meter** | A module in the diffserv architecture that measures the rate of traffic flow for a particular class. The IPQoS implementation includes two meters, `tokenmt` and `tswtclmt`. |
| **minimal encapsulation** | An optional form of IPv4 in IPv4 tunneling that can be supported by home agents, foreign agents, and mobile nodes. Minimal encapsulation has 8 or 12 bytes less of overhead than does IP in IP encapsulation. |
| **mobile node** | A host or router that can change its point of attachment from one network to another network while maintaining all existing communications by using its IP home address. |
| **mobility agent** | Either a home agent or a foreign agent. |
| **mobility binding** | The association of a home address with a care-of address, along with the remaining lifetime of that association. |
| **mobility security association** | A collection of security measures, such as an authentication algorithm, between a pair of nodes, which are applied to Mobile IP protocol messages that are exchanged between the two nodes. |
| **MTU** | Maximum Transmission Unit. The size, given in octets, that can be transmitted over a link. For example, the MTU of an Ethernet is 1500 octets. |
| **multicast address** | An IPv6 address that identifies a group of interfaces in a particular way. A packet that is sent to a multicast address is delivered to all of the interfaces in the group. The IPv6 multicast address has similar functionality to the IPv4 broadcast address. |
| **multihomed host** | A system that has more than one physical interface and that does not perform packet forwarding. A multihomed host can run routing protocols. |
| **NAT** | See network address translation. |
| **neighbor advertisement** | A response to a neighbor solicitation message or the process of a node sending unsolicited neighbor advertisements to announce a link-layer address change. |
| **neighbor discovery** | An IP mechanism that enables hosts to locate other hosts that reside on an attached link. |
| **neighbor solicitation** | A solicitation that is sent by a node to determine the link-layer address of a neighbor. A neighbor solicitation also verifies that a neighbor is still reachable by a cached link-layer address. |
| **Network Access Identifier (NAI)** | A designation that uniquely identifies the mobile node in the format of user@domain. |
| **network address translation** | NAT. The translation of an IP address used within one network to a different IP address known within another network. Used to limit the number of global IP addresses that are needed. |

**network interface card (NIC)** Network adapter card that is an interface to a network. Some NICs can have multiple physical interfaces, such as the `qfe` card.

**node** In IPv6, any system that is IPv6-enabled, whether a host or a router.

**outcome** The action to take as a result of metering traffic. The IPQoS meters have three outcomes, red, yellow, and green, which you define in the IPQoS configuration file.

**packet** A group of information that is transmitted as a unit over communications lines. Contains an IP header plus a payload.

**packet filter** A firewall function that can be configured to allow or disallow specified packets through a firewall.

**packet header** See IP header.

**payload** The data that is carried in a packet. The payload does not include the header information that is required to get the packet to its destination.

**per-hop behavior (PHB)** A priority that is assigned to a traffic class. The PHB indicates the precedence which flows of that class have in relation to other traffic classes.

**perfect forward secrecy (PFS)** In PFS, the key that is used to protect transmission of data is not used to derive additional keys. Also, the source of the key that is used to protect data transmission is never used to derive additional keys.

PFS applies to authenticated key exchange only. See also Diffie-Hellman protocol.

**physical interface** A system's attachment to a link. This attachment is often implemented as a device driver plus a network interface card (NIC). Some NICs can have multiple points of attachment, for example, `qfe`.

**PKI** Public Key Infrastructure. A system of digital certificates, Certificate Authorities, and other registration authorities that verify and authenticate the validity of each party involved in an Internet transaction.

**plumb** The act of opening a device that is associated with a physical interface name. When an interface is plumbed, streams are set up so that the IP protocol can use the device. You use the `ifconfig` command to plumb an interface during a system's current session.

**private address** An IP address that is not routable through the Internet. Private addresses can used by internal networks on hosts that do not require Internet connectivity. These addresses are defined in Address Allocation for Private Internets (`http://www.ietf.org/rfc/rfc1918.txt?number=1918`) and often referred to as "1918" addresses.

**protocol stack** See IP stack.

**proxy server** A server that sits between a client application, such as a Web browser, and another server. Used to filter requests – to prevent access to certain web sites, for instance.

**public key cryptography** A cryptographic system that uses two different keys. The public key is known to everyone. The private key is known only to the recipient of the message. IKE provides public keys for IPsec.

**redirect** In a router, to inform a host of a better first-hop node to reach a particular destination.

**registration** The process by which a mobile node registers its care-of address with its home agent and foreign agent when it is away from home.

**repair detection** The process of detecting when a NIC or the path from the NIC to some layer-3 device starts operating correctly after a failure.

**replay attack** In IPsec, an attack in which a packet is captured by an intruder. The stored packet then replaces or repeats the original at a later time. To protect against such attacks, a packet can contain a field that increments during the lifetime of the secret key that is protecting the packet.

**reverse tunnel** A tunnel that starts at the mobile node's care-of address and terminates at the home agent.

**router** A system that usually has more than one interface, runs routing protocols, and forwards packets. You can configure a system with only one interface as a router if the system is the endpoint of a PPP link.

**router advertisement** The process of routers advertising their presence together with various link and Internet parameters, either periodically or in response to a router solicitation message.

**router discovery** The process of hosts locating routers that reside on an attached link.

**router solicitation** The process of hosts requesting routers to generate router advertisements immediately, rather than at their next scheduled time.

**RSA** A method for obtaining digital signatures and public key cryptosystems. The method was first described in 1978 by its developers, Rivest, Shamir, and Adleman.

**SA** See security association (SA).

**SADB** Security Associations Database. A table that specifies cryptographic keys and cryptographic algorithms. The keys and algorithms are used in the secure transmission of data.

**SCTP** See streams control transport protocol.

**security association (SA)** An association that specifies security properties from one host to a second host.

**security parameter index (SPI)** An integer that specifies the row in the security associations database (SADB) that a receiver should use to decrypt a received packet.

**security policy database (SPD)** Database that specifies the level of protection to apply to a packet. The SPD filters IP traffic to determine whether a packet should be discarded, should be passed in the clear, or should be protected with IPsec.

**selector** The element that specifically defines the criteria to be applied to packets of a particular class in order to select that traffic from the network stream. You define selectors in the filter clause of the IPQoS configuration file.

**SHA-1** Secure Hashing Algorithm. The algorithm operates on any input length less than $2^{64}$ to produce a message digest. The SHA-1 algorithm is input to DSA.

**site-local-use address** A designation that is used for addressing on a single site.

**smurf attack** To use ICMP echo request packets directed to an IP broadcast address or multiple broadcast addresses from remote locations to create severe network congestion or outages.

**sniff** To eavesdrop on computer networks – frequently used as part of automated programs to sift information, such as clear-text passwords, off the wire.

**SPD** See security policy database (SPD).

**SPI** See security parameter index (SPI).

**spoof** To gain unauthorized access to a computer by sending a message to it with an IP address indicating that the message is coming from a trusted host. To engage in IP spoofing, a hacker must first use a variety of techniques to find an IP address of a trusted host and then modify the packet headers so that it appears that the packets are coming from that host.

**stack** See IP stack.

**standby** A physical interface that is not used to carry data traffic unless some other physical interface has failed.

**stateful packet filter** A packet filter that can monitor the state of active connections and use the information obtained to determine which network packets to allow through the firewall. By tracking and matching requests and replies, a stateful packet filter can screen for a reply that doesn't match a request.

**stateless autoconfiguration** The process of a host generating its own IPv6 addresses by combining its MAC address and an IPv6 prefix that is advertised by a local IPv6 router.

**stream control transport protocol** A transport layer protocol that provides connection-oriented communications in a manner similar to TCP. Additionally, SCTP supports multihoming, in which one of the endpoints of the connection can have more than one IP address.

**symmetric key cryptography** An encryption system in which the sender and receiver of a message share a single, common key. This common key is used to encrypt and decrypt the message. Symmetric keys are used to encrypt the bulk of data transmission in IPsec. DES is one example of a symmetric key system.

**TCP/IP** TCP/IP (Transmission Control Protocol/Internet Protocol) is the basic communication language or protocol of the Internet. It can also be used as a communications protocol in a private network (either an intranet or an extranet).

**test address** An IP address in an IPMP group which must be used as the source or destination address for probes, and must not be used as a source or destination address for data traffic. Compare with

**Triple-DES** Triple-Data Encryption Standard. A symmetric-key encryption method. Triple-DES requires a key length of 168 bits. Triple-DES is also written as 3DES.

**tunnel** The path that is followed by a datagram while it is encapsulated. See encapsulation.

**unicast address** An IPv6 address that identifies a single interface of an IPv6-enabled node. The parts of the unicast address are site prefix, subnet ID, and interface ID.

**user-priority** A 3-bit value that implements class-of-service marks, which define how Ethernet datagrams are forwarded on a network of VLAN devices.

**virtual LAN (VLAN) device** Network interfaces that provide traffic forwarding at the Ethernet (data link) level of the IP protocol stack.

**virtual private network (VPN)** A single, secure, logical network that uses tunnels across a public network such as the Internet.

# Index


## Numbers and Symbols

* (asterisk), wildcard in `bootparams` database, 250
> prompt
  `ikeadm` command mode, 561
  `ipseckey` command mode, 504
"r" commands, in UNIX, 43
3DES encryption algorithm
  IPsec and, 486
  key length, 505
6to4 address
  format, 258
  host address, 259
6to4 advertisement, 193
6to4 prefix
  `/etc/inet/ndpd.conf` advertisement, 194
  explanation of parts, 259
6to4 pseudo-interface configuration, 192
6to4 relay router
  in a 6to4 tunnel, 270
  security issues, 232, 291-293
  tunnel configuration tasks, 195, 196
  tunnel topology, 292
6to4 router configuration
  examples, 194
  tasks, 192
6to4 tunnels
  6to4 relay router, 195
  definition, 192
  known problems, 232
  packet flow, 291, 292
  sample topology, 290
`6to4relay` command, 195
`6to4relay` command *(Continued)*
  definition, 270
  examples, 270
  syntax, 270
  tunnel configuration tasks, 195

## A

`-A` option
  `ikecert certlocal` command, 566
  `ikecert` command, 600
`-a` option
  `ikecert certdb` command, 567, 573
  `ikecert certrldb` command, 583
  `ikecert` command, 577
  `ipsecconf` command, 499, 562
AAAA records, 198, 293
accelerating
  IKE computations, 549, 592
`acctadm` command, for flow accounting, 767, 831, 847
ACK segment, 46
`action` statement, 849
active-active interface configuration, IPMP, 725
active rule sets, *See* Solaris IP Filter
active-standby interface configuration, IPMP, 725
adding
  CA certificates (IKE), 571-577
  IPsec SAs, 498, 503-507
  keys manually (IPsec), 503-507
  preshared keys (IKE), 559-563
  public key certificates (IKE), 571-577

adding *(Continued)*
self-signed certificates (IKE), 565
address autoconfiguration
definition, 80, 81
enabling, on an IPv6 node, 173, 174, 176
IPv6, 275, 278
address pools
appending, 643
configuring, 613-614
overview, 613-614
removing, 642
viewing, 642
viewing statistics, 646-647
address resolution, in IPv6, 80
Address Resolution Protocol (ARP)
comparison to Neighbor Discovery protocol, 282-283
definition, 41
`Address` section
labels and values, 706
Mobile IP configuration file, 704, 706-709
NAI labels and values, 708
Node-Default labels and values, 709
private addresses, 706, 707
addresses
6to4 format, 258
CIDR format, 59
data addresses, IPMP, 721
default address selection, 225-228
displaying addresses of all interfaces, 207
Ethernet addresses
`ethers` database, 247, 250
IPv4 format, 58
IPv4 netmask, 242
IPv6, 6to4 format, 192
IPv6 global unicast, 77
IPv6 link-local, 78
loopback address, 238
multicast, in IPv6, 260-261
temporary, in IPv6, 181-184
test addresses, IPMP, 722-723
administrative model, 418
administrative subdivisions, 65
`Advertisements` section
labels and values, 702
Mobile IP configuration file, 701-703
`AdvertiseOnBcast` label, 682, 702
`AdvFrequency` label, 682, 702
`AdvInitCount` label, 703
`AdvLifetime` label, 682, 686, 702
`AdvLimitUnsolicited` label, 703
AES encryption algorithm, IPsec and, 486
agent advertisement
Mobile IP, 666
over dynamic interfaces, 666, 702
agent discovery, Mobile IP, 666-667
agent solicitation, Mobile IP, 665, 666, 667
aggregations
creating, 165-168
definition, 161
features, 162
load balancing policy, 164
modifying, 168-169
removing interfaces, 169
requirements, 165
topologies
back-to-back, 164
basic, 162
with switch, 163
AH, *See* authentication header (AH)
anonymous FTP program, description, 42
anonymous login name, 42
anycast addresses, 195
definition, 79
anycast groups, 6to4 relay router, 195
application layer
OSI, 38
packet life cycle
receiving host, 48
sending host, 45
TCP/IP, 42, 44
description, 39, 42
file services, 44
name services, 43
network administration, 44
routing protocols, 44
standard TCP/IP services, 42, 43

