

User Guide for Cisco Network Registrar

Software Release 7.0
December 2007

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number: OL-10272-01

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

CCVP, the Cisco logo, and Welcome to the Human Network are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PIX, ProConnect, ScriptShare, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0711R)

THIS PRODUCT INCLUDES THE FOLLOWING THIRD PARTY LICENSED SOFTWARE:

This product is distributed with Apache Tomcat 5.5.20 software developed by the Apache Software Foundation.
Copyright © 2004 The Apache Software Foundation. All rights reserved.
The list of conditions and disclaimer for the use of this software are included in the /docs/licenses directory of the installation directory.

This product is distributed with com.oreilly.servlet class library software.
Copyright © 2001-2002 by Jason Hunter. All rights reserved.
The list of conditions and disclaimer for the use of this software are included in the /docs/licenses directory of the installation directory.

This product is distributed with the Tool Command Language (Tcl) software as part of the standard Tcl/Tk distribution.
Copyright © The Regents of the University of California, Sun Microsystems, Inc., Scriptics Corporation, and other parties.
The terms and agreement for the use of this software are included in the /docs/licenses directory of the installation directory.

This product is distributed with the gtar 1.13 software.
Copyright © 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.
The terms and agreement for the use of this software are included in the /docs/licenses directory of the installation directory.

This product is distributed with Henry Spencer's regular expression library software, rxspencer-alpha 3.8.
Copyright © 1992, 1993, 1994, 1997 Henry Spencer. All rights reserved.
This software is not subject to any license of the American Telephone and Telegraph Company or of the Regents of the University of California.
The restrictions on the use of this software are included in the /docs/licenses directory of the installation directory.

This product is distributed with the JFreeChart 1.0.1 software.
Copyright © 1991, 1999 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.
The terms and agreement for the use of this software are included in the /doc/licenses directory of the installation directory.

User Guide for Cisco Network Registrar Release 7.0
Copyright © 1995 – 2007 Cisco Systems, Inc. All rights reserved.



CONTENTS

Preface xxv

- Who Should Read This Guide xxv
- How This Guide Is Organized xxv
  - Part 1—Getting Started xxv
  - Part 2—Local and Regional Administration xxvi
  - Part 3—Address Management xxvi
  - Part 4—Domain and Zone Administration xxvi
  - Part 5—Dynamic Host Administration xxvii
  - Part 6—Appendixes, Glossary, and Index xxvii
- Document Conventions xxviii
  - Formatting xxviii
  - Navigation and Screens xxviii
  - Callouts xxix
- Network Registrar Documentation xxix
- Obtaining Documentation and Submitting a Service Request xxix

PART 1 **Getting Started**

CHAPTER 1 **Network Registrar Components** 1-1

- Management Components 1-1
- Trivial File Transfer 1-2
  - Viewing and Editing the TFTP Server 1-2
  - Managing the TFTP Server Network Interfaces 1-3
- Simple Network Management 1-3
  - Setting Up the SNMP Server 1-4
  - How Notification Works 1-6
  - Handling SNMP Notification Events 1-7
  - Handling SNMP Queries 1-10
- Default Ports for Network Registrar Services 1-11

CHAPTER 2 **Network Registrar User Interfaces** 2-1

Introduction to the Web-Based User Interfaces 2-1
Supported Web Browsers 2-2
Access Security 2-2
Logging In to the Web UIs 2-2
Changing Passwords 2-3
Navigating the Web UIs 2-4
Waiting for Page Resolution Before Proceeding 2-4
Committing Changes in the Web UIs 2-4
Role and Attribute Visibility Settings 2-5
Displaying and Modifying Attributes 2-5
Grouping and Sorting Attributes 2-5
Modifying Attributes 2-6
Displaying Attribute Help 2-6
Help Pages 2-7
Logging Out 2-7
Local Cluster Web UI 2-7
Local Basic Main Menu Page 2-8
Local Advanced Main Menu Page 2-9
Setting Local User Preferences 2-10
Configuring Clusters in the Local Web UI 2-11
Regional Cluster Web UI 2-11
Command Line Interface 2-12
Central Configuration Management Server 2-13

CHAPTER 3 **Server Status Dashboard** 3-1

Opening the Dashboard 3-1
Display Types 3-2
Tables 3-3
Line Charts 3-3
Stacked Area Charts 3-5
Other Chart Types 3-5
Getting Help for the Dashboard Elements 3-6
Customizing the Display 3-6
Selecting Dashboard Elements to Include 3-7
Host Metrics 3-8
System Metrics 3-9
JVM Memory Utilization 3-10

DHCP Metrics 3-10
DHCP Server Request Activity 3-10
DHCP Server Response Activity 3-11
DHCP Buffer Capacity 3-12
DHCP Response Latency 3-12
DHCP DNS Updates 3-13
DHCP Address Current Utilization 3-13
DHCP Failover Status 3-14
DHCP General Indicators 3-15
DNS Metrics 3-15
DNS Query Responses 3-16
DNS Forwarding Errors 3-16
DNS Outbound Zone Transfers 3-17
DNS Inbound Zone Transfers 3-17
DNS Network Errors 3-18
DNS Related Servers Errors 3-18
DNS General Indicators 3-19

CHAPTER 4 **Deploying Network Registrar** 4-1
Target Users 4-1
Regional and Local Clusters 4-2
Deployment Scenarios 4-2
Small-to-Medium-Size LANs 4-3
Large Enterprise and Service Provider Networks 4-3
Configuration and Performance Guidelines 4-5
General Configuration Guidelines 4-5
Special Configuration Cases 4-6
Interoperability with Earlier Releases 4-6

PART 2 Local and Regional Administration

CHAPTER 5 Configuring Administrators 5-1

Administrators, Groups, and Roles 5-1
How Administrators Relate to Groups and Roles 5-1
Administrator Types 5-2
Roles, Subroles, and Constraints 5-2
Groups 5-4
Managing Administrators 5-5
Managing Groups 5-6
Managing Roles 5-6
Managing Passwords 5-7
Licensing 5-8
Centrally Managing Administrators 5-8
Pushing and Pulling Administrators 5-9
Pushing Administrators to Local Clusters 5-9
Pulling Administrators from the Replica Database 5-10
Pushing and Pulling Groups 5-11
Pushing Groups to Local Clusters 5-11
Pulling Groups from the Replica Database 5-12
Pushing and Pulling Roles 5-12
Pushing Roles to Local Clusters 5-13
Pulling Roles from the Replica Database 5-13
Pushing and Pulling Owners or Regions 5-14
Pushing Owners or Regions to Local Clusters 5-14
Pulling Owners and Regions from the Replica Database 5-15
Local Cluster Management Tutorial 5-15
Administrator Responsibilities and Tasks 5-16
Create the Administrators 5-16
Create the Address Infrastructure 5-18
Create the Zone Infrastructure 5-19
Create the Forward Zones 5-19
Create the Reverse Zones 5-20
Create the Initial Hosts 5-21
Create a Host Administrator Role with Constraints 5-21
Create a Group to Assign to the Host Administrator 5-23
Test the Host Address Range 5-24

Regional Cluster Management Tutorial 5-25
Administrator Responsibilities and Tasks 5-26
Create the Regional Cluster Administrator 5-26
Create the Central Configuration Administrator 5-26
Create the Local Clusters 5-27
Add a Router and Modify an Interface 5-28
Add Zone Management to the Configuration Administrator 5-29
Create a Zone for the Local Cluster 5-30
Pull Zone Data and Create a Zone Distribution 5-30
Create a Subnet and Pull Address Space 5-31
Push a DHCP Policy 5-31
Create a Scope Template 5-33
Create and Synchronize the Failover Pair 5-33

CHAPTER 6 **Managing the Central Configuration** 6-1
Central Configuration Tasks 6-1
Configuring Server Clusters 6-2
Adding Local Clusters 6-2
Editing Local Clusters 6-3
Listing Related Servers for DHCP Servers 6-4
Connecting to Local Clusters 6-6
Synchronizing with Local Clusters 6-7
Replicating Local Cluster Data 6-7
Viewing Replica Data 6-7
Deactivating, Reactivating, and Recovering Data for Clusters 6-8
Polling Subnet Utilization and Lease History Data 6-9
Polling Process 6-10
Adjusting the Polling Intervals 6-10
Enabling Subnet Utilization Collection 6-11
Enabling Lease History Collection 6-12
Managing DHCP Scope Templates 6-12
Pushing Scope Templates to Local Clusters 6-12
Pulling Scope Templates from Replica Data 6-13
Managing DHCP Policies 6-13
Pushing Policies to Local Clusters 6-14
Pulling Policies from Replica Data 6-14
Managing DHCP Client-Classes 6-15
Pushing Client-Classes to Local Clusters 6-15
Pulling Client-Classes from Replica Data 6-16

Managing Virtual Private Networks 6-16
Pushing VPNs to Local Clusters 6-16
Pulling VPNs from Replica Data 6-17
Managing DHCP Failover Pairs 6-17
Managing Lease Reservations 6-18

CHAPTER 7 **Maintaining Servers and Databases** 7-1
Managing Servers 7-1
Scheduling Recurring Tasks 7-4
Logging Server Events 7-6
Searching the Logs 7-6
Logging Format and Settings 7-7
Log Files 7-8
Change Logs and Tasks 7-9
Monitoring and Reporting Server Status 7-10
Server States 7-10
Displaying Health 7-10
Displaying Statistics 7-11
DNS Statistics 7-12
DHCP Statistics 7-13
TFTP Statistics 7-14
Displaying IP Address Usage 7-15
Displaying Related Servers 7-16
Monitoring Remote Servers Using Persistent Events 7-16
DNS Zone Distribution Servers 7-17
DHCP Failover Servers 7-18
Displaying Leases 7-18
Running Data Consistency Rules 7-19
Troubleshooting Servers 7-21
Immediate Troubleshooting Actions 7-21
Modifying the cnr.conf File 7-22
Troubleshooting Server Failures 7-22
Troubleshooting and Optimizing the TFTP Server 7-23
Tracing TFTP Server Activity 7-23
Optimizing TFTP Message Logging 7-24
Enabling TFTP File Caching 7-24
Solaris and Linux Troubleshooting Tools 7-25
Using the TAC Tool 7-25

CHAPTER 8 **Backup and Recovery** 8-1

Backing Up Databases 8-1
Syntax and Location 8-2
Backup Strategy 8-2
Setting Shadow Backup Times 8-3
Performing Manual Backups 8-3
Using Third Party Backup Programs with mcdshadow 8-3
Database Recovery Strategy 8-3
Backing Up and Recovering MCD Data 8-4
Checking MCD Database Integrity 8-5
Recovering MCD Data from Damaged Databases 8-5
Recovering MCD Data from Backups 8-5
MCD Recovery Errors 8-6
MCD Data Files 8-6
Backing Up CNRDB Data 8-7
Recovering CNRDB Data from Damaged Databases 8-7
Recovering CNRDB Data from Backups 8-9
Using Session Assert Commands for Data Management 8-10
Virus Scanning While Running Network Registrar 8-10
Troubleshooting Databases 8-10
Using the cnr_exim Data Import and Export Tool 8-11
Using the mcdadmin Tool 8-12
Using the cnrdb_recover Utility 8-14
Using the cnrdb_verify Utility 8-15
Using the cnrdb_checkpoint Utility 8-15
Using the keybuild Tool 8-15
Using the dbcheck Tool 8-16
Restoring DHCP Data from a Failover Server 8-16

PART 3 **Address Management**

CHAPTER 9 **Managing Address Space** 9-1

Address Block Administrator Role 9-1
Required Permissions 9-1
Role Functions 9-2
Viewing Address Space 9-2
Pulling Replica Address Space from Local Clusters 9-3

Address Blocks and Subnets 9-4
Viewing Address Blocks, Subnets, and Address Types 9-4
Knowing When to Add Address Blocks 9-5
Adding Address Blocks 9-5
Delegating Address Blocks 9-7
Pushing Subnets to Local DHCP Servers and Routers 9-8
Creating Reverse Zones from Subnets 9-9
Reclaiming Subnets 9-10
Adding Children to Address Blocks 9-10
Adding Address Ranges to Subnets 9-11
Viewing Address Utilization for Address Blocks, Subnets, and Scopes 9-12
Generating Subnet Utilization History Reports 9-14
Enabling Subnet Utilization History Collection at the Local Cluster 9-14
Querying Subnet Utilization History Data 9-15
Trimming and Compacting Subnet Utilization History Data 9-16
Viewing Subnet Utilization History Data 9-17

CHAPTER 10 **Managing Hosts** 10-1
Managing Hosts in Zones 10-1
Adding Additional RRs for the Host 10-2
Editing Hosts 10-3
Removing Hosts 10-3

CHAPTER 11 **Managing Router Interface Configurations** 11-1
Adding Routers 11-2
Managed Versus Virtual Routers 11-2
Secure Mode Connections with Routers 11-3
Alternative Login Method to Routers 11-3
Editing Routers 11-3
Resynchronizing Routers 11-4
Pushing and Reclaiming Subnets for Routers 11-4
Viewing and Editing the Router Interfaces 11-4
Changeable Router Interface Attributes 11-5
Bundling Interfaces 11-5

CHAPTER 12 **Managing Owners and Regions** 12-1
Managing Owners 12-1
Managing Regions 12-2

CHAPTER 13 **Managing Reports** 13-1

ARIN Reports and Allocation Reports 13-1

Managing ARIN Reports 13-1

Managing Point of Contact and Organization Reports 13-2

Creating a Point of Contact Report 13-3

Registering a Point of Contact 13-4

Editing a Point of Contact Report 13-4

Creating an Organization Report 13-5

Registering an Organization 13-6

Editing an Organization Report 13-6

Managing IPv4 Address Space Utilization Reports 13-7

Managing Shared WHOIS Project Allocation and Assignment Reports 13-8

PART 4 **Domain and Zone Administration**

CHAPTER 14 **Introduction to the Domain Name System** 14-1

How DNS Works 14-1

Domains 14-2

Learning ExampleCo Address 14-3

Establishing a Domain 14-3

Difference Between Domains and Zones 14-3

Nameservers 14-5

Reverse Nameservers 14-6

High-Availability DNS 14-7

CHAPTER 15 **Managing Zones** 15-1

Staged and Synchronous Modes 15-1

Creating and Applying Zone Templates 15-2

Managing Primary DNS Servers 15-5

Configuring Primary Forward Zones 15-5

Creating Primary Zones 15-6

Editing Primary Zones 15-8

Confirming Zone Nameservers 15-9

Synchronizing Zones and Zone Commands 15-9

Importing and Exporting Zone Data 15-9

Zone Lists and Zone Trees 15-12

Adding Primary Reverse Zones 15-12
Adding Reverse Zones as Zones 15-12
Adding Reverse Zones from Subnets 15-13
Getting Zone Counts on the Server 15-13
Managing Secondary Servers 15-14
Adding Secondary Forward Zones 15-14
Adding Secondary Reverse Zones 15-15
Enabling Zone Transfers 15-15
Adding Subzones 15-16
Choosing Subzone Names and Servers 15-16
Creating and Delegating Subzones 15-17
Undelegating Subzones 15-18
Editing Subzone Delegation 15-18
Enabling DNS Updates 15-19
Managing Zone Distributions 15-19
Preparing the Zone Distribution Map 15-20
Creating a Zone Distribution 15-21
Pulling Zone Distributions from Replica Data 15-24

CHAPTER 16 **Managing Resource Records** 16-1
Managing Resource Records 16-1
Adding Resource Records 16-2
Protecting Resource Record Sets 16-3
Editing Resource Records 16-4
Removing Resource Records 16-4
Removing Cached Records 16-5
Listing Records 16-5
Searching Server-Wide for Records and Addresses 16-6
Filtering Records 16-8
Deleting Leftover Zone Records After Recreating Zones 16-9
Using Service Location (SRV) Records 16-9
Using NAPTR Records 16-9
Adding IPv6 Records 16-11
Managing Hosts in Zones 16-11