application layer, TCP/IP *(Continued)*
UNIX "r" commands, 43
application server, configuring for IPQoS, 808
assured forwarding (AF), 770, 842
AF codepoints table, 842
for a marker `action` statement, 803
asterisk (*), wildcard in `bootparams` database, 250
ATM, IPMP support for, 736
ATM support, IPv6 over, 295
`auth_algs` security option, `ifconfig` command, 542-543
authentication algorithms
IKE, 600
specifying for IPsec, 542
authentication header (AH)
IPsec protection mechanism, 484-486
protecting IP datagram, 484
protecting IP packets, 478
security considerations, 485
automatic tunnels, transition to IPv6, 286
autonomous system (AS), *See* network topology

## B

bandwidth regulation, 763
planning, in the QoS policy, 781
`BaseAddress` label, 683, 705
BGP, *See* routing protocols
binary to decimal conversion, 243
binding table
home agent, 692, 694
Mobile IP, 711
Blowfish encryption algorithm, IPsec and, 486
booting, network configuration server booting protocols, 96
BOOTP protocol
and DHCP, 299
supporting clients with DHCP service, 367
BOOTP relay agent
configuring
with DHCP Manager, 333
with `dhcpconfig -R`, 337
hops, 356
`bootparams` database
corresponding name service files, 247
overview, 249
wildcard entry, 250
Bootparams protocol, 96
border router, 114
boundary router, in 6to4 site, 291
broadcast address, 705
broadcast datagrams, Mobile IP, 675
BSD-based operating systems
`/etc/inet/hosts` file link, 237
`/etc/inet/netmasks` file link, 243
bypassing
IPsec on LAN, 516, 526
IPsec policy, 487

## C

`-c` option, `in.iked` daemon, 556
care-of address
acquiring, 667
colocated, 665, 667, 672, 675
foreign agent, 667, 670, 673
Mobile IP, 662
mobile node location, 664
mobile node registration, 670
mobility agents, 663
sharing, 667
state information, 712
`cert_root` keyword
IKE configuration file, 574, 579
`cert_trust` keyword
IKE configuration file, 568, 579
`ikecert` command and, 600
certificate requests
from CA, 572
on hardware, 578
use, 601
certificate revocation lists, *See* CRLs
certificates
adding to database, 573
creating self-signed (IKE), 565
description, 573
from CA, 573

certificates *(Continued)*
from CA on hardware, 580
ignoring CRLs, 575
IKE, 548
in `ike/config` file, 579
listing, 568
requesting
from CA, 572
on hardware, 578
storing
IKE, 601
on computer, 565
on hardware, 549, 592
`Challenge` label, 682, 704
changing, privilege level in IKE, 562
Changing IKE Transmission Parameters (Task Map), 594
ciphers, *See* encryption algorithms
class A, B, and C network numbers, 56, 60
class A network numbers
description, 254
IPv4 address space division, 60
range of numbers available, 60
class B network numbers
description, 255
IPv4 address space division, 60
range of numbers available, 60
class C network numbers
description, 255
IPv4 address space division, 60
range of numbers available, 60
`class` clause, in the IPQoS configuration file, 799
class clause, in the IPQoS configuration file, 850
class of service (CoS) mark, 767
classes, 765
defining, in the IPQoS configuration file, 806, 811
selectors, list of, 836
syntax of `class` clause, 850
classes of service, *See* classes
classifier module, 765
`action` statement, 798
functions of the classifier, 836
client configuration, 418
client ID, 419
colocated care-of address, 665, 672, 675
acquiring, 667
color awareness, 766, 839
commands
IKE, 599-602
`ikeadm` command, 550, 561, 597, 598-599
`ikecert` command, 550, 597, 599
`in.iked` daemon, 597
IPsec
`in.iked` command, 483
`ipsecalgs` command, 486, 540
`ipsecconf` command, 491, 499, 537-538
`ipseckey` command, 492, 504, 541-542
list of, 491-492
security considerations, 541-542
`snoop` command, 542, 544
computations
accelerating IKE in hardware, 549, 592, 593-594
configuration files
creating for Solaris IP Filter, 651-652
IPv6
`/etc/inet/hostname6.`*interface* file, 267-268
`/etc/inet/ipaddrsel.conf` file, 268
`/etc/inet/ndpd.conf` file, 263-267, 266
Solaris IP Filter examples, 608
TCP/IP networks
`/etc/defaultdomain` file, 237
`/etc/defaultrouter` file, 237
`/etc/hostname.`*interface* file, 236
`/etc/nodename` file, 104, 236
`hosts` database, 237, 239
`netmasks` database, 241
configuring
address pools, 613-614
DHCP client, 417
DHCP service, 329
IKE, 553
`ike/config` file, 598
IKE with CA certificates, 571-577
IKE with certificates on hardware, 577-580
IKE with mobile systems, 583-591
IKE with public key certificates, 564, 565-571
IKE with self-signed certificates, 565-571
interfaces manually, for IPv6, 172-174

configuring *(Continued)*
IPsec, 537-538
IPsec on LAN, 519, 529
`ipsecinit.conf` file, 538-539
IPv6-enabled routers, 177
NAT rules, 612-613
network configuration server, 102
network security with a role, 508-509
packet filtering rules, 609-611
routers, 253
network interfaces, 115, 118
overview, 116
TCP/IP configuration files, 235
`/etc/defaultdomain` file, 237
`/etc/defaultrouter` file, 237
`/etc/hostname.interface` file, 236
`/etc/nodename` file, 104, 236
`hosts` database, 237, 239
`netmasks` database, 241
TCP/IP configuration modes
local files mode, 95, 102
mixed configurations, 97
network client mode, 105
sample network, 97
TCP/IP networks
configuration files, 235
local files mode, 102
network clients, 104
network databases, 245, 247, 249
`nsswitch.conf` file, 247, 249
prerequisites, 94
standard TCP/IP services, 132
VPN in transport mode with IPsec, 524-530
VPN in tunnel mode with IPsec, 509, 514-520
VPN protected by IPsec, 514-520
Configuring IKE (Task Map), 553
Configuring IKE for Mobile Systems (Task Map), 583
Configuring IKE to Find Attached Hardware (Task Map), 591
Configuring IKE With Preshared Keys (Task Map), 554
Configuring IKE With Public Key Certificates (Task Map), 564
connectivity, ICMP protocol reports of failures, 41
converting DHCP data store, 408-410
CRC (cyclical redundancy check) field, 47
creating
certificate requests, 572
DHCP macros, 392
DHCP options, 399
IPsec SAs, 498, 503-507
`ipsecinit.conf` file, 497
security parameter index (SPI), 502
security-related role, 508-509
self-signed certificates (IKE), 565
CRLs
accessing from central location, 581
ignoring, 575
`ike/crls` database, 602
`ikecert certrldb` command, 601
listing, 581
cyclical redundancy check (CRC) field, 47

## D

`-D` option
`ikecert certlocal` command, 566
`ikecert` command, 600
daemons
`in.iked` daemon, 546, 550, 597
`in.mpathd` daemon, 718
`in.ndpd` daemons, 275
`in.ripngd` daemon, 178, 276
`in.routed` routing daemon, 130
`in.tftpd` daemon, 102
`inetd` Internet services, 244
network configuration server booting protocols, 96
data addresses, IPMP, definition, 721
data communications, 44, 48
packet life cycle, 45, 48
data encapsulation
definition, 45
TCP/IP protocol stack and, 45, 48
data-link layer
framing, 47
OSI, 39
packet life cycle
receiving host, 47

data-link layer, packet life cycle *(Continued)*
sending host, 47
TCP/IP, 39, 40
databases
IKE, 599-602
`ike/crls` database, 601, 602
`ike.privatekeys` database, 600, 602
`ike/publickeys` database, 601, 602
security associations database (SADB), 540
security policy database (SPD), 479
datagrams
IP, 478
IP header, 47
IP protocol formatting, 40
packet process, 47
UDP protocol functions, 42
deactivating Solaris IP Filter, 625, 630-631
decimal to binary conversion, 243
default address selection, 268-270
definition, 225-228
IPv6 address selection policy table, 226-227
default mobile node
Mobile IP `Address` section, 684, 709
default router
configuration example, 119
definition, 115
`defaultdomain` file
deleting for network client mode, 104
description, 237
local files mode configuration, 101
`defaultrouter` file
automatic router protocol selection and, 128
description, 237
local files mode configuration, 101
deleting
DHCP options, 404
IPsec SAs, 504
`deprecated` attribute, `ifconfig` command, 723
deregistering
Mobile IP, 666, 670, 671
DES encryption algorithm, IPsec and, 486
designing the network
domain name selection, 64
IP addressing scheme, 55, 62
designing the network *(Continued)*
naming hosts, 63
overview, 55
subnetting, 241
DHCP client
administration, 426
client ID, 372
definition, 312
disabling, 425-426
displaying interface status, 427
dropping IP address, 427
enabling, 425
event scripts, 436-439
extending lease, 427
host name
specifying, 430-431
host name generation, 323
incorrect configuration, 456-457
logical interfaces, 429
multiple network interfaces, 429
name services, 355
network information without lease, 407-408, 427
on diskless client systems, 406
option information, 405
parameters, 428-429
releasing IP address, 427
running in debugging mode
sample output, 449
running programs with, 436-439
shutdown, 424
starting, 427
startup, 422
testing interface, 427
troubleshooting, 447
unconfiguring, 425-426
DHCP command-line utilities, 307
privileges, 343
DHCP Configuration Wizard
description, 330
for BOOTP relay agent, 333
DHCP data store
choosing, 320
converting, 408-410
exporting data, 412, 413-414

DHCP data store *(Continued)*
importing data, 414
modifying imported data, 415, 416
moving data between servers, 410-416
overview, 305
DHCP events, 436-439
DHCP lease
and reserved IP addresses, 325
dynamic and permanent, 325
expiration date, 373
negotiation, 321
policy, 321
reserved IP addresses, 373
time, 321
type, 373
DHCP macros
automatic processing, 311
categories, 311
client class macros, 311
client ID macros, 311
configuration, 372
creating, 392
default, 324
deleting, 395
Locale macro, 331
modifying, 388
network address macro, 311, 331
network booting, 406
order processed, 312
overview, 311
server macro, 331
size limit, 312
working with, 385
DHCP Manager
description, 306
features, 327
menus, 341
starting, 342
stopping, 343
window and tabs, 340
DHCP network tables
created during server configuration, 331
description, 306
removing when unconfiguring, 334
DHCP Network Wizard, 361
DHCP networks
adding to DHCP service, 360
modifying, 363
removing from DHCP service, 365
working with, 358-367
DHCP options
creating, 399
deleting, 404
modifying, 402
overview, 310
properties, 397
working with, 396
DHCP protocol
advantages in Solaris implementation, 300
overview, 299
sequence of events, 301
DHCP server
configuration
information gathered, 318
overview, 308
configuring
`dhcpconfig` command, 336-337
with DHCP Manager, 330
data store, 305
enabling to update DNS, 352-353
functions, 304
how many to configure, 317
management, 305
options, 346
DHCP Manager, 356-357
`dhcpconfig` command, 357-358
planning for multiple servers, 326
running in debugging mode, 448-449
sample output, 450-453
selecting, 320
troubleshooting, 441
DHCP service
adding networks to, 360
cache offer time, 356
enabling and disabling
DHCP Manager, 345
`dhcpconfig` command, 345
effects of, 344

DHCP service *(Continued)*
error messages, 444, 452
IP address allocation, 309
IP addresses
adding, 374
modifying properties, 377
removing, 380
reserving for client, 383
unusable, 380
logging
overview, 348
transactions, 349
modifying service options, 346
network configuration overview, 310
network interface monitoring, 358-359
network topology, 316
planning, 315
Service Management Facility, 346
Solaris network boot and install, 405-406
starting and stopping
DHCP Manager, 344
effects of, 344
supporting BOOTP clients, 367
unconfiguring, 334
with DHCP Manager, 335
WAN boot installation support, 405
`dhcpagent` daemon, 422
debugging mode, 448
parameter file, 470
`dhcpconfig` command
description, 308, 461
`dhcpinfo` command, description, 462
`dhcpmgr` command, description, 462
`dhcpsvc.conf` file, 469
`dhcptab` table, 331
description, 469
overview, 305
reading automatically, 356
removing when unconfiguring, 334
`dhcptags` file, 471
DHCPv4 client, management of network interface, 423
DHCPv4 versus DHCPv6, 418
DHCPv6, client name, 419
DHCPv6 administrative model, 418
DHCPv6 client, management of network interface, 423
DHCPv6 versus DHCPv4, 418
`dhtadm` command
creating macros with, 392
creating options with, 399
deleting macros with, 395
deleting options with, 404
description, 308, 461
modifying macros with, 388
modifying options with, 402
differentiated services, 759
differentiated services model, 764
network topologies, 774
providing different classes of service, 764
Diffserv-aware router
evaluating DS codepoints, 842
planning, 779
Diffserv model
classifier module, 765
flow example, 767
IPQoS implementation, 764, 766, 767
marker modules, 766
meter modules, 766
digital signatures
DSA, 600
RSA, 601
directories
certificates (IKE), 601
`/etc/inet`, 550
`/etc/inet/ike`, 550
`/etc/inet/publickeys`, 601
`/etc/inet/secret`, 550
`/etc/inet/secret/ike.privatekeys`, 600
preshared keys (IKE), 599
private keys (IKE), 600
public keys (IKE), 601
directory name (DN), for accessing CRLs, 581
diskless clients, DHCP support of, 406
displaying, IPsec policy, 501
`dladm` command
configuring a VLAN, 160-161
displaying status, 149
for checking aggregation status, 166
for creating an aggregation, 166

dladm command *(Continued)*
for modifying an aggregation, 168
removing interfaces from an aggregation, 169
dlcosmk marker, 767
planning datagram forwarding, 787
user priority values, table of, 843
VLAN tags, 843
domain name system (DNS)
description, 43
domain name registration, 38
enabling dynamic updates by DHCP server, 352-353
extensions for IPv6, 293
network databases, 64, 246
preparing, for IPv6 support, 88-89
reverse zone file, 197
selecting as name service, 64
zone file, 197
domain names
/etc/defaultdomain file, 101, 104, 237
registering, 38
selecting, 64
top-level domains, 64
dotted-decimal format, 58
dropped or lost packets, 41, 216
DS codepoint (DSCP), 767, 769
AF forwarding codepoint, 770, 842
color-awareness configuration, 840
configuring, on a diffserv router, 819, 841
defining, in the IPQoS configuration file, 802
dscp_map parameter, 843
EF forwarding codepoint, 770, 841
PHBs and the DSCP, 769
planning, in the QoS policy, 787
dscpmk marker, 767
invoking, in a marker action statement, 802, 808, 814, 817
PHBs for packet forwarding, 841
planning packet forwarding, 786
DSS authentication algorithm, 600
dual-stack protocols, 86, 262-263
duplicate address detection
algorithm, 281
DHCP service, 356
duplicate address detection *(Continued)*
IPv6, 80
Dynamic Host Configuration Protocol, *See* DHCP protocol
dynamic interfaces
agent advertisement over, 666, 702
dynamic reconfiguration (DR)
adding interfaces to an IPMP group, 729-730
definition, 720-721
detaching interfaces to an IPMP group, 730
DR-attach procedures, 751
DR-detach procedures, 750-751
interfaces not present at boot time, 731
interoperation with IPMP, 729-731
reattaching interfaces in an IPMP group, 730
replacing an interface not present at boot time, 752-754
replacing failed interfaces, 750-751
dynamic routing, 130
best uses, 122
configuring on a single-interface host, 130
host configuration example, 131

## E

EGP, *See* routing protocols
enabling Solaris IP Filter, in previous Solaris 10 releases, 626-628
encapsulated datagram, Mobile IP, 664
encapsulating security payload (ESP)
description, 484-485
IPsec protection mechanism, 484-486
protecting IP packets, 478
security considerations, 485
encapsulation types, Mobile IP, 674
encr_algs security option, ifconfig command, 543
encr_auth_algs security option, ifconfig command, 543
encryption algorithms
IPsec
3DES, 486
AES, 486
Blowfish, 486
DES, 486

encryption algorithms *(Continued)*
specifying for IPsec, 542
error messages for IPQoS, 824
ESP, *See* encapsulating security payload (ESP)
/etc/bootparams file, 249
/etc/default/dhcpagent file, 428-429
/etc/default/dhcpagent file, description, 470
/etc/default/inet_type file, 217-218
DEFAULT_IP value, 273
/etc/default/mpathd file, 754
/etc/defaultdomain file
deleting for network client mode, 104
description, 237
local files mode configuration, 101
/etc/defaultrouter file
description, 237
local files mode configuration, 101
/etc/dhcp/dhcptags file
converting entries, 471
description, 471
/etc/dhcp/eventhook file, 437
/etc/dhcp/inittab file
description, 471
modifying, 405
/etc/dhcp/*interface*.dhc file, description, 470
/etc/dhcp.*interface* file, 422, 428
/etc/dhcp.*interface* file, description, 470
/etc/ethers file, 250
/etc/hostname.*interface* file
description, 236
local files mode configuration, 100
/etc/hostname.*interface* file
manual configuration, 139, 151
/etc/hostname.*interface* file, network client mode configuration, 104
/etc/hostname.*interface* file
router configuration, 117
/etc/hostname6.interface file, IPv6 tunneling, 287
/etc/hostname6.*interface* file, manually configuring interfaces, 172-174
/etc/hostname6.*interface* file, syntax, 267-268
/etc/hostname6.ip.6to4tun0 file, 192
/etc/hostname6.ip.tun file, 190, 191
/etc/hosts file, *See* /etc/inet/hosts file
/etc/inet/dhcpsvc.conf file, 331
/etc/inet/hosts file, 497
adding subnets, 98
format, 237
host name, 238
initial file, 238, 239
local files mode configuration, 101
loopback address, 238
multiple network interfaces, 238, 239
network client mode configuration, 104
/etc/inet/ike/config file
cert_root keyword, 574, 579
cert_trust keyword, 568, 579
description, 548, 598
ignore_crls keyword, 575
ikecert command and, 600
ldap-list keyword, 582
PKCS #11 library entry, 599
pkcs11_path keyword, 577, 599
preshared keys, 555
proxy keyword, 582
public key certificates, 574, 579
putting certificates on hardware, 579
sample, 555
security considerations, 598
self-signed certificates, 568
summary, 550
transmission parameters, 595
use_http keyword, 582
/etc/inet/ike/crls directory, 602
/etc/inet/ike/publickeys directory, 602
/etc/inet/ipaddrsel.conf file, 226, 268
/etc/inet/ipnodes file, 240, 497
/etc/inet/ipsecinit.conf file, 538-539
/etc/inet/ndpd.conf file, 179, 275
6to4 advertisement, 258
6to4 router advertisement, 193
creating, 179
interface configuration variables, 264
keywords, 263-267, 275
prefix configuration variables, 266
temporary address configuration, 182
/etc/inet/netmasks file
adding subnets, 98

/etc/inet/netmasks file *(Continued)*
editing, 243, 244
router configuration, 118
/etc/inet/networks file, overview, 251
/etc/inet/protocols file, 252
/etc/inet/secret/ike.privatekeys directory, 602
/etc/inet/services file, sample, 253
/etc/ipf/ipf.conf file, *See* Solaris IP Filter
/etc/ipf/ipnat.conf file, *See* Solaris IP Filter
/etc/ipf/ippool.conf file, *See* Solaris IP Filter
/etc/ipnodes file removed, 477-478
/etc/netmasks file, 243
/etc/nodename file
deleting for network client mode, 104
description, 236
/etc/nsswitch.conf file, 247, 249
changing, 248, 249
examples, 248
modifications, for IPv6 support, 293-294
name service templates, 248
network client mode configuration, 105
syntax, 248
use by DHCP, 470
/etc/resolv.conf file, use by DHCP, 470
Ethernet addresses
*See* ethers database
*See* MAC address
ethers database
checking entries, 230
corresponding name service files, 247
overview, 250
eventhook file, 437
example IPQoS configuration files
application server, 808
best-effort web server, 797
color-awareness segment, 839
premium web server, 795
VLAN device configuration, 844
expedited forwarding (EF), 770, 841
defining, in the IPQoS configuration file, 803
expire_timer keyword, IKE configuration file, 595
extending DHCP lease, 427

## F

-F option, ikecert certlocal command, 566
-f option
in.iked daemon, 556
ipseckey command, 499
failback
definition, 720
dynamic reconfiguration (DR), with, 730
failover
definition, 720
dynamic reconfiguration (DR), and, 730
examples, 727
standby interface, 724
failover option, ifconfig command, 722
failure detection, in IPMP, 725
definition, 720
NICs missing at boot time, 731
probing rate, 718
failure detection time, IPMP, 726
file services, 44
files
IKE
crls directory, 550, 602
ike/config file, 492, 548, 550, 598
ike.preshared file, 550, 599
ike.privatekeys directory, 550, 602
publickeys directory, 550, 602
IPsec
ipsecinit.conf file, 491, 538-539
ipseckeys file, 492
filter clause, in the IPQoS configuration file, 800, 850
filters, 765
creating, in the IPQoS configuration file, 807, 812
filter clause syntax, 850
planning, in the QoS policy, 782
selectors, list of, 836
flow accounting, 829, 845
flow record table, 845
flow control, through the metering modules, 766
flowacct module, 767, 845
acctadm command, for creating a flow accounting file, 847
action statement for flowacct, 805
attributes of flow records, 846

flowacct module *(Continued)*
flow record table, 845
flow records, 830
parameters, 845
flushing, *See* deleting
foreign agent
authentication, 687
care-of address, 667, 670, 674
considerations, 673
datagrams, 663
definition, 663
determining functionality, 680
encapsulation support, 674
functioning without, 667
implementation, 695
message authentication, 706, 707
registering by using, 670
registering with multiple, 670
registration message, 665
relaying registration request, 672
requesting service from, 673
security association support, 672
serving mobile nodes, 666
visitor list, 692, 711
foreign network, 664, 670, 674
ForeignAgent label, 682, 691, 701, 702
forwarding traffic
datagram forwarding, 843
effect of PHBs on packet forwarding, 841
IP packet forwarding, with DSCP, 769
planning, in the QoS policy, 780
traffic flow through Diffserv networks, 770
fragmented packets, 40
framing
data-link layer, 40, 47
description, 47
ftp program, 42
anonymous FTP program
description, 42