CHAPTER 17 **Managing DNS Server Properties** 17-1

Managing DNS Servers 17-1
Running DNS Commands 17-1
Configuring DNS Server Network Interfaces 17-2
Setting DNS Server Properties 17-2
Setting General DNS Server Properties 17-3
Defining Forwarders for DNS Servers 17-3
Setting Subzone Forwarding 17-4
Using Resolution Exception 17-4
Configuring Caching-Only DNS Servers 17-6
Specifying Delegation-Only Zones 17-7
Defining Root Nameservers 17-7
Enabling Recursive Queries 17-8
Enabling Round-Robin 17-8
Enabling Subnet Sorting 17-9
Enabling Incremental Zone Transfers (IXFR) 17-9
Changesets and Checkpointing 17-10
Restricting Zone Queries 17-10
Enabling NOTIFY 17-11
Setting Advanced DNS Server Properties 17-11
Setting SOA Time to Live 17-12
Setting Secondary Refresh Times 17-12
Setting Secondary Retry Times 17-13
Setting Secondary Expiration Times 17-13
Fetching Glue Records 17-13
Reporting Lame Delegation 17-14
Setting Maximum Negative Cache Times 17-14
Setting Maximum Cache TTLs 17-15
Setting Maximum Memory Cache Sizes 17-15
Flushing DNS Cache 17-15
Handling Rogue Address Records and Other Cache Attributes 17-16
Setting Query Source Addresses and Port Numbers 17-17
Setting Local and External Port Numbers 17-17
Tuning DNS Properties 17-17
Troubleshooting DNS Servers 17-18

CHAPTER 18 **Configuring High-Availability DNS Servers** 18-1

HA DNS Processing 18-1

Configuring an HA DNS Server Pair 18-2

DNS Server Configuration for HA DNS 18-4

HA DNS Statistics 18-4

PART 5 **Dynamic Host Administration**

CHAPTER 19 **Introduction to Dynamic Host Configuration** 19-1

How DHCP Works 19-1

Sample DHCP User 19-2

Typical DHCP Administration 19-2

Leases 19-3

Scopes and Policies 19-3

Network Registrar DHCP Implementations 19-4

DHCP and IPv6 19-4

Virtual Private Networks 19-4

Subnet Allocation and DHCP Address Blocks 19-5

DNS Update 19-6

Effect on DNS of Obtaining Leases 19-7

Effect on DNS of Releasing Leases 19-7

Effect on DNS of Reacquiring Leases 19-8

DHCP Failover 19-8

How Failover Works 19-8

Failover States and Transitions 19-9

Allocating Addresses Through Failover 19-11

Client-Classes 19-11

DHCP Processing Without Client-Classes 19-12

DHCP Processing with Client-Classes 19-13

Defining Scopes for Client-Classes 19-14

Choosing Networks and Scopes 19-14

CHAPTER 20 **Configuring Scopes and Networks** 20-1

Configuring DHCP Servers 20-1
General Configuration Guidelines 20-2
Configuring DHCP Server Interfaces 20-2
Defining and Configuring Scopes 20-3
Creating and Applying Scope Templates 20-3
Using Expressions in Scope Templates 20-4
Additional Scope Template Attributes 20-7
Editing Scope Templates 20-8
Applying Scope Templates to Scopes 20-8
Cloning a Scope Template 20-9
Creating Scopes 20-9
Getting Scope Counts on the Server 20-11
Configuring Multiple Scopes 20-11
Configuring Multiple Scopes for Round-Robin Address Allocation 20-12
Configuring Multiple Scopes Using Allocation Priority 20-12
Editing Scopes 20-17
Staged and Synchronous Mode 20-18
Configuring Embedded Policies for Scopes 20-19
Configuring Multiple Subnets on a Network 20-19
Enabling and Disabling BOOTP for Scopes 20-20
Disabling DHCP for Scopes 20-21
Deactivating Scopes 20-21
Setting Scopes to Renew-Only 20-21
Setting Free Address SNMP Traps on Scopes 20-22
Removing Scopes 20-23
Removing Scopes if Not Reusing Addresses 20-23
Removing Scopes if Reusing Addresses 20-24
Managing DHCP Networks 20-24
Listing Networks 20-24
Editing Networks 20-24

CHAPTER 21 **Configuring Policies and Options** 21-1

Configuring DHCP Policies 21-1
Types of Policies 21-1
Policy Hierarchy 21-3
Creating and Applying DHCP Policies 21-3
Cloning a Policy 21-6
Setting DHCP Options and Attributes for Policies 21-6
Adding Option Values 21-6
Adding Complex Values for Suboptions 21-7
Creating and Editing Embedded Policies 21-8
Creating DHCP Option Definition Sets and Option Definitions 21-9
Using Standard Option Definition Sets 21-10
Creating Custom Option Definitions 21-12
Creating Vendor-Specific Option Definitions 21-12
Option Definition Data Types and Repeat Counts 21-14
Adding Suboption Definitions 21-15
Importing and Exporting Option Definition Sets 21-15
Pushing Option Definition Sets to Local Clusters 21-16
Pulling Option Definition Sets from Replica Data 21-16
Setting Option Values for Policies 21-17

CHAPTER 22 **Managing Leases** 22-1

Configuring Leases in Scopes 22-1
Viewing Leases 22-1
Lease States 22-4
Guidelines for Lease Times 22-4
Importing and Exporting Lease Data 22-5
Pinging Hosts Before Offering Addresses 22-7
Deactivating Leases 22-8
Excluding Leases from Ranges 22-8
Searching Server-Wide for Leases 22-9
Creating Lease Reservations 22-11
Setting Advanced Lease and Reservation Properties 22-14
Reserving Currently Leased Addresses 22-14
Unreserving Leases 22-16
Extending Reservations to Non-MAC Addresses 22-16
Forcing Lease Availability 22-18
Inhibiting Lease Renewals 22-18
Handling Leases Marked as Unavailable 22-19

Setting Timeouts for Unavailable Leases 22-20
Running Address and Lease Reports 22-21
Running Address Usage Reports 22-21
Running IP Lease Histories 22-21
Enabling Lease History Recording at the Local Cluster 22-22
Querying IP Lease History 22-22
Trimming Lease History Data 22-26
Running Lease Utilization Reports 22-27
Receiving Lease Notification 22-27
Running Lease Notification Automatically in Solaris and Linux 22-28
Running Lease Notification Automatically in Windows 22-28
Specifying Configuration Files for Lease Notification 22-29
Querying Leases 22-29
Leasequery Implementations 22-29
Pre-RFC Leasequery for DHCPv4 22-30
RFC 4388 Leasequery for DHCPv4 22-31
Leasequery for DHCPv6 22-31
Leasequery Statistics 22-32
Leasequery Example 22-32

CHAPTER 23 **Advanced DHCP Server Properties** 23-1
Configuring BOOTP 23-1
About BOOTP 23-2
Enabling BOOTP for Scopes 23-2
Moving or Decommissioning BOOTP Clients 23-3
Using Dynamic BOOTP 23-3
BOOTP Relay 23-3
Defining Advanced Server Parameters 23-4
Setting Advanced DHCP Server Parameters 23-4
Deferring Lease Extensions 23-6
Integrating Windows System Management Servers 23-7
Using Extensions to Affect DHCP Server Behavior 23-9
Writing Extensions 23-9
Preventing Chatty Clients by Using an Extension 23-11
Tuning the DHCP Server 23-12

Configuring Virtual Private Networks and Subnet Allocation 23-14
Configuring Virtual Private Networks Using DHCP 23-15
Typical Virtual Private Networks 23-15
Creating and Editing Virtual Private Networks 23-16
VPN Usage 23-18
Configuring DHCP Subnet Allocation 23-19
VPN and Subnet Allocation Tuning Parameters 23-21
Setting DHCP Forwarding 23-21

CHAPTER 24 **Configuring Client-Classes and Clients** 24-1
Configuring Client-Classes 24-1
Client-Class Process 24-2
Defining Client-Classes 24-2
Setting Selection Tags on Scopes and Prefixes 24-4
Defining Client-Class Hostname Properties 24-4
Editing Client-Classes and Their Embedded Policies 24-5
Processing Client Data Including External Sources 24-6
Processing Order to Determine Client-Classes 24-7
Processing Order to Determine Selection Tags 24-7
Troubleshooting Client-Classes 24-8
Configuring Clients 24-9
Editing Clients and Their Embedded Policies 24-11
Setting Windows Client Properties 24-11
Allocating Provisional Addresses 24-12
Skipping Client Entries for Client-Classing 24-13
Limiting Client Authentication 24-13
Setting Client Caching Parameters 24-14
Subscriber Limitation Using Option 82 24-14
General Approach to Subscriber Limitation 24-15
Typical Limitation Scenario 24-16
Calculating Client-Classes and Creating Keys 24-16
Client-Class Lookup Expression Processing 24-16
Limitation Processing 24-16
Expression Processing for Subscriber Limitation 24-17
Configuring Option 82 Limitation 24-17
Lease Renewal Processing for Option 82 Limitation 24-18
Administering Option 82 Limitation 24-18
Troubleshooting Option 82 Limitation 24-19
Expression Examples 24-19