## G

gateway, in a network topology, 121
General section
Mobile IP configuration file, 701
Version label, 701
generating, random numbers, 502-503
gethostbyname command, 294
getipnodebyname command, 294
global zone, IKE, 545
GlobalSecurityParameters section
labels and values, 704
Mobile IP configuration file, 703-704
group failures, IPMP, 727
group parameter
ifconfig command, 737, 749

## H

HA-FAauth label, 682, 687, 704
handshake, three-way, 46
hardware
accelerating IKE computations, 549, 592
physical layer (OSI), 39
physical network layer (TCP/IP), 39, 40
storing IKE keys, 549, 593-594
hardware for IPQoS-enabled networks, 774
header fields, IPv6, 261
header of packets
IP header, 47
TCP protocol functions, 41
home address, 662, 663, 664
home agent
Address section, 706, 707
authentication, 687
binding table, 692, 694, 711
considerations, 673
delivery of datagram, 663
deregistering, 670
determining functionality, 680
dynamic address assignment, 704
dynamic discovery, 674
encapsulation, 674
forwarding datagrams, 675
implementation, 695
message replay protection, 703
mobile node location, 665

home agent *(Continued)*
registration message, 665, 670
registration reply, 673
registration request, 672, 673
security association support, 672
state information, 712
home-foreign agent authentication, 672
home network, 663, 664, 670, 673
`HomeAgent` label, 682, 691, 701, 702
hop, in packet forwarding, 110
hops, relay agent, 356
host, configuring a 6to4 address, 259
host configuration modes (TCP/IP), 95, 97
IPv4 network topology, 97
local files mode, 95, 96
mixed configurations, 97
network client mode, 96
network configuration servers, 96
sample network, 97
host name, enabling client request of, 430-431
host-to-host communications, 40
`hostconfig` program, 104
`hostname.`*interface* file
description, 236
`hostname.`*interface* file, in IPMP, 744
`hostname.`*interface* file
router configuration, 117
`hostname6.`*interface* `file`, manually configuring interfaces, 172-174
`hostname6.`*interface* file, syntax, 267-268
`hostname6.ip.tun` file, 190, 191
hosts
checking host connectivity with `ping`, 216
checking IP connectivity, 217
configuring for IPv6, 181-188
host name
administering, 63
`/etc/inet/hosts` file, 238
in an IPv4 network topology, 97
in an IPv4 routing topology, 115
multihomed
configuring, 124
definition, 115
hosts *(Continued)*
receiving
packet travel through, 47, 48
routing protocol selection, 118
sample network, 97
sending
packet travel through, 45, 47
TCP/IP configuration modes, 97
configuration information, 95
local files mode, 95, 96, 102
mixed configurations, 97
network client mode, 96, 105
network configuration servers, 96
sample network, 97
temporary IPv6 addresses, 181-184
troubleshooting general problems, 229
`hosts.byaddr` map, 198
`hosts.byname` map, 198
`hosts` database, 237, 239
checking entries, 230
corresponding name service files, 247
`/etc/inet/hosts` file
adding subnets, 98
format, 237
host name, 238
initial file, 238, 239
local files mode configuration, 101
loopback address, 238
multiple network interfaces, 238, 239
network client mode configuration, 105
router configuration, 118
name service
affect on, 239
forms of, 246
name services' affect, 239
`hosts` file, 497
`hosts.org_dir` table, 198
http access to CRLs, `use_http` keyword, 582

## I

ICMP protocol
description, 41
displaying statistics, 209

ICMP protocol *(Continued)*
invoking, with `ping`, 216
messages, for Neighbor Discovery protocol, 278
ICMP Router Discovery (RDISC) protocol, 254
identity association, 419
`ifconfig` command, 287, 608
6to4 extensions, 193
`auth_algs` security option, 542-543
checking order of STREAMS modules, 736
configuring
IPv6 tunnels, 272
VLAN devices, 143
controlling DHCP client, 427
`deprecated` attribute, 723
DHCP and, 462
displaying interface status, 205, 207, 725
displaying IPMP group, 747
`encr_algs` security option, 543
`encr_auth_algs` security option, 543
`failover` option, 722
`group` parameter, 737, 749
information in output, 205
IPMP extensions to, 718
IPsec security options, 542-543
IPv6 extensions to, 271
output format, 205
plumbing an interface, 117, 138, 148, 151
`standby` parameter, 724, 744
syntax, 204
`test` parameter, 737
use as troubleshooting tool, 229
`ignore_crls` keyword, IKE configuration file, 575
IGP, *See* routing protocols
IKE
adding self-signed certificates, 565
certificates, 548
checking if valid policy, 556
command descriptions, 550-551
configuration files, 550-551
configuring
for mobile systems, 583-591
with CA certificates, 571-577
with preshared keys, 554
with public key certificates, 564
IKE *(Continued)*
creating self-signed certificates, 565
`crls` database, 602
daemon, 597
databases, 599-602
finding attached hardware, 591
generating certificate requests, 572
global zone, 545
hardware acceleration, 549
hardware storage of keys, 549
`ike.preshared` file, 599
`ike.privatekeys` database, 602
`ikeadm` command, 598-599
`ikecert certdb` command, 573
`ikecert certrldb` command, 583
`ikecert` command, 599
`ikecert tokens` command, 593
implementing, 553
`in.iked` daemon, 597
ISAKMP SAs, 547
key management, 546
mobile systems and, 583-591
NAT and, 587-588, 589-590
overview, 546
perfect forward secrecy (PFS), 546
Phase 1 exchange, 547
Phase 1 key negotiation, 594-596
Phase 2 exchange, 548
PKCS #11 library, 601
preshared keys, 548
privilege level
changing, 562
checking, 559, 563
lowering, 562
`publickeys` database, 602
reference, 597
RFCs, 479
security associations, 597
storage locations for keys, 550-551
troubleshooting transmission timing, 594-596
using Sun Crypto Accelerator 1000 board, 592
using Sun Crypto Accelerator 4000 board, 593-594
`ike/config` file, *See* `/etc/inet/ike/config` file
`ike_mode` keyword, `ikeadm` command, 561

ike.preshared file, 557, 599
  sample, 562
ike.privatekeys database, 602
ikeadm command
  description, 597, 598-599
  interactive mode, 561
  privilege level
    changing, 562
    checking, 559, 563
ikecert certdb command
  -a option, 567, 573
ikecert certlocal command
  -kc option, 572
  -ks option, 565
ikecert certrldb command, -a option, 583
ikecert command
  -A option, 600
  -a option, 577
  description, 597, 599
  -T option, 577, 601
  -t option, 600
ikecert tokens command, 593
in.dhcpd daemon, 307
  debugging mode, 448-449
  description, 462
in.iked daemon
  activating, 597
  -c option, 556
  description, 546
  -f option, 556
  privilege level
    changing, 562
    checking, 559, 563
  stop and start, 499, 559
in.mpathd daemon
  definition, 718
  probing rate, 718
  probing targets, 726
in.ndpd daemon
  checking the status, 230
  creating a log, 219-220
  options, 275
in.rarpd daemon, 96
in.rdisc program, description, 254
in.ripngd daemon, 178, 276
in.routed daemon, 130
  creating a log, 219
  description, 254
  space-saving mode, 254
in.telnet daemon, 43
in.tftpd daemon
  description, 96
  turning on, 102
inactive rule sets, *See* Solaris IP Filter
inbound load balancing, 282
inet_type file, 217-218
inetd daemon
  administering services, 244
inetd daemon, checking the status, 230
inetd daemon
  IPv6 services and, 276-277
  services started by, 132
interactive mode
  ikeadm command, 561
  ipseckey command, 504
interface, definition, 146
interface ID
  definition, 77
  format, in an IPv6 address, 75
  using a manually-configured token, 187
interfaces
  checking packets, 222-223
  configuring
    as part of a VLAN, 160-161
    in Solaris 10 1/06, 150-153
    in Solaris 10 3/05, 138
    into aggregations, 165-168
    IPv6 logical interfaces, 267-268
    manually, for IPv6, 172-174
    plumbing, 148
    temporary addresses, 181-184
  displaying status, 205, 207, 725
  displaying status, Solaris 10 1/06, 149-150
  failover, with IPMP, 727
  IPMP interface types, 724-725
  legacy interface types, 148
  multihomed hosts, 124, 238
  naming conventions, 147

interfaces *(Continued)*
non-VLAN interface types, 148
order of STREAMS modules on an interface, 736
pseudo-interface, for 6to4 tunnels, 192
removing, 140-141
in Solaris 10 1/06, 153-154
router configuration, 115, 118
standby, in IPMP, 724, 743-745
types, in Solaris 10 1/06, 148
types of NICs, 137, 147
types that support aggregations, 165
verifying MAC address uniqueness, 154-155
VLANs, 156-161
VLANs, in Solaris 10 3/05, 141-143
Internet, domain name registration, 38
Internet Assigned Numbers Authority (IANA),
registration services, 60
Internet drafts
definition, 49
SCTP with IPsec, 479
Internet layer (TCP/IP)
ARP protocol, 41
description, 39, 40
ICMP protocol, 41
IP protocol, 40
packet life cycle
receiving host, 48
sending host, 47
Internet Protocol (IP), 662
Internet Security Association and Key Management
Protocol (ISAKMP) SAs
description, 547
storage location, 599
internetworks
definition, 66
packet transfer by routers, 67, 68
redundancy and reliability, 66
topology, 66
InterNIC
registration services
domain name registration, 38
interoperability
IPsec with other platforms in tunnel mode, 477
interoperability *(Continued)*
IPsec with other platforms using preshared
keys, 558
IP address
`BaseAddress` label, 705
care-of address, 667
IP source address, 674
mobile node, 664, 672
source IP address, 675
IP addresses
allocation with DHCP, 322
designing an address scheme, 55, 62
DHCP
adding, 374
errors, 444
modifying properties, 377
properties, 371
removing, 380
reserving for client, 383
tasks, 370
unusable, 380
displaying addresses of all interfaces, 207
IP protocol functions, 40
network classes
network number administration, 56
network interfaces and, 62
subnet issues, 243
IP datagrams
IP header, 47
IP protocol formatting, 40
packet process, 47
protecting with IPsec, 478
UDP protocol functions, 42
IP Filter, *See* Solaris IP Filter
IP forwarding
in IPv4 VPNs, 514, 517, 525
in IPv6 VPNs, 521, 523, 531, 534
in VPNs, 489
IP link, in IPMP terminology, 719
IP network multipathing (IPMP), *See* IPMP
IP protocol
checking host connectivity, 216, 217
description, 40
displaying statistics, 209

IP security architecture, *See* IPsec
`ipaddrsel` command, 226, 268-270
`ipaddrsel.conf` file, 226, 268
`ipf` command
*See also* Solaris IP Filter
`-6` option, 615-616
`-a` option, 634-636
append rules from command line, 636-637
`-D` option, 625
`-E` option, 621-622
`-F` option, 623-624, 634-636, 636, 639
`-f` option, 621-622, 634-636, 636-637, 637-638
`-I` option, 637-638, 639
`-s` option, 638-639
`ipf.conf` file, 609-611
*See* Solaris IP Filter
`ipfstat` command, 644-645
*See also* Solaris IP Filter
`-6` option, 615-616
`-I` option, 634
`-i` option, 633-634, 634
`-o` option, 633-634, 634
`-s` option, 645-646
`-t` option, 644-645
`ipgpc` classifier, *See* classifier module
`ipmon` command
*See also* Solaris IP Filter
`-a` option, 648-649
`-F` option, 649
IPv6 and, 615-616
`-o` option, 648-649
IPMP
administering, 746-749
ATM support, 736
basic requirements, 721
data addresses, 721
dynamic reconfiguration, 720-721, 729-731
Ethernet support, 736
failover
definition, 720
failure detection
definition, 720
failure detection time, 726
IPMP *(Continued)*
group configuration
planning for an IPMP group, 735-737
tasks for configuring, 737-741
troubleshooting, 740
`hostname.`*interface* file, 744
interface configuration
active-active, 725
active-standby, 725
standby interface, 724, 743-745
types of interface configurations, 724
IP links, types of, 719
IPMP configuration file, 754-755
link-based failure detection, 725-726
load spreading, 718
multipathing group definition
*See* IPMP group
network drivers supported, 725
overview, 717-721
preserving configuration across reboots, 739, 740, 744
probe-based failure detection, 726-727
probe traffic, 722
repair detection, 720
replacing an interface not present at system boot, 752-754
replacing interfaces, DR, 750-751
software components, 718
target systems, 720
configuring in a script, 742-743
configuring manually, 742
terminology, 718-721
test addresses, 722-723
Token ring support, 736
IPMP daemon `in.mpathd`, 718
IPMP groups
adding an interface to a group, 747-748
adding interfaces, through DR, 729-730
affect of interfaces not present at boot time, 731
configuring, 737-741
configuring a group for a single interface, 745-746
displaying group membership, 747
group failures, 727
moving an interface between groups, 749

IPMP groups *(Continued)*
NIC speed in a group, 719
planning tasks, 735-737
removing an interface from a group, 748-749
removing interfaces, through DR, 730
troubleshooting group configuration, 740
`ipnat` command
*See also* Solaris IP Filter
append rules from command line, 641
`-C` option, 624
`-F` option, 624, 640-641
`-f` option, 621-622, 641
`-l` option, 640
`-s` option, 646
`ipnat.conf` file, 612-613
*See* Solaris IP Filter
`ipnodes.byaddr` map, 198
`ipnodes.byname` map, 198
`ipnodes` file, 240, 497
`ipnodes.org_dir` table, 198
`ippool` command
*See also* Solaris IP Filter
append rules from command line, 643
`-F` option, 642
`-f` option, 643
IPv6 and, 615-616
`-l` option, 642
`-s` option, 646-647
`ippool.conf` file, 613-614
*See* Solaris IP Filter
IPQoS, 759
configuration example, 789-791
configuration file, 795, 847
`action` statement syntax, 849
`class` clause, 799
`filter` clause, 800
initial `action` statement, 849
initial action statement, 798
list of IPQoS modules, 849
marker `action` statement, 802
syntax, 848
configuration planning, 773
Diffserv model implementation, 764
error messages, 824
IPQoS *(Continued)*
features, 760
man pages, 761
message logging, 823
network example, 795
network topologies supported, 774, 775, 776
policies for IPv6-enabled networks, 88
QoS policy planning, 777
related RFCs, 761
routers on an IPQoS network, 818
statistics generation, 832
traffic management capabilities, 763, 764
VLAN device support, 843
`ipqosconf`, 794
`ipqosconf` command
applying a configuration, 822, 823
command options, 851
listing the current configuration, 823
IPsec
activating, 491
adding security associations (SAs), 498
algorithm source, 540
authentication algorithms, 486
bypassing, 487, 500
commands, list of, 491-492
components, 478
configuration files, 491-492
configuring, 486, 537-538
creating SAs manually, 503-507
displaying policies, 501
encapsulating data, 484
encapsulating security payload (ESP), 484-486
encryption algorithms, 486
`/etc/hosts` file, 497
`/etc/inet/ipnodes` file, 497
extensions to utilities
`ifconfig` command, 542-543
`snoop` command, 542, 544
getting random numbers for keys, 502-503
`ifconfig` command
configuring VPN, 517, 523, 533
security options, 542-543
implementing, 495
`in.iked` daemon, 483

IPsec *(Continued)*
inbound packet process, 480
interoperating with other platforms
IP-in-IP tunnels, 477
preshared keys, 502, 558
`ipsecalgs` command, 486, 540
`ipsecconf` command, 487, 537-538
`ipsecinit.conf` file
bypassing LAN, 516, 526, 542
configuring, 498
description, 538-539
policy file, 487
protecting web server, 500
removing IPsec bypass of LAN, 519, 529
`ipseckey` command, 483, 541-542
IPv4 VPN in tunnel transport mode, and, 524-530
IPv4 VPNs, and, 514-520
IPv6 VPN in tunnel transport mode, and, 530-536
IPv6 VPNs, and, 520-524
key management, 483
keying utilities
IKE, 546
`ipseckey` command, 541-542
Mobile IP, 677
NAT and, 490
outbound packet process, 480
overview, 478
policy command, 537-538
policy files, 538-539
protecting
mobile systems, 583-591
packets, 478
VPNs, 514-520
web servers, 499-501
protecting a VPN, 509-511, 511-536
protection mechanisms, 484-486
protection policy, 486-487
RBAC and, 496
replacing security associations (SAs), 504
RFCs, 479
`route` command, 518, 523, 528, 534
SCTP protocol and, 491, 496
securing traffic, 496-499
security associations (SAs), 482-483
IPsec *(Continued)*
security associations database (SADB), 478, 540
security mechanisms, 479
security parameter index (SPI), 482-483
security policy database (SPD), 479, 480, 537
security protocols, 478, 482-483
security roles, 508-509
services
`ipsecalgs`, 492
`manual-key`, 492
`policy`, 491
setting policy
permanently, 538-539
temporarily, 537-538
`snoop` command, 542, 544
Solaris cryptographic framework and, 540
specifying
authentication algorithms, 542
encryption algorithms, 542
terminology, 480
transport mode, 487-489
tunnel mode, 487-489
tunnels, 489
verifying packet protection, 507-508
virtual private networks (VPNs), 489, 514-520
zones and, 491, 496
IPsec policy
example of tunnels in transport mode, 529
example of using deprecated syntax, 529-530
examples of tunnel syntax, 509-511
IP-in-IP datagrams, 477-478
LAN example, 519
specifying, 522, 533
IPsec tunnels, simplified syntax, 477-478
`ipsecconf` command
`-a` option, 499, 562
configuring IPsec policy, 537-538
description, 491
displaying IPsec policy, 499-501, 501
`-f` option, 499
purpose, 487
security considerations, 499, 539
setting tunnels, 488
viewing IPsec policy, 538-539

`ipsecinit.conf` file
bypassing LAN, 516, 526
configuring tunnel options, 542
description, 491
location and scope, 491
protecting web server, 500
purpose, 487
removing IPsec bypass of LAN, 519, 529
sample, 538
security considerations, 539
`ipseckey` command
description, 492, 541-542
interactive mode, 504
purpose, 483
security considerations, 541-542
`ipseckeys` file, storing IPsec keys, 492
IPv4 addresses
applying netmasks, 242, 243
dotted-decimal format, 58
format, 58
IANA network number assignment, 60
network classes, 60
addressing scheme, 59, 60
class A, 254
class B, 255
class C, 255
parts, 60
range of numbers available, 60
subnet issues, 241
subnet number, 60
symbolic names for network numbers, 244
IPv6
6to4 address, 258
adding
addresses to NIS, 198
DNS support, 197
address autoconfiguration, 275, 278
addressing plan, 91-92
and Solaris IP Filter, 615-616
ATM support, 295
automatic tunnels, 286
checking the status of `in.ndpd`, 230
comparison with IPv4, 70, 282-283
configuring tunnels, 189-190
IPv6 *(Continued)*
default address selection policy table, 269
DNS AAAA records, 198
DNS support preparation, 88-89
dual-stack protocols, 86
duplicate address detection, 80
enabling, on a server, 187-188
extension header fields, 262
extensions to `ifconfig` command, 271
`in.ndpd` daemon, 275
`in.ripngd` daemon, 276
known issues with 6to4 router, 232
link-local addresses, 280, 283
monitoring traffic, 225
multicast addresses, 260-261, 283
Neighbor Discovery protocol, 278-283
neighbor solicitation, 278
neighbor solicitation and unreachability, 280
neighbor unreachability detection, 80, 283
next-hop determination, 80
`nslookup` command, 199
packet header format, 261-262
protocol overview, 278
redirect, 80, 278, 283
router advertisement, 278, 279, 282, 284
router discovery, 275, 282
router solicitation, 278, 279
routing, 284
security considerations, 90
site-local addresses, 81
stateless address autoconfiguration, 280
subnets, 73
temporary address configuration, 181-184
troubleshooting common IPv6 problems, 231-233
tunnels, 287-289
IPv6 addresses
address autoconfiguration, 80, 81
address resolution, 80
anycast, 79
interface ID, 77
link-local, 78
multicast, 79
unicast, 77
uniqueness, 280

IPv6 addresses *(Continued)*
VPN example of use with IPsec, 520-524
IPv6 features, Neighbor Discovery functionality, 79
IPv6 link-local address, with IPMP, 723

## K

`-kc` option
`ikecert certlocal` command, 566, 572, 600
`-ks` option
`ikecert certlocal` command, 565, 600
`Key` label, 683, 688, 706
key management
automatic, 546
IKE, 546
IPsec, 483
manual, 541-542
zones and, 496
key negotiation, IKE, 594-596
key storage
IPsec SAs, 492
ISAKMP SAs, 599
softtoken, 599
softtoken keystore, 478, 594
token IDs from metaslot, 594
`KeyDistribution` label, 682, 704
keying utilities
IKE protocol, 546
`ipseckey` command, 483
keys
automatic management, 546
creating for IPsec SAs, 503-507
generating random numbers for, 502-503
`ike.privatekeys` database, 602
`ike/publickeys` database, 602
managing IPsec, 483
manual management, 541-542
preshared (IKE), 548
storing (IKE)
certificates, 601
private, 600
public keys, 601
storing on hardware, 549
keystore name, *See* token ID
`kstat` command, use with IPQoS, 832

## L

`-L` option, `ipsecconf` command, 501
`-l` option
`ikecert certdb` command, 568
`ipsecconf` command, 501
`ldap-list` keyword, IKE configuration file, 582
legacy interfaces, 148
libraries, PKCS #11, 601
link, IPv6, 73
link aggregation control protocol (LACP)
modes, 165
modifying LACP modes, 168
link aggregations, *See* aggregations
link-based failure detection, definition, 725-726
link-layer address change, 282
link-local address
as an IPMP test address, 723
format, 78
manually configuring, with a token, 187
link-local addresses
IPv6, 280, 283, 287
listing
algorithms (IPsec), 485, 543
certificates (IPsec), 568, 581
CRL (IPsec), 581
hardware (IPsec), 593
token IDs (IPsec), 593
token IDs from metaslot, 594
load balancing
across aggregations, 164
in an IPQoS-enabled network, 775
on an IPv6-enabled network, 282
load spreading
definition, 718
outbound, 720
local files mode
definition, 95
host configuration, 102
network configuration servers, 96
systems requiring, 95, 96

local files name service
description, 64
/etc/inet/hosts file, 497
example, 239
format, 237
initial file, 238, 239
requirements, 239
/etc/inet/ipnodes file, 497
local files mode, 95, 96
network databases, 246
log file, flushing in Solaris IP Filter, 649
log files
creating for Solaris IP Filter, 647-648
viewing for Solaris IP Filter, 648-649
logged packets, saving to a file, 650
logical interface, 419, 420
logical interfaces
definition, 137, 146
DHCP client systems, 429
for IPv6 address, 267-268
for IPv6 tunnels, 190, 191
loopback address, 104, 238
lost or dropped packets, 41, 216

## M

-m option, ikecert certlocal command, 566
MAC address, 419
IPMP requirements, 721
IPv6 interface ID, 77
mapping to IP in ethers database, 250
used in DHCP client ID, 311
verifying uniqueness, 154-155
machines, protecting communication, 496-499
macros
DHCP
*See* DHCP macros
marker modules, 766
*See also* dlcosmk marker
*See also* dscpmk marker
PHBs, for IP packet forwarding, 769
specifying a DS codepoint, 842
support for VLAN devices, 843
MaxClockSkew label, 682, 704
maximum transmission unit (MTU), 282
MD5 authentication algorithm, key length, 505
media access control (MAC) address, *See* MAC address
message authentication
Mobile IP, 672, 705, 707
message replay protection, 703
messages, router advertisement, 285
metaslot
key storage, 478, 545, 594
metering modules
*See also* tokenmt meter
*See also* tswtclmt meter
introduction, 766
invoking, in the IPQoS configuration file, 816
outcomes of metering, 766, 838
mipagent.conf configuration file, 680, 682, 696, 710
configuring, 680
mipagent daemon, 681, 696, 712
mipagent_state file, 712
mipagentconfig command
configuring mobility agent, 710
description of commands, 710
modifying
Address section, 689
Advertisements section, 686
configuration file, 685
General section, 686
GlobalSecurityParameters section, 687
Pool section, 687
SPI section, 688
mipagentstat command
displaying agent status, 692-693
mobility agent status, 711
MN-FAauth label, 682, 704
mobile-foreign agent authentication, 672
mobile-home agent authentication, 672
Mobile IP
Address section
default mobile node, 684, 709
Network Access Identifier, 684, 708
agent advertisement, 665, 666, 670
agent discovery, 666-667
agent solicitation, 665, 666, 667
broadcast datagrams, 675