Configuring Network Registrar to Use LDAP 24-19
About LDAP Directory Servers 24-20
Adding and Editing LDAP Remote Servers 24-20
Configuring DHCP Client Queries in LDAP 24-21
Configuring DHCP LDAP Update and Create Services 24-24
Lease State Attributes 24-25
Configuring DHCP to Write Lease States to LDAP 24-26
Using LDAP Updates 24-27
Configuring LDAP State Updates 24-27
Configuring LDAP Entry Creation 24-29
Troubleshooting LDAP 24-30
LDAP Connection Optimization 24-30
Recommended Values for LDAP 24-31

CHAPTER 25 **Using Expressions** 25-1
Using Expressions 25-1
Entering Expressions 25-2
Creating Expressions 25-3
Expression Syntax 25-4
Expression Datatypes 25-4
Literals in Expressions 25-5
Expressions Return Typed Values 25-5
Expressions Can Fail 25-6
Expression Functions 25-6
Datatype Conversions 25-19
Expressions in the CLI 25-20
Expression Examples 25-21
Limitation Example 1: DOCSIS Cable Modem 25-21
Limitation Example 2: Extended DOCSIS Cable Modem 25-22
Limitation Example 3: DSL Over Asynchronous Transfer Mode 25-23
Debugging Expressions 25-24

CHAPTER 26 **Managing DHCPv6 Addresses** 26-1

DHCPv6 Concepts 26-2
IPv6 Addressing 26-2
Links and Prefixes 26-2
Determining Links and Prefixes 26-3
Generating Addresses 26-3
Generating Delegated Prefixes 26-4
DHCPv6 Clients and Leases 26-4
DHCPv6 Bindings 26-5
Lease Affinity 26-5
Lease Life Cycle 26-6
DHCPv6 Reservations 26-7
Searching for Leases 26-7
Querying Leases for DHCPv6 26-7
DHCPv6 Policy Hierarchy 26-7
DHCPv6 Options 26-8
DHCPv6 Configuration 26-8
Viewing IPv6 Address Space 26-9
Configuring Links 26-9
Creating and Editing Link Templates 26-9
Creating and Editing Links 26-12
Configuring Prefixes 26-14
Creating and Editing Prefix Templates 26-14
Creating and Editing Prefixes 26-18
Viewing Address Utilization for Prefixes 26-22
Viewing DHCPv6 Networks 26-24
Editing DHCPv6 Server Attributes 26-25
Configuring DHCPv6 Policies 26-25
Configuring DHCPv6 Client-Classes 26-26
Configuring DHCPv6 Clients 26-26
Setting DHCPv6 Options 26-27
Reconfigure Support 26-27
DNS Update for DHCPv6 26-29

CHAPTER 27 **Configuring DHCP Failover** 27-1

Failover Scenarios 27-1
Simple Failover 27-1
Back Office Failover 27-2
Symmetrical Failover 27-3
Failover Checklist 27-3
Creating and Synchronizing Failover Server Pairs 27-4
Adding Failover Pairs 27-4
Synchronizing Failover Pairs 27-6
Restarting the Failover Servers 27-8
Confirming Failover 27-9
State Transitions During Integration 27-9
Setting Advanced Failover Attributes 27-12
Setting Backup Percentages 27-13
Server and Scope Backup Percentages 27-14
BOOTP Backup Percentage 27-14
Setting Backup Allocation Boundaries 27-15
Setting the Maximum Client Lead Time 27-15
Using the Failover Safe Period to Move Servers into PARTNER-DOWN State 27-16
Setting DHCP Request and Response Packet Buffers 27-18
Changing Polling Attributes 27-18
Setting the Network Discovery Attribute 27-19
Setting Load Balancing 27-19
Load Balancing Compatibility with Earlier Network Registrar Versions 27-19
Configuring Load Balancing 27-19
Changing Failover Server Roles 27-20
Making Nonfailover Servers Failover Mains 27-20
Replacing Servers Having Defective Storage 27-21
Removing Backup Servers and Halting Failover Operation 27-21
Adding Main Servers to Existing Backup Servers 27-21
Configuring Failover on Multiple Interface Hosts 27-22
Supporting BOOTP Clients in Failover 27-22
Static BOOTP 27-22
Dynamic BOOTP 27-22
Configuring BOOTP Relays 27-23
DHCPLEASEQUERY and Failover 27-23
Troubleshooting Failover 27-23
Monitoring Failover Operations 27-23

Detecting and Handling Network Failures 27-24

CHAPTER 28 **Configuring DNS Update** 28-1

DNS Update Process 28-2

Special DNS Update Considerations 28-2

DNS Update for DHCPv6 28-2

DHCPv6 Upgrade Considerations 28-3

Generating Synthetic Names in DHCPv6 28-3

Determining Reverse Zones for DNS Updates 28-4

Using the Client FQDN 28-5

Creating DNS Update Configurations 28-5

Creating DNS Update Maps 28-7

Configuring Access Control Lists and Transaction Security 28-9

Access Control Lists 28-9

Configuring Zones for Access Control Lists 28-10

Transaction Security 28-11

Creating TSIG Keys 28-11

Generating Keys 28-12

Considerations for Managing Keys 28-13

Adding Supporting TSIG Attributes 28-13

Configuring DNS Update Policies 28-13

Compatibility with Previous Network Registrar Releases 28-13

Creating and Editing Update Policies 28-14

Defining and Applying Rules for Update Policies 28-14

Defining Rules for Named Update Policies 28-15

Applying Update Policies to Zones 28-17

Confirming Dynamic Records 28-17

Scavenging Dynamic Records 28-18

Troubleshooting DNS Update 28-19

Configuring DNS Update for Windows Clients 28-20
Client DNS Updates 28-20
Dual Zone Updates for Windows Clients 28-22
DNS Update Settings in Windows Clients 28-22
Windows Client Settings in DHCP Servers 28-23
SRV Records and DNS Updates 28-23
Recommended Windows Design Practices 28-25
Windows Rules and Suggestions 28-25
Naming Rules for Windows Domains 28-26
Issues Related to Windows Environments 28-26
Frequently Asked Questions About Windows Integration 28-30

CHAPTER 29 **Using Extension Points** 29-1
Using Extensions 29-1
Creating, Editing, and Attaching Extensions 29-2
Determining Tasks 29-3
Deciding on Approaches 29-4
Choosing Extension Languages 29-4
Language-Independent API 29-4
Routine Signature 29-5
Dictionaries 29-5
Utility Methods in Dictionaries 29-6
Configuration Errors 29-6
Communicating with External Servers 29-6
Recognizing Extensions 29-7
Tcl Extensions 29-7
Tcl Application Program Interface 29-7
Dealing with Tcl errors 29-8
Handling Boolean Variables in Tcl 29-8
Configuring Tcl Extensions 29-8
Init-Entry Extension Point in Tcl 29-9
C/C++ Extensions 29-9
C/C++ API 29-9
Using Types in C/C++ 29-10
Building C/C++ Extensions 29-10
Using Thread-Safe Extensions in C/C++ 29-10
Configuring C/C++ Extensions 29-11