Mobile IP *(Continued)*
configuration file
`Address` section, 704, 706-709
`Advertisements` section, 701-703
`General` section, 701
`GlobalSecurityParameters` section, 703-704
`Pool` section, 704-705
`SPI` section, 705-706, 706, 707
configuration file format, 696-697
configuration file sections, 701
configuring, 680-684
datagram movement, 663
deploying, 679
deregistering, 666, 670, 671
displaying agent status, 692-693
encapsulated datagram, 664
encapsulation types, 674
functions not supported, 696
how it works, 664-666
IPsec, use of, 677
message authentication, 672, 676, 705
multicast datagram routing, 675-676
Network Access Identifier, 706
private addresses, 668-669
registering, 664, 665, 670
reverse tunnel flag, 672
registration messages, 670, 671, 672, 696
registration reply message, 673
registration request, 672
registration request message, 673
reverse tunnel, 666, 668-669
foreign agent considerations, 673
home agent considerations, 673
multicast datagram routing, 676
unicast datagram routing, 675
RFCs not supported, 696
RFCs supported, 695
router advertisement, 696
sample configuration files, 697-701
security association, 672
security considerations, 676-677
security parameter index (SPI), 672, 705
state information, 712
unicast datagram routing, 674-675
Mobile IP *(Continued)*
wireless communications, 663, 667, 676
Mobile IP topology, 662
mobile node, 662, 663, 664, 708
`Address` section, 683
definition, 663
mobility agent, 665, 673
`Address` section, 706, 707
configuring, 710
`mipagent_state` file, 712
router advertisements, 696
software, 695
status of, 711
mobility binding, 670, 672, 673, 675
modifying
DHCP macros, 388
DHCP options, 402
`mpathd` file, 754-755
multicast addresses, IPv6
compared to broadcast addresses, 283
format, 260-261
overview, 79
multicast datagram routing, Mobile IP, 675-676
multihomed hosts
configuration example, 126
configuring, 125-127
configuring during installation, 238-239
definition, 115, 124
enabling for IPv6, 172-174
on firewalled networks, 124
multiple network interfaces
DHCP client systems, 429
`/etc/inet/hosts` file, 238, 239
router configuration, 115, 118

## N

name services
administrative subdivisions, 65
database search order specification, 247, 249
domain name registration, 38
domain name system (DNS), 43, 64
files corresponding to network databases, 246
`hosts` database and, 239

name services *(Continued)*
local files
description, 64
/etc/inet/hosts file, 237, 239
local files mode, 95, 96
network databases and, 64, 245
NIS, 64
NIS+, 64
nsswitch.conf file templates, 248
registration of DHCP clients, 355
selecting a service, 63, 65
supported services, 63
names/naming
domain names
registration, 38
selecting, 64
top-level domains, 64
host name
administering, 63
/etc/inet/hosts file, 238
naming network entities, 63, 65
node name
local host, 104, 236
NAT
compliant with RFCs, 478
configuring rules for, 612-613
deactivating, 624
IPsec supports multiple clients, 477-478
limitations with IPsec, 490
NAT rules
appending, 641
viewing, 640
overview, 612-613
removing NAT rules, 640-641
using IPsec and IKE, 587-588, 589-590
viewing statistics, 646
ndd command, viewing pfil module and, 631-632
ndpd.conf file
6to4 advertisement, 193
creating, on an IPv6 router, 179
ndpd.conf file
interface configuration variables, 264
keyword list, 263-267
prefix configuration variables, 266
ndpd.conf file
temporary address configuration, 182
Neighbor Discovery protocol
address autoconfiguration, 80, 278
address resolution, 80
capabilities, 79
comparison to ARP, 282-283
duplicate address detection algorithm, 281
major features, 278-283
neighbor solicitation, 280
prefix discovery, 80, 280
router discovery, 80, 279
neighbor solicitation, IPv6, 278
neighbor unreachability detection
IPv6, 80, 280, 283
/net/if_types.h file, 736
netmasks database, 241
adding subnets, 98, 101
corresponding name service files, 247
/etc/inet/netmasks file
adding subnets, 98
editing, 243, 244
router configuration, 118
network masks
applying to IPv4 address, 242, 243
creating, 242, 243
description, 242
subnetting, 241
netstat command
-a option, 212
description, 208
displaying status of known routes, 215-216
-f option, 212
inet option, 212
inet6 option, 212
IPv6 extensions, 273
Mobile IP extensions, 712
per-protocol statistics display, 209
-r option, 215-216
running software checks, 230
syntax, 208
Network Access Identifier
Mobile IP, 706
Mobile IP Address section, 684, 708

Network Address Translation (NAT), *See* NAT
network administration
  designing the network, 55
  host names, 63
  network numbers, 56
  Simple Network Management Protocol (SNMP), 44
network classes, 60
  addressing scheme, 59, 60
  class A, 254
  class B, 255
  class C, 255
  IANA network number assignment, 60
  network number administration, 56
  range of numbers available, 60
network client mode
  definition, 95
  host configuration, 105
  overview, 96
network clients
  `ethers` database, 250
  host configuration, 105
  network configuration server for, 96, 102
  systems operating as, 96
network configuration
  configuring
    network clients, 104
    services, 132
  configuring security, 475
  enabling IPv6 on a host, 181-188
  hop, description, 110
  host configuration modes, 95
  IPv4 network configuration tasks, 99
  IPv4 network topology, 97
  IPv6-enabled multihomed hosts, 172-174
  IPv6 router, 177
  network configuration server setup, 102
  router, 116
  TCP/IP configuration modes, 97
    configuration information, 95
    local files mode, 96
    network client mode, 96
    network configuration servers, 96
network configuration servers
  booting protocols, 96
network configuration servers *(Continued)*
  definition, 96
  setting up, 102
network databases, 245, 247
  `bootparams` database, 249
  corresponding name service files, 246
  DNS boot and data files and, 246
  `ethers` database
    checking entries, 230
    overview, 250
  `hosts` database
    checking entries, 230
    name services, affect on, 239
    name services, forms of, 246
    name services affect on, 239
    overview, 237, 239
  name services' affect, 245, 247
  `netmasks` database, 241, 247
  `networks` database, 251
  `nsswitch.conf` file and, 245, 247, 249
  `protocols` database, 252
  `services` database, 253
network example for IPQoS, 795
network interface, configuring, 137-143
network interface card (NIC)
  administering NICs not present at boot time, 731
  attaching NICs with DR, 729-730
  definition, 719
  detaching NICs with DR, 730
  dynamic reconfiguration, 720-721
  failure and failover, 720
  NIC speed in an IPMP group, 719
  NICs, types of, 137, 147
  NICs that support IPMP, 725
  repair detection, 720
network interface names, 147
network interfaces
  displaying DHCP status, 427
  IP addresses and, 62
  monitoring by DHCP service, 358-359
  multiple network interfaces
    `/etc/inet/hosts` file, 238, 239
network layer (OSI), 39
Network Management rights profile, 509

network numbers, 38
network planning, 53, 68
  adding routers, 65, 68
  design decisions, 55
  IP addressing scheme, 55, 62
  name assignments, 63, 65
  registering your network, 57
network prefix, IPv4, 61
network security, configuring, 475
Network Security rights profile, 508-509
network topologies for IPQoS, 774
  configuration example, 790
  LAN with IPQoS-enabled firewall, 776
  LAN with IPQoS-enabled hosts, 775
  LAN with IPQoS-enabled server farms, 774
network topology, 66
  autonomous system, 113
  DHCP and, 316
`networks` database
  corresponding name service files, 247
  overview, 251
new features
  configuring target systems in IPMP, 741-743
  default address selection, 225-228
  DHCP event scripts, 436-439
  DHCP on logical interfaces, 429
  IKE enhancements, 551
  `inetconv` command, 103
  interface status with `dladm` command, 149
  IPsec enhancements, 492-493
  link-based failure detection, 725-726
  manually configuring a link-local address, 185-187
  `routeadm` command, 178
  SCTP protocol, 133-136
  Service Management Facility (SMF), 103
  site prefix, in IPv6, 74, 75-76
  temporary addresses in IPv6, 181-184
next-hop, 110, 283
next-hop determination, IPv6, 80
NFS services, 44
NIC
  *See* network interface card (NIC)
  specifying for Solaris IP Filter, 628-630
NIS
  adding IPv6 address, 198
  domain name registration, 38
  network databases, 64, 246
  selecting as name service, 64
NIS+
  and DHCP data store, 441-444
  selecting as name service, 64
`nisaddcred` command, and DHCP, 444
`nischmod` command, and DHCP, 443
`nisls` command, and DHCP, 443
`nisstat` command, and DHCP, 442
node, IPv6, 73
node name
  local host, 104, 236
`nodename` file
  deleting for network client mode, 104
  description, 236
non-VLAN interfaces, 148
`nslookup` command, 294
  IPv6, 199
`nsswitch.conf` file, 247, 249
  changing, 248, 249
  examples, 248
  modifications, for IPv6 support, 293-294
  name service templates, 248
  network client mode configuration, 105
  syntax, 248

## O

`od` command, 556
Open Systems Interconnect (OSI) Reference Model, 38, 39
`/opt/SUNWconn/lib/libpkcs11.so` entry, in `ike/config` file, 599
option requests, 420

## P

`-p` option
  `in.iked` daemon, 561, 564
packet filter hooks, 614

packet filtering
activating a different rule set, 634-636
appending
rules to active set, 636-637
rules to inactive set, 637-638
configuring, 609-611
deactivating, 623-624
managing rule sets, 633-639
reloading after updating current rule set, 634-636
removing
active rule set, 636
inactive rule set, 639
specifying a NIC, 628-630
switching between rule sets, 638-639
packet flow
relay router, 292
through tunnel, 291
packet flow, IPv6
6to4 and native IPv6, 292
through 6to4 tunnel, 291
packet forwarding router, 115
packets
checking flow, 222
data encapsulation, 46, 47
description, 45
displaying contents, 222
dropped or lost, 41, 216
forwarding, 109
fragmentation, 40
header
IP header, 47
TCP protocol functions, 41
IP protocol functions, 40
IPv6 header format, 261-262
life cycle, 45, 48
application layer, 45
data-link layer, 47
Internet layer, 47
physical network layer, 47
receiving host process, 47, 48
transport layer, 46, 47
protecting
inbound packets, 480
outbound packets, 480
packets, protecting *(Continued)*
with IKE, 547
with IPsec, 480, 484-486
transfer
router, 67, 68
TCP/IP stack, 45, 48
UDP, 46
verifying protection, 507-508
`params` clause
defining global statistics, 798, 851
for a `flowacct` `action`, 805
for a marker `action`, 802
for a metering `action`, 816
syntax, 851
per-hop behavior (PHB), 769
AF forwarding, 770
defining, in the IPQoS configuration file, 817
EF forwarding, 770
using, with `dscpmk` marker, 841
perfect forward secrecy (PFS)
description, 547
IKE, 546
`PF_KEY` socket interface
IPsec, 483, 492
`pfil` module, 615
viewing statistics, 631-632
PFS, *See* perfect forward secrecy (PFS)
physical interface, 162-163
*See also* interfaces
adding, after installation, 138, 150
configuring, 137-143
definition, 137, 146, 719
failure detection, 725
naming conventions, 147
network interface card (NIC), 137, 147
removing, 153-154
in Solaris 10 3/05, 140-141
repair detection with IPMP, 727
VLANs, definition, 141-143
physical layer (OSI), 39
physical network layer (TCP/IP), 40, 47
physical point of attachment (PPA), 142, 158
`ping` command, 217
description, 216