Debugging C/C++ Extensions 29-11
Pointers into DHCP Server Memory in C/C++ 29-12
Init-Entry Entry Point in C/C++ 29-12
DHCP Request Processing Using Extensions 29-12
Enabling DHCPv6 Extensions 29-14
Receiving Packets 29-14
Decoding Packets 29-14
Determining Client-Classes 29-15
Modifying Client-Classes 29-15
Processing Client-Classes 29-15
Building Response Containers 29-16
Determining Networks and Links 29-16
Finding Leases 29-16
Serializing Lease Requests 29-17
Determining Lease Acceptability 29-17
DHCPv6 Leasing 29-19
DHCPv6 Prefix Usability 29-19
DHCPv6 Lease Usability 29-19
DHCPv6 Lease Allocation 29-20
Gathering Response Packet Data 29-20
Encoding Response Packets 29-20
Updating Stable Storage 29-21
Sending Packets 29-21
Processing DNS Requests 29-21
Tracing Lease State Changes 29-21
Extension Dictionaries 29-22
Environment Dictionary 29-22
Environment Dictionary Data Items 29-23
Initial Environment Dictionary 29-25
Request and Response Dictionaries 29-25
Decoded DHCP Packet Data Items 29-26
Using Parameter List Option 29-27
Extension Point Descriptions 29-27
init-entry 29-28
pre-packet-decode 29-28
post-packet-decode 29-29
post-class-lookup 29-30

pre-client-lookup 29-30
Environment Dictionary for pre-client-lookup 29-31
Client-Class Data Input to pre-client-lookup 29-31
Request Dictionary for pre-client-lookup 29-33
post-client-lookup 29-34
Environment Dictionary for post-client-lookup 29-34
Request Dictionary for post-client-lookup 29-35
generate-lease 29-35
check-lease-acceptable 29-36
lease-state-change 29-37
Response Dictionary for lease-state-change 29-37
Environment Dictionary for lease-state-change 29-38
pre-packet-encode 29-39
Request Dictionary for pre-packet-encode 29-39
Response Dictionary for pre-packet-encode 29-39
post-packet-encode 29-41
pre-dns-add-forward 29-42
post-send-packet 29-42
environment-destructor 29-42
Handling Objects and Options 29-43
Using Object and Option Handling Methods 29-43
Options and Suboptions in C/C++ 29-43

PART 6 **Appendices, Glossary, and Index**

APPENDIX A **Resource Records** A-1

APPENDIX B **DHCP Options** B-1
Option Descriptions B-1
RFC 1497 Vendor Extensions B-1
IP Layer Parameters Per Host B-3
IP Layer Parameters Per Interface B-4
Link Layer Parameters Per Interface B-4
TCP Parameters B-5
Application and Service Parameters B-5
DHCPv4 Extension Options B-8
DHCPv6 Options B-10
Microsoft Client Options B-13

Option Tables B-14
Options by Number B-14
Options by Network Registrar Name B-18
Option Validation Types B-24

APPENDIX C **DHCP Extension Dictionary** C-1
Extension Dictionary Entries C-1
Decoded DHCP Packet Data Items C-1
Request Dictionary C-9
Response Dictionary C-15
Extension Dictionary API C-22
Tcl Attribute Dictionary API C-23
Tcl Request and Response Dictionary Methods C-23
Tcl Environment Dictionary Methods C-25
DEX Attribute Dictionary API C-27
DEX Request and Response Dictionary Methods C-27
DEX Environment Dictionary Methods C-35
Examples of Option and Object Method Calls C-37
Handling Vendor Class Option Data C-37
Handling Object Data C-37

GLOSSARY

INDEX



Preface

This guide describes configuring Cisco Network Registrar by using the web-based user interface (web UI) and command line interface (CLI).

Who Should Read This Guide

This guide is designed for network managers who are responsible for maintaining the network Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), Trivial File Transfer Protocol (TFTP), and Simple Network Management Protocol (SNMP) servers. The network manager should be familiar with the following topics:

- Basic concepts and terminology used in internetworking
- Network topology and protocols

How This Guide Is Organized

This guide describes how to become familiar with Network Registrar features so that you can use them to administer network addresses. The parts of this guide are described in the following subsections.

Part 1—Getting Started

Getting Started introduces Network Registrar, describes the management and protocol components, and describes the user interfaces. This part includes the following chapters:

Chapter 1	Network Registrar Components	Introduces Network Registrar, its deployment scenarios, and some deployment guidelines.
Chapter 2	Network Registrar User Interfaces	Describes the Network Registrar management and protocol components.
Chapter 3	Server Status Dashboard	Describes the Network Registrar server status dashboard features and functions.
Chapter 4	Deploying Network Registrar	Describes the Network Registrar local and regional web UIs and CLIs.

Part 2—Local and Regional Administration

Local and Regional Administration describes how to configure administrators, manage the central configuration, and maintain the servers and databases (including backup and recovery). This part includes the following chapters:

Chapter 5	Configuring Administrators	Describes how to configure the local and regional administrators, and provides administration tutorials.
Chapter 6	Managing the Central Configuration	Describes how to manage the central network configuration from the regional cluster.
Chapter 7	Maintaining Servers and Databases	Describes how to maintain the Network Registrar servers.
Chapter 8	Backup and Recovery	Describes how to back up or recover the databases.

Part 3—Address Management

Address Management describes how to manage the IP address space and its hierarchy, hosts, Router Interface Configuration (RIC) servers, owners and regions, and reports. This part includes the following chapters:

Chapter 9	Managing Address Space	Describes how to manage address space elements known as address blocks and subnets.
Chapter 10	Managing Hosts	Describes how to manage network hosts.
Chapter 11	Managing Router Interface Configurations	Describes how to manage the RIC server.
Chapter 12	Managing Owners and Regions	Describes how to manage network owners and regions.
Chapter 13	Managing Reports	Describes how to manage American Registry of Internet Numbers (ARIN) and address allocation reports.

Part 4—Domain and Zone Administration

Domain and Zone Administration describes how to configure DNS servers, zones, resource records, server attributes, and High Availability (HA) servers. This part includes the following chapters:

Chapter 14	Introduction to the Domain Name System	Introduces the Domain Name System (DNS) protocol and its Network Registrar implementation.
Chapter 15	Managing Zones	Describes how to manage DNS zones.

Chapter 16	Managing Resource Records	Describes how to manage DNS resource records (RRs).
Chapter 17	Managing DNS Server Properties	Describes how to set more advanced DNS server properties.
Chapter 18	Configuring High-Availability DNS Servers	Describes how to configure a High Availability (HA) DNS server.

Part 5—Dynamic Host Administration

Dynamic Host Administration describes DHCP and how to configure scopes and leases and their several deployments, IPv6 addresses, clients and client-classes, failover, DNS Update, and special processing using extensions. This part includes the following chapters:

Chapter 19	Introduction to Dynamic Host Configuration	Introduces DHCP and its Network Registrar implementation.
Chapter 20	Configuring Scopes and Networks	Describes how to configure scopes and networks.
Chapter 21	Configuring Policies and Options	Describes how to configure policies and options.
Chapter 22	Managing Leases	Describes how to manage leases.
Chapter 23	Advanced DHCP Server Properties	Describes how to manage the more advanced DHCP server properties.
Chapter 24	Configuring Client-Classes and Clients	Describes how to configure DHCP clients and client-classes.
Chapter 25	Using Expressions	Describes how to use expressions for DHCP processing.
Chapter 26	Managing DHCPv6 Addresses	Describes how to manage the DHCPv6 address space.
Chapter 27	Configuring DHCP Failover	Describes how to configure DHCP failover servers.
Chapter 28	Configuring DNS Update	Describes how to configure DNS Update for DHCP.
Chapter 29	Using Extension Points	Describes how to use extensions for DHCP processing.

Part 6—Appendixes, Glossary, and Index

Appendices, Glossary, and Index includes appendixes that describe DNS RRs, DHCP options, and the DHCP extension dictionary. This part also includes a glossary and an index.

Appendix A	Resource Records	Describes the DNS RRs.
Appendix B	DHCP Options	Describes the DHCP options.
Appendix C	DHCP Extension Dictionary	Describes the DHCP extension dictionary.
Glossary	Glossary	Glossary of terms used in Network Registrar.
Index	Index	Index to the guide.

Document Conventions

This guide uses the following documentation conventions.

Formatting

This guide uses the following formatting conventions:

- User input and controls are indicated in **bold**; for example, “enter **1234**” and “click **Modify Scope**.”
- Object attributes are indicated in *italics*; for example, “the *failover-safe-period* attribute.”
- Cross-references to chapters or sections of chapters are indicated in blue type; for example, “see the “Document Conventions” section on page xxx.”

Navigation and Screens

This guide uses the following navigation and screen display conventions:

- Windows systems use a two-button mouse. To drag and drop an object, click and hold the left mouse button on the object, drag the object to the target location, then release the button.
- Solaris systems use a three-button mouse. To drag and drop an object, click and hold the middle mouse button on the object, drag the object to the target location, then release the button.
- Screen displays can differ slightly from those included in this guide, depending on the system or browser you use.
- Web UI Navigation bar labels can have IPv4 and IPv6 variants depending on the administrator role privileges assigned. To simplify procedural instructions, this *User Guide* uses the most generic versions of the menu bar labels, unless there is a need to be more specific. For example, the **Address Space** menu label might be rendered as **Address Space v4** and **Address Space v6**. The instructions will have the label simply as **Address Space**.