ping command *(Continued)*
extensions for IPv6, 274
running, 217
-s option, 216
syntax, 216
PKCS #11 library
in ike/config file, 599
specifying path to, 601
pkcs11_path keyword
description, 599
ikecert command and, 601
using, 577
plumbing an interface, 117, 138, 148, 151
pntadm command
description, 308, 461
examples, 370
using in scripts, 462
policies, IPsec, 486-487
policies, for aggregations, 164
policy files
ike/config file, 492, 550, 598
ipsecinit.conf file, 538-539
security considerations, 539
Pool label, 684, 689, 708, 709
Pool section
labels and values, 705
Mobile IP configuration file, 704-705
ports, TCP, UDP, and SCTP port numbers, 253
PPP links
troubleshooting
packet flow, 222
prefix
network, IPv4, 61
site prefix, IPv6, 75-76
subnet prefix, IPv6, 76
prefix discovery, in IPv6, 80
prefixes
router advertisement, 280, 282, 284
PrefixFlags label, 682, 702
presentation layer (OSI), 38
preshared keys (IKE)
description, 548
replacing, 558-559
shared with other platforms, 558
preshared keys (IKE) *(Continued)*
storing, 599
task map, 554
preshared keys (IPsec), creating, 503-507
primary network interface, 137, 147
private addresses, Mobile IP, 668-669
private keys, storing (IKE), 600
privilege level
checking in IKE, 559, 563
setting in IKE, 561, 564
probe-based failure detection
configuring target systems, 741-743
definition, 726-727
failure detection time, 726
probe traffic, IPMP, 722
probing targets, 726
probing targets, in.mpathd daemon, 722
protecting
IPsec traffic, 478
keys in hardware, 549
mobile systems with IPsec, 583-591
packets between two systems, 496-499
VPN with IPsec tunnel in transport mode, 524-530
VPN with IPsec tunnel in tunnel mode, 514-520
web server with IPsec, 499-501
Protecting a VPN With IPsec (Task Map), 511-536
Protecting Traffic With IPsec (Task Map), 495
protection mechanisms, IPsec, 484-486
protocol layers
OSI Reference Model, 38, 39
packet life cycle, 45, 48
TCP/IP protocol architecture model, 39, 44
application layer, 39, 42, 44
data-link layer, 39, 40
Internet layer, 39, 40
physical network layer, 39, 40
transport layer, 39, 41
protocol statistics display, 209
protocols database
corresponding name service files, 247
overview, 252
proxy keyword, IKE configuration file, 582
public key certificates, *See* certificates
public keys, storing (IKE), 601

public topology, IPv6, 77
publickeys database, 602

## Q

-q option, in.routed daemon, 254
QoS policy, 762
  creating filters, 782
  implementing, in the IPQoS configuration file, 793
  planning task map, 778
  template for policy organization, 777
quality of service (QoS)
  QoS policy, 762
  tasks, 759

## R

random numbers, generating with od command, 556
RARP protocol
  checking Ethernet addresses, 230
  description, 96
  Ethernet address mapping, 250
  RARP server configuration, 102
RBAC
  and DHCP commands, 308
  IPsec and, 496
RDISC
  description, 44, 254
receiving hosts
  packet travel through, 47, 48
Reconfiguration Coordination Manager (RCM)
  framework, 730
redirect
  IPv6, 80, 278, 283
refreshing, preshared keys (IKE), 558-559
registering
  autonomous systems, 115
  domain names, 38
  networks, 57
registration
  messages, 670, 673
  Mobile IP, 664, 665, 670
  reply message, 673
registration *(Continued)*
  request, 672
  reverse tunnel flag, 672
RegLifetime label, 682, 702
relay router, 6to4 tunnel configuration, 195, 196
repair detection, with IPMP, 720, 727
replacing
  IPsec SAs, 504
  manual keys (IPsec), 504
  preshared keys (IKE), 558-559
ReplayMethod label, 683, 706
Requests for Comments (RFCs), 49
  definition, 49
  IKE, 479
  IPQoS, 761
  IPsec, 479
  IPv6, 71
requirements for IPMP, 721
retry_limit keyword, IKE configuration file, 595
retry_timer_init keyword, IKE configuration file, 595
retry_timer_max keyword, IKE configuration file, 595
reverse tunnel
  foreign agent considerations, 673
  home agent considerations, 673
  Mobile IP, 666, 668-669
  multicast datagram routing, 676
  unicast datagram routing, 675
reverse zone file, 197
ReverseTunnel label, 682, 703
ReverseTunnelRequired label, 682, 703
rights profiles, Network Management, 509
rlogin command, packet process, 46
roles, creating network security role, 508-509
route command
  inet6 option, 274
  IPsec, 518, 523, 528, 534
routeadm command
  configuring VPN with IPsec, 529
  enabling dynamic routing, 131
  IP forwarding, 515, 525
  IPv6 router configuration, 178
  multihomed hosts, 125
  turning on dynamic routing, 119

router advertisement, 422
IPv6, 278, 279, 282, 284-285
Mobile IP, 696
prefix, 280
router discovery, in IPv6, 80, 275, 279, 282
router solicitation
IPv6, 278, 279
routers
adding, 65, 68
addresses for DHCP clients, 322
border, 114
configuring, 253
for IPv4 networks, 115
IPv6, 177
network interfaces, 118
default address, 99
default routers, 115
definition, 110, 116, 253
dynamic routing, 130
`/etc/defaultrouter` file, 237
example, configuring a default router, 119
local files mode configuration, 101
network topology, 66
packet forwarding router, 115
packet transfer, 67, 68
problems upgrading for IPv6, 231
role, in 6to4 topology, 290
routing protocols
automatic selection, 118
description, 44, 253, 254
static routing, 128
routing
configuring static, 128
definition, 110
direct route, 110
dynamic routing, 121
gateway, 121
indirect route, 110
IPv6, 284
manually configuring a routing table, 121
on multihomed hosts, 124
on single-interface hosts, 127
routing table configuration, 122
static routing, 121
routing information protocol (RIP)
description, 44, 254
routing protocols
associated routing daemons, 111
automatic selection, 118
Border Gateway Protocol (BGP), 114
description, 44, 110, 253, 254
exterior gateway protocol (EGP), 110
in the Solaris OS, 110
interior gateway protocol (IGP), 110
RDISC
description, 44, 254
RIP
description, 44, 254
routing tables
definition, 110
description, 67
displaying, 229
`in.routed` daemon creation of, 254
manually configuring, 121, 122
packet transfer example, 68
space-saving mode, 254
subnetting and, 241
tracing all routes, 221-222
`rpc.bootparamd` daemon, 96
RSA encryption algorithm, 601
rule sets
*See* See Solaris IP Filter
inactive
*See also* Solaris IP Filter
NAT, 612-613
packet filtering, 608-614

## S

`-S` option
`ikecert certlocal` command, 566
`in.routed` daemon, 254
`-s` option, `ping` command, 217
SCTP protocol
adding SCTP-enabled services, 133-136
description, 42
displaying statistics, 209
displaying status, 211

SCTP protocol *(Continued)*
IPsec and, 496
limitations with IPsec, 491
service in `/etc/inet/services` file, 253
security
IKE, 597
IPsec, 478
security associations, Mobile IP, 672
security associations (SAs)
adding IPsec, 498
creating manually, 503-507
flushing IPsec SAs, 504
getting keys for, 502-503
IKE, 597
IPsec, 482-483, 498
IPsec database, 540
ISAKMP, 547
random number generation, 548
replacing IPsec SAs, 504
security associations database (SADB), 540
security considerations
6to4 relay router issues, 232
authentication header (AH), 485
configuring
IKE to find hardware, 577
IKE transmission parameters, 595
IKE with certificates, 565
IKE with preshared keys, 555
IPsec, 497
encapsulating security payload (ESP), 485
`ike/config` file, 598
`ipsecconf` command, 539
`ipsecinit.conf` file, 539
`ipseckey` command, 541-542
`ipseckeys` file, 506
IPv6-enabled networks, 90
latched sockets, 539
Mobile IP, 676-677
preshared keys, 548
security protocols, 485
security parameter index (SPI)
constructing, 502
description, 482-483
key size, 502
security parameter index (SPI) *(Continued)*
Mobile IP, 672, 705
security policy
`ike/config` file (IKE), 492
IPsec, 486-487
`ipsecinit.conf` file (IPsec), 497, 538-539
security policy database (SPD)
configuring, 537
IPsec, 479, 480
security protocols
authentication header (AH), 484
encapsulating security payload (ESP), 484-485
IPsec protection mechanisms, 484
overview, 478
security considerations, 485
selectors, 765
IPQoS 5-tuple, 765
planning, in the QoS policy, 782
selectors, list of, 836
sending hosts
packet travel through, 45, 47
server, DHCPv6, 418
servers, IPv6
enabling IPv6, 187-188
planning tasks, 87
service-level agreement (SLA), 762
billing clients, based on flow accounting, 830
classes of services, 765
providing different classes of service, 764
services
network and `svcadm` command, 515, 521, 526, 532
`services` database
corresponding name service files, 247
overview, 253
updating, for SCTP, 133
session layer (OSI), 38
Simple Network Management Protocol (SNMP), 44
site-local addresses, IPv6, 81
site prefix, IPv6
advertising, on the router, 179
definition, 74, 76
how to obtain, 90-91
site topology, IPv6, 77
`Size` label, 683, 705

slots, in hardware, 602
SNMP (Simple Network Management Protocol), 44
snoop command
  checking packet flow, 222
  checking packets between server and client, 224
  displaying packet contents, 222
  extensions for IPv6, 274
  ip6 protocol keyword, 274
  Mobile IP extensions, 712-713
  monitoring DHCP traffic, 449
    sample output, 454
  monitoring IPv6 traffic, 225
  verifying packet protection, 507-508
  viewing protected packets, 542, 544
sockets
  displaying socket status with netstat, 212
  IPsec security, 539
  security considerations, 499
softtoken keystore
  key storage with metaslot, 478, 545, 594, 599
Solaris cryptographic framework, IPsec, and, 540
Solaris IP Filter
  address pools
    appending, 643
    removing, 642
    viewing, 642
  address pools and, 613-614
  configuration file examples, 608
  creating
    log files, 647-648
  creating configuration files, 651-652
  deactivating, 625
    NAT, 624
    on a NIC, 630-631
  enabling in previous Solaris 10 releases, 626-628
  /etc/ipf/ipf.conf file, 651-652
  /etc/ipf/ipf6.conf file, 615-616
  /etc/ipf/ipnat.conf file, 651-652
  /etc/ipf/ippool.conf file, 651-652
  flush log file, 649
  guidelines for using, 608
  ifconfig command, 608
  ipf command, 621-622
    -6 option, 615-616
Solaris IP Filter *(Continued)*
  ipf.conf file, 609-611
  ipf6.conf file, 615-616
  ipfstat command
    -6 option, 615-616
  ipmon command
    IPv6 and, 615-616
  ipnat command, 621-622
  ipnat.conf file, 612-613
  ippool command, 642
    IPv6 and, 615-616
  ippool.conf file, 613-614
  IPv6, 615-616
  loopback filtering, 622-623
  managing packet filtering rule sets, 633-639
  NAT and, 612-613
  NAT rules
    appending, 641
    viewing, 640
  open source information, 604-605
  overview, 604-605
  packet filter hooks, 614, 620-621
  packet filtering overview, 609-611
  pfil module, 615
  re-enabling, 621-622
  removing
    NAT rules, 640-641
  rule set
    activating different, 634-636
  rule sets
    active, 633-634
    appending to active, 636-637
    appending to inactive, 637-638
    inactive, 634
    removing, 636
    removing inactive, 639
    switching between, 638-639
  rule sets and, 608-614
  saving logged packets to a file, 650
  specifying a NIC, 628-630
  viewing
    address pool statistics, 646-647
    log files, 648-649
    NAT statistics, 646