Callouts

Callouts in the text have the following meaning:



Caution *Be careful*. The description alerts you to potential data damage or loss.



Note *Take note*. The description is particularly noteworthy.



Timesaver *Save time*. The description can present a timesaver.



Tip *Consider this helpful hint*. The description can present an optimum action to take.

Network Registrar Documentation

The Network Registrar version 7.0 documentation set consists of:

- *Release Notes for Cisco Network Registrar Release 7.0*
- *Installation Guide for Cisco Network Registrar Release 7.0*
- *Quick Start Guide for Cisco Network Registrar Release 7.0*
- *User Guide for Cisco Network Registrar Release 7.0*
- Command Reference Guide as the **CLIContents.html** file in the /docs directory
- Online help as part of the Cisco Network Registrar web UI application

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Subscribe to the *What's New in Cisco Product Documentation* as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS version 2.0.



PART 1

Getting Started



CHAPTER 1

Network Registrar Components

Cisco Network Registrar provides the tools to configure and control the servers necessary to manage your IP address space. This chapter provides an overview of the management components and concentrates on the Trivial File Transfer Protocol (TFTP) and Simple Network Management Protocol (SNMP), which are not covered in subsequent parts of this *User Guide*.

Management Components

Network Registrar contains two management components:

- Regional component, consisting of:
  - Web-based user interface (web UI)
  - Command line interface (CLI)
  - Central Configuration Management (CCM) server to provide to local cluster, address space, and router management
- Local component, consisting of:
  - Web UI
  - CLI
  - CCM server
  - Domain Name System (DNS) server
  - Dynamic Host Configuration Protocol (DHCP) server
  - Trivial File Transport Protocol (TFTP) server
  - Simple Network Management Protocol (SNMP) server
  - Router Interface Configuration (RIC) server
  - Management of local address space, zones, scopes, DHCPv6 prefixes and links, and users

The remainder of this chapter describes the TFTP and SNMP protocols. The CCM server, web UIs, and CLI are described in Chapter 2, "Network Registrar User Interfaces." The DNS, DHCP, and RIC servers are described in their respective sections of this manual.

Trivial File Transfer

The Trivial File Transfer Protocol (TFTP) is a way of transferring files across the network using the User Datagram Protocol (UDP), a connectionless TCP/IP transport layer protocol. Network Registrar maintains a TFTP server so that systems can provide device provisioning files to cable modems that comply with the Data Over Cable Service Interface Specification (DOCSIS) standard. The TFTP server buffers the DOCSIS file in its local memory as it sends the file to the modem. After a TFTP transfer, the server flushes the file from local memory. TFTP also supports non-DOCSIS configuration files.

Here are some of the features of the Network Registrar TFTP server:

- Complies with RFCs 1123, 1350, 1782, and 1783
- Includes a high performance multithreaded architecture
- Supports IPv6
- Caches data for performance enhancements
- Is configurable and controllable in the web UI and using the **tftp** command in the CLI
- Includes flexible path and file access controls
- Includes audit logging of TFTP connections and file transfers
- Has a default root directory in the Network Registrar *install-path*/data/tftp

See Also

Viewing and Editing the TFTP Server
Managing the TFTP Server Network Interfaces, page 1-3

Viewing and Editing the TFTP Server

At the local cluster, you can edit the TFTP server to modify its attributes. You must be assigned the server-management subrole of the ccm-admin role.

Local Basic or Advanced Web UI

Step 1 Click **Servers**, then **Manage Servers** to open the Manage Servers page (see the "Managing Servers" section on page 7-1).

Step 2 Click the Local TFTP Server link to open the Edit TFTP Server page.

Step 3 Values for the attributes marked with asterisks are required for TFTP server operation. You can click the name of any attribute to open a description window for the attribute.

Step 4 To unset any attribute value, click the check box in the Unset? column, then click **Unset Fields** at the bottom of the page. To modify values you change or unset, click **Modify Server**, or **Cancel** to cancel the changes.

CLI Commands

Use **tftp** to show the attribute values. Use **tftp set** *attribute*=*value* or **tftp enable** *attribute* to set or enable attributes. You can also use **tftp serverLogs show**, and **tftp serverLogs nlogs=***number* **logsize=***size*.

Managing the TFTP Server Network Interfaces

You can manage the network interfaces for the TFTP server.

Local Advanced Web UI

Manage the network interfaces associated with the TFTP server by clicking the icon in the Interfaces column of the TFTP server on the Manage Servers page. You can view the default configured network interfaces, and create and edit additional ones. To create and edit them, you must be assigned the server-management subrole of the ccm-admin role.

The columns on the Manage TFTP Server Network Interface page are:

- **Name**—Name of the network interface, such as the LAN adapter, loopback, and Fast Ethernet interfaces. If the name is under the Configured Interfaces column, you can edit and delete the interface. Clicking the name opens the Edit TFTP Server Network Interface page so that you can edit the interface name and addresses. Make changes, then click **Modify Interface** on this page.
- **IP Address**—IP address of the network interface.
- **IPv6 Address**—IPv6 address, if applicable, of the network interface.
- **Flags**—Flags for whether the interface should be zero-broadcast, virtual, v4, v6, no-multicast, or receive-only.
- **Configure**—To configure a new network interface, click the Configure icon next to the interface name. This creates another interface based on the one selected, but with a more general IP address, and adds this interface to the Configured Interfaces for this TFTP Server.
- **Configured Interfaces for this TFTP Server**—User-configured network interfaces, showing each name and associated address. Click the interface name to edit it, or click the Delete icon to delete it.

To return to managing the server, click **Return**.

CLI Commands

Use the **tftp-interface** commands.

Simple Network Management

The Network Registrar Simple Network Management Protocol (SNMP) notification support allows you to query the DHCP and DNS counters, be warned of error conditions and possible problems with the DNS and DHCP servers, and monitor threshold conditions that can indicate failure or impending failure conditions.

Network Registrar implements SNMP Trap Protocol Data Units (PDUs) according to the SNMPv2c standard. Each trap PDU contains:

- Generic-notification code, if enterprise-specific.
- A specific-notification field that contains a code indicating the event or threshold crossing that occurred.
- A variable-bindings field that contains additional information about certain events.

Refer to the Management Information Base (MIB) for the details. The SNMP server supports only reads of the MIB attributes. Writes to the attributes are not supported.

The following MIB files are required:

- **Traps**—CISCO-NETWORK-REGISTRAR-MIB.my
- **DNS server**—CISCO-DNS-SERVER-MIB.my
- **DHCPv4 server**—CISCO-IETF-DHCP-SERVER-MIB.my
- **DHCPv4 server capability**—CISCO-IETF-DHCP-SERVER-CAPABILITY.my
- **DHCPv4 server extensions**—CISCO-IETF-DHCP-SERVER-EXT-MIB.my
- **DHCPv4 server extensions capability**—CISCO-IETF-DHCP-SERVER-EXT-CAPABILITY.my
- **DHCPv6 server**—CISCO-NETREG-DHCPV6-MIB.my (experimental)
- **Dependency for DHCPv4 and DHCPv6**—CISCO-SMI.my
- **Additional dependencies for DHCPv6**—INET-ADDRESS-MIB.my

The trap, server, and DHCPv4 extension and capability MIB files are available in the **misc** directory of the Network Registrar installation path. The remaining MIB files (except the experimental CISCO-NETREG-DHCPV6-MIB) are available at:

ftp://ftp.cisco.com/pub/mibs/v2/

To get the object identifiers (OIDs) for the MIB attributes, go to the equivalently named .oid file at:

ftp://ftp.cisco.com/pub/mibs/oid/

See Also

Setting Up the SNMP Server
How Notification Works, page 1-6
Handling SNMP Notification Events, page 1-7
Handling SNMP Queries, page 1-10

Setting Up the SNMP Server

To perform queries to the SNMP server, you need to set up the server properties.

Local Basic or Advanced Web UI

Step 1 Click **Servers**, then **Manage Servers** to open the Manage Servers page (see the "Managing Servers" section on page 7-1).

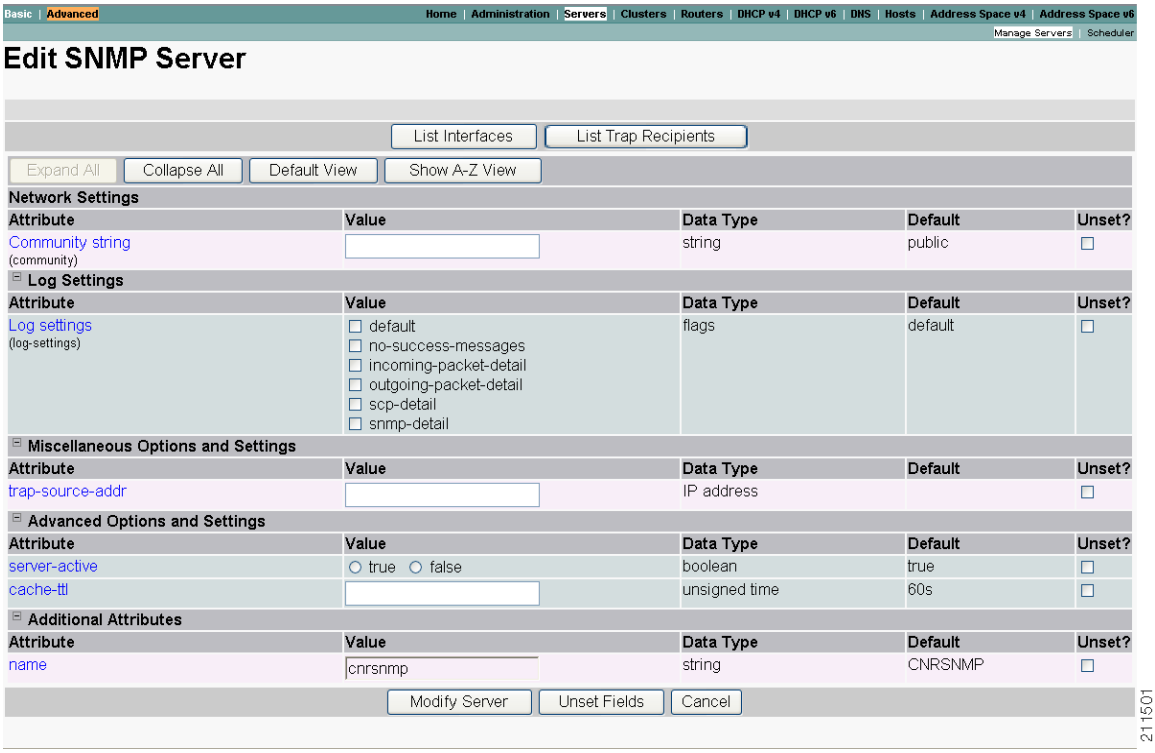
Step 2 Click the Local SNMP Server link to open the Edit SNMP Server page (see Figure 1-1 on page 1-5, which shows all attributes as expanded by having clicked **Expand All**).

Step 3 The *Community string* attribute is the password to access the server. (The community string is a read community string only.) The preset value is **public**.

Step 4 You can specify log settings and other miscellaneous and advanced options:

- ***trap-source-addr***—Optional sender address to use for outgoing traps.
- ***server-active***—Determines if the SNMP server is active for queries; the preset value is true. If set to false, the server will run, but is not accessible for queries and does not send out traps.
- ***cache-ttl***—Determines how long the SNMP caches responses to queries, preset to 60 seconds.

Figure 1-1 Edit SNMP Server Page (Local Advanced)



Step 5 To manage the SNMP server interfaces, click **List Interfaces** (or click the icon in the Interfaces column of the SNMP server on the Manage Servers page). You can view the default configured network interfaces, and create and edit additional ones. To create and edit them, you must be assigned the server-management subrole of the ccm-admin role. The interface properties are similar to those for the TFTP server (see the "Managing the TFTP Server Network Interfaces" section on page 1-3).

Step 6 To manage trap recipients for the server:

a. Click **List Trap Recipients** to open the List/Add Trap Recipients page.

b. Enter the name and IP address of a trap recipient (both are required).

c. Click **Add Trap Recipient**.

d. Repeat for each additional trap recipient.

e. To set the port, community string, and agent address for a trap recipient, click its name on the List/Add Trap Recipients page to open the Edit Trap Recipient page, then set the values.

f. Click **Modify Recipient**.

Step 7 Complete the SNMP server setup by clicking **Modify Server**.

CLI Commands

To set the community string in the CLI so that you can access the SNMP server, use **snmp set community=***name*. Use **snmp set trap-source-addr** to set the trap source address. Use **snmp disable server-active** to deactivate the SNMP server, and **snmp set cache-ttl=***time* to set the cache time-to-live.

To set trap recipients, use **trap-recipient**, in the following syntax to include the IP address:

```
nrcmd> trap-recipient name create ip-addr=ip-addr
```

You can also add the *agent-address*, *community*, and *port-number* values for the trap recipient.

Other SNMP-related commands include **snmp disable server-active** to prevent the server from running when started, and the **snmp-interface** commands to configure the interfaces. The **addr-trap** command is described in the "Handling SNMP Notification Events" section on page 1-7.

How Notification Works

Network Registrar SNMP notification support allows a standard SNMP management station to receive notification messages from the DHCP and DNS servers. These messages contain the details of the event that triggered the SNMP trap.

Network Registrar generates notifications in response to predetermined events that the application code detects and signals. Each event can also carry with it a particular set of parameters or current values. For example, the *free-address-low-threshold* event can occur in the scope with a value of 10% free. Other scopes and values are also possible for such an event, and each type of event can have different associated parameters.

Table 1-1 describes the events that can generate notifications.

Table 1-1 SNMP Notification Events

Event	Notification
Address conflict with another DHCP server detected (*address-conflict*)	An address conflicts with another DHCP server.
DNS queue becomes full (*dns-queue-size*)	The DHCP server DNS queue fills and the DHCP server stops processing requests. (This is usually a rare internal condition.)
Duplicate IP address detected (*duplicate-address* and *duplicate-address6*)	A duplicate IPv4 or IPv6 address occurs.
Duplicate IPv6 prefix detected (*duplicate-prefix6*)	A duplicate IPv6 prefix occurs.
Failover configuration mismatch (*failover-config-error*)	A DHCP failover configuration does not match between partners.
DNS forwarders not responding (*forwarders-not-responding*)	Forwarding servers stop responding to the DNS server.
DNS forwarders responding (*forwarders-responding*)	Forwarding servers respond after having been unresponsive.
Free-address thresholds (*free-address-low* and *free-address-high*; or *free-address6-low* and *free-address6-high*)	The high trap when the number of free IPv4 or IPv6 addresses exceeds the high threshold; or a low trap when the number of free addresses falls below the low threshold after previously triggering the high trap.

Table 1-1 SNMP Notification Events (continued)

Event	Notification
High-availability (HA) DNS configuration mismatch (*ha-dns-config-error*)	An HA DNS configuration does not match between partners.
HA DNS partner not responding (*ha-dns-partner-down*)	An HA DNS partner stops responding to the DNS server.
HA DNS partner responding (*ha-dns-partner-up*)	An HA DNS partner responds after having been unresponsive.
DNS masters not responding (*masters-not-responding*)	Master DNS servers stop responding to the DNS server.
DNS masters responding (*masters-responding*)	Master DNS servers respond after having been unresponsive.
Other server not responding (*other-server-down*)	A DHCP failover partner, or a DNS or LDAP server, stops responding to the DHCP server.
Other server responding (*other-server-up*)	DHCP failover partner, or a DNS or LDAP server, responds after having been unresponsive.
DNS secondary zones expire (*secondary-zone-expired*)	A DNS secondary server can no longer claim authority for zone data when responding to queries during a zone transfer.
Server start (*server-start*)	The DHCP or DNS server is started or reinitialized.
Server stop (*server-stop*)	The DHCP or DNS server is stopped.

Handling SNMP Notification Events

When Network Registrar generates a notification, it transmits a single copy of the notification as an SNMP Trap PDU to each recipient. All events (and scopes or prefixes) share the list of recipients and other notification configuration data, and the server reads them when you initialize the notification.

You can set SNMP attributes in three ways:

- For the DHCP server, which includes the traps to enable and the default free-address trap configuration if you are not specifically configuring traps for scopes or prefixes (or their templates).
- On the scope or prefix (or its template) level by setting the *free-address-config* attribute.
- For the DNS server, which includes a *traps-enabled* setting.

To use SNMP notifications, you must specify trap recipients that indicate where trap notifications should go. By default, all notifications are enabled, but you must explicitly define the recipients, otherwise no notifications can go out. The IP address you use is often **localhost**.