Solaris IP Filter, viewing *(Continued)*
pfil statistics, 631-632
state statistics, 645-646
state tables, 644-645
space-saving mode, in.routed daemon option, 254
SPI label, 688, 707, 708, 709
SPI section
labels and values, 706
Mobile IP configuration file, 705-706, 706, 707
standby interface
configuring for an IPMP group, 743-745
configuring test address on, 744
definition, 724
standby parameter
ifconfig command, 724, 744
state information, Mobile IP, 712
state statistics, viewing, 645-646
state tables, viewing, 644-645
stateless address autoconfiguration, 280
static routing, 128, 237
adding a static route, 121, 122-124
best uses, 122
configuration example, 123-124
host configuration example, 129
manually configuring on a host, 128
statistics
packet transmission (ping), 216, 217
per-protocol (netstat), 209
statistics for IPQoS
enabling class-based statistics, 850
enabling global statistics, 799, 850
generating, through the kstat command, 832
storing
IKE keys on disk, 573, 601, 602
IKE keys on hardware, 549, 593-594
subdivisions, administrative, 65
subnet prefix, IPv6, 76
subnets
IPv4
addresses and, 242
netmask configuration, 101
IPv4 addresses and, 243
IPv6
6to4 topology and, 291
subnets, IPv6 *(Continued)*
definition, 73
suggestions for numbering, 91
netmasks database, 241
editing /etc/inet/netmasks file, 243, 244
network mask creation, 242, 243
network configuration servers, 96
network masks
applying to IPv4 address, 242, 243
creating, 243
overview, 241
subnet number, IPv4, 241
subnet number in IPv4 addresses, 60
subnet prefix, IPv6, 76
Sun Crypto Accelerator 1000 board, 549
using with IKE, 592
Sun Crypto Accelerator 4000 board
accelerating IKE computations, 549
storing IKE keys, 549
using with IKE, 593-594
svcadm command
disabling network services, 515, 521, 526, 532
switch configuration
in a VLAN topology, 157
in an aggregation topology, 163
link aggregation control protocol (LACP) modes, 165, 168
symbolic names for network numbers, 244
SYN segment, 46
sys-unconfig command
and DHCP client, 425, 426
syslog.conf file logging for IPQoS, 823
systems, protecting communication, 496-499

## T

-T option
ikecert command, 577, 601
ikecert certlocal command, 566
-t option
ikecert certlocal command, 566
ikecert command, 600
inetd daemon, 132

target system, in IPMP
configuring, in a shell script, 742-743
configuring manually, 742
definition, 720
task map
IPQoS
configuration planning, 773
task maps
Changing IKE Transmission Parameters (Task Map), 594
Configuring IKE (Task Map), 553
Configuring IKE for Mobile Systems (Task Map), 583
Configuring IKE to Find Attached Hardware (Task Map), 591
Configuring IKE With Preshared Keys (Task Map), 554
Configuring IKE With Public Key Certificates (Task Map), 564
DHCP
IP address management decisions, 323
making decisions for DHCP server configuration, 319
modifying DHCP service options, 346
moving DHCP server configuration data, 410
preparing network for DHCP, 315
supporting BOOTP clients, 368
supporting information-only clients, 407
supporting remove boot and diskless clients with DHCP, 406
working with DHCP macros, 386
working with DHCP networks, 358
working with DHCP options, 396
working with IP addresses, 370
IPMP
dynamic reconfiguration (DR) administration, 734-735
IPMP group configuration, 733-734
IPQoS
configuration file creation, 793
flow-accounting setup, 829
QoS policy planning, 778
IPv4 network
adding subnets, 98
task maps *(Continued)*
IPv6
configuration, 176-177
planning, 83-84
tunnel configuration, 188-189
Mobile IP
configuration, 679-680
modifying a configuration, 684-685
network administration tasks, 203-204
network configuration, 94-95
Protecting a VPN With IPsec (Task Map), 511-536
Protecting Traffic With IPsec (Task Map), 495
TCP/IP networks
configuration files, 235
`/etc/defaultdomain` file, 237
`/etc/defaultrouter` file, 237
`/etc/hostname.`*interface* file, 236
`/etc/nodename` file, 104, 236
`hosts` database, 237, 239
`netmasks` database, 241
configuring
host configuration modes, 95, 97
local files mode, 102
network clients, 104
network configuration server setup, 102
network databases, 245, 247, 249
`nsswitch.conf` file, 247, 249
prerequisites, 94
standard TCP/IP services, 132
host configuration modes, 95, 97
local files mode, 95, 96
mixed configurations, 97
network client mode, 96
network configuration servers, 96
sample network, 97
IPv4 network configuration tasks, 99
IPv4 network topology, 97
network numbers, 38
protecting with ESP, 484
troubleshooting, 224
displaying packet contents, 222
general methods, 229
`ifconfig` command, 204
`netstat` command, 208

TCP/IP networks, troubleshooting *(Continued)*
packet loss, 216, 217
`ping` command, 216, 217
software checks, 230
third-party diagnostic programs, 229
TCP/IP protocol suite, 37
data communications, 44, 48
data encapsulation, 45, 48
displaying statistics, 209
dual-stack protocols, 86
further information, 48
books, 48
FYIs, 49
internal trace support, 48
OSI Reference Model, 38, 39
overview, 37, 38
standard services, 132
TCP/IP protocol architecture model, 39, 44
application layer, 39, 42, 44
data-link layer, 39, 40
Internet layer, 39, 40
physical network layer, 39, 40
transport layer, 39, 41
TCP protocol
description, 41
displaying statistics, 209
establishing a connection, 46
segmentation, 46
services in `/etc/inet/services` file, 253
TCP wrappers, enabling, 136
Telnet protocol, 43
temporary address, in IPv6
configuring, 182-184
definition, 181-184
test addresses, IPMP
configuring
IPv4, 737
IPv6, 738
on a standby interface, 744
definition, 722
IPv4 requirements, 722
IPv6 requirements, 723
preventing use by applications, 723
probe traffic and, 722
test addresses, IPMP *(Continued)*
standby interface, 724
`test` parameter, `ifconfig` command, 737
`tftp` protocol
description, 43
network configuration server booting protocol, 96
`/tftpboot` directory creation, 102
three-way handshake, 46
timestamps, 683, 703
token ID, in hardware, 602
Token ring, IPMP support for, 736
`tokenmt` meter, 766
color-awareness configuration, 766, 839
metering rates, 838
rate parameters, 838
single-rate meter, 839
two rate-meter, 839
`tokens` argument, `ikecert` command, 600
topology, 66
`traceroute` command
definition, 220-222
extensions for IPv6, 274
tracing routes, 221-222
traffic conformance
defining, 816
outcomes, 766, 838
planning
outcomes in the QoS policy, 785
rates in the QoS policy, 785
rate parameters, 838
traffic management
controlling flow, 766
forwarding traffic, 769, 770, 771
planning network topologies, 774
prioritizing traffic flows, 764
regulating bandwidth, 763
transition to IPv6, 6to4 mechanism, 289
transmission parameters
IKE global parameters, 595
IKE tuning, 594-596
transmission parameters (IKE), changing, 594
transport layer
data encapsulation, 46, 47
obtaining transport protocol status, 210-211

transport layer *(Continued)*
OSI, 39
packet life cycle
receiving host, 48
sending host, 46, 47
TCP/IP
description, 39, 41
SCTP protocol, 42, 133-136
TCP protocol, 41
UDP protocol, 42
transport mode
IPsec, 487-489
protected data with ESP, 488
protecting data with AH, 488
Triple-DES encryption algorithm, IPsec and, 486
troubleshooting
checking PPP links
packet flow, 222
DHCP, 441
IKE payload, 577
IKE transmission timing, 594-596
IPv6 problems, 231-233
TCP/IP networks
checking packets between client and server, 224
displaying interface status with `ifconfig` command, 204, 207
displaying status of known routes, 215-216
general methods, 229
monitoring network status with `netstat` command, 208
monitoring packet transfer with `snoop` command, 222
observing transmissions from interfaces, 212
obtaining per-protocol statistics, 209-210
obtaining transport protocol status, 210-211
packet loss, 216, 217
`ping` command, 217
probing remote hosts with `ping` command, 216
software checks, 230
third-party diagnostic programs, 229
`traceroute` command, 220-222
tracing `in.ndpd` activity, 219-220
tracing `in.routed` activity, 219
trunking, *See* aggregations
`tswtclmt` meter, 766, 840
metering rates, 840
`tun` module, 287
`tunnel` keyword
IPsec policy, 488, 510, 516, 522
tunnel mode
IPsec, 487-489
protecting entire inner IP packet, 488
tunneling, 664, 674, 677
tunnels
6to4 tunnels, 289
known problems, 232
packet flow, 291, 292
topology, 290
configuring IPv6
6to4 tunnels, 192
examples, 272
IPv4 over IPv6, 191
IPv6 over IPv4, 189-190
IPv6 over IPv6, 190
to a 6to4 relay router, 195
`ifconfig` security options, 542-543
IPsec, 489
IPv6, automatic
*See* tunnels, 6to4 tunnels
IPv6, manually configured, 287-289
IPv6 tunneling mechanisms, 285
modes in IPsec, 487-489
planning, for IPv6, 89
protecting packets, 489
topology, to 6to4 relay router, 292
transport mode, 487
tunnel mode, 487
turning on
an IPv6-enabled network, 176-177
network configuration daemons, 102
`Type` label, 689, 707, 708, 709

## U

UDP protocol
description, 42
displaying statistics, 209
services in `/etc/inet/services` file, 253

UDP protocol *(Continued)*
UDP packet process, 46
unicast datagram routing, Mobile IP, 674-675
uniform resource indicator (URI), for accessing CRLs, 581
UNIX "r" commands, 43
unusable DHCP address, 374, 380
`use_http` keyword, IKE configuration file, 582
user priority value, 767
`/usr/sbin/6to4relay` command, 195
`/usr/sbin/in.rdisc` program, description, 254
`/usr/sbin/in.routed` daemon
description, 254
space-saving mode, 254
`/usr/sbin/inetd` daemon
checking the status of `inetd`, 230
services started by, 132
`/usr/sbin/ping` command, 217
description, 216
running, 217
syntax, 216

## V

`-V` option
`snoop` command, 542, 544
`/var/inet/ndpd_state.interface` file, 275
verifying, packet protection, 507-508
`Version` label, 682, 701
viewing
IPsec configuration, 538-539
IPsec policy, 501
virtual LAN (VLAN) devices on an IPQoS network, 843
virtual private networks (VPNs)
configuring with `routeadm` command, 515, 525, 529
constructed with IPsec, 489
IPv4 example, 514-520
IPv6 example, 520-524
protecting with IPsec, 514-520
protecting with IPsec in tunnel transport mode, 524-530
visitor list
foreign agent, 692
visitor list *(Continued)*
Mobile IP, 711
VLAN
configuration, 156-161
configuration, in Solaris 10 3/05, 141-143
definition, 141-143, 156-161
interfaces supported in Solaris 10 1/06, 159
physical point of attachment (PPA), 142, 158
planning, 159
sample scenarios, 156
static
configuration, in Solaris 10 3/05, 142-143
switch configuration, 157
tag header format, 141
topologies, 156-158
virtual device, 143, 160
VLAN ID (VID), 141, 157-158
VPN, *See* virtual private networks (VPNs)

## W

web servers
configuring for IPQoS, 795, 797, 806, 807
protecting with IPsec, 499-501
wide area network (WAN)
Internet
domain name registration, 38
wildcards in `bootparams` database, 250
wireless communications
Mobile IP, 663, 667, 676
wrappers, TCP, 136

## Z

zone file, 197
zones
IPsec and, 491, 496
key management and, 496