The DHCP server provides special trap configurations so that it can send notifications, especially about free addresses for DHCPv4 and DHCPv6. You can set the trap configuration name, mode, and percentages for the low threshold and high threshold. The mode determines how scopes aggregate their free-address levels.

DHCPv4 Notification

The DHCPv4 modes and thresholds are (see also the "Handling Deactivated Scopes or Prefixes" section):

- **scope mode**—Causes each scope to track its own free-address level independently (the default).
- **network mode**—Causes all scopes set with this trap configuration (through the scope or scope template *free-address-config* attribute) to aggregate their free-address levels if the scopes share the same *primary-subnet*.
- **selection-tags mode**—Causes scopes to aggregate their free-address levels if they share a primary subnet and have a matching list of selection tag values.
- **low-threshold**—Free-address percentage at which the DHCP server generates a low-threshold trap and re-enables the high threshold. The free-address level for scopes is the following calculation:

  ```
  100 * available-nonreserved-leases
      total-configured-leases
  ```

- **high-threshold**—Free-address percentage at which the DHCP server generates a high-threshold trap and re-enables the low threshold.

DHCPv6 Notification

The DHCPv6 modes and thresholds are (see also the "Handling Deactivated Scopes or Prefixes" section):

- **prefix mode**—Causes each prefix to track its own free-address level independently.
- **link mode**—Causes all prefixes configured for the link to aggregate their own free-address levels if all prefixes share the same link.
- **v6-selection-tags mode**—Causes prefixes to aggregate their free-address levels if they share a link and have a matching list of selection tag values.
- **low-threshold**—Free-address percentage at which the DHCP server generates a low-threshold trap and re-enables the high threshold. The free-address level for prefixes is the following calculation:

  ```
  100 * max-leases - dynamic-leases
      max-leases
  ```

- **high-threshold**—Free-address percentage at which the DHCP server generates a high-threshold trap and re-enables the low threshold.

Handling Deactivated Scopes or Prefixes

A deactivated scope or prefix never aggregates its counters with other scopes or prefixes. For example, if you configure a prefix with **link** or **v6-selection-tags** trap mode, and then deactivate the prefix, its counters disappear from the total count on the aggregation. Any changes to the leases on the deactivated prefix do not apply to the aggregate totals.

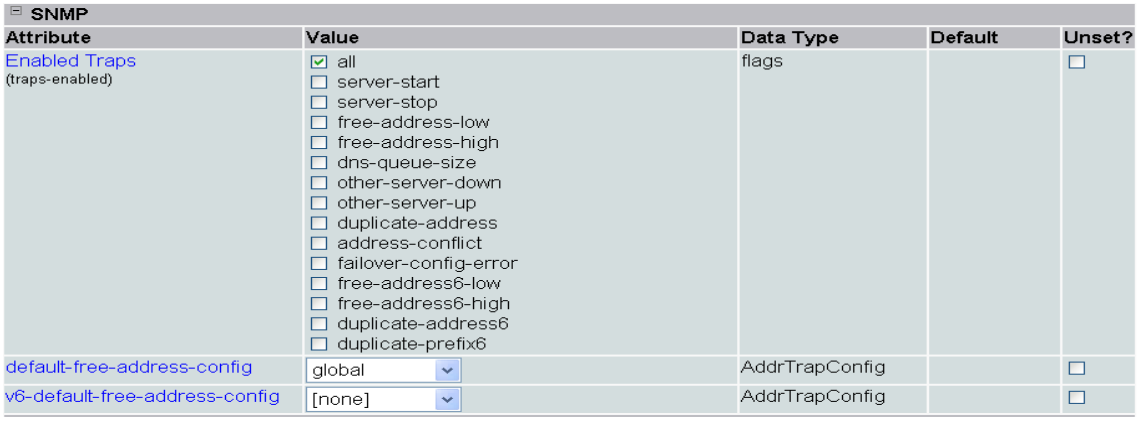
Therefore, to detect clients for deactivated scopes or prefixes, you must set the event mode to **scope** or **prefix**, and not to any of the aggregate modes (**network**, **selection-tags**, **link**, or **v6-selection-tags**).

The use case for setting traps on deactivated prefixes, for example, is network renumbering. In this case, you might want to monitor both the new prefixes (as an aggregate, ensuring that you have enough space for all the clients) and old prefixes to ensure that their leases are freed up. You would probably also want to set the high threshold on an old prefix to 90% or 95%, so that you get a trap fired when most of its addresses are free.

Local Basic or Advanced Web UI

Access the SNMP attributes for the DHCP server by clicking **Servers**, then the Local DHCP Server link on the Manage Servers page. The SNMP attributes are about two-thirds down the Edit DHCP Server page (see Figure 1-2).

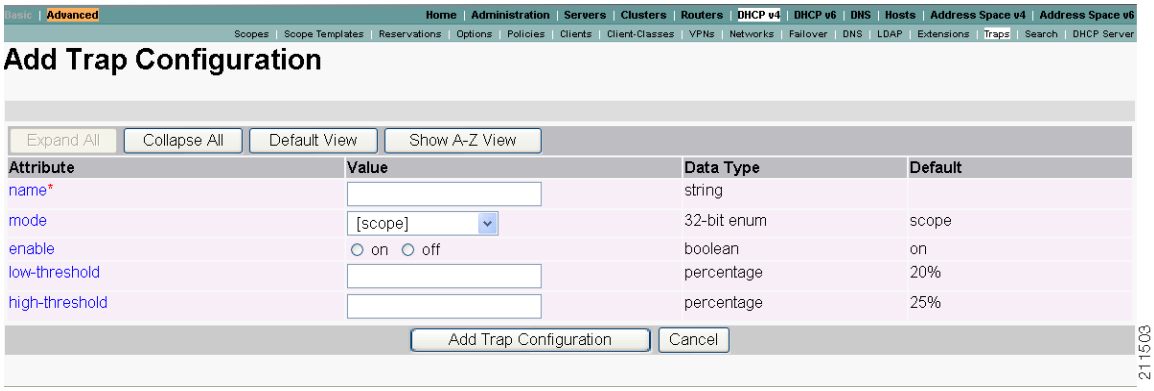
Figure 1-2 SNMP Attributes on the Edit DHCP Server Page (Local)



The last four *lease-enabled* values pertain to DHCPv6 only. Along with the traps to enable, you can specify the default free-address trap configuration by name, which affects all scopes and prefixes or links not explicitly configured.

Access the DHCP trap configurations in Advanced mode if you click **DHCP**, then **Traps** to open the List Trap Configurations page. On this page, click **Add Trap Configuration** to open the Add Trap Configuration page (see Figure 1-3).

Figure 1-3 Add Trap Configuration Page for DHCPv4 (Local Advanced)



Enter the name, mode, and threshold percentages. The trap configuration is preset to *enable*=on. After making these settings, click **Add Trap Configuration**. You can edit the configuration.

Regional Web UI

In the regional web UI, you can add and edit trap configurations as in the local web UI. You can also pull replica trap configurations and push trap configurations to the local cluster on the List Trap Configurations page.

CLI Commands

To set the trap values for the DHCP server at the local cluster, use **dhcp set traps-enabled=***value*. You can also set the *default-free-address-config* attribute to the trap configuration. For example:

```
nrcmd> dhcp set traps-enabled=server-start,server-stop,free-address-low,free-address-high
nrcmd> dhcp set default-free-address-config=v4-trap-config
```

To define trap configurations for DHCPv4 and DHCPv6, use **addr-trap** *name* **create** followed by the *attribute*=*value* pairs for the settings. For example:

```
nrcmd> addr-trap v4-trap-conf create mode=scope low-threshold=25% high-threshold=30%
nrcmd> addr-trap v6-trap-conf create mode=prefix low-threshold=20% high-threshold=25%
```

Handling SNMP Queries

You can use SNMP client applications to query the following MIBs:

- CISCO-DNS-SERVER-MIB.my
- CISCO-IETF-DHCP-SERVER-MIB.my
- CISCO-IETF-DHCP-SERVER-EXT-MIB.my
- CISCO-NETREG-DHCPV6-MIB.my (experimental)

When the SNMP server receives a query for an attribute defined in one of these MIBs, it returns a response PDU containing that attribute value. For example, using the NET-SNMP client application (available over the Internet), you can use one of these commands to obtain a count of the DHCPDISCOVER packets for a certain address:

```
C:\net-snmp5.2.2\bin>snmpget -m ALL -v 2c -c public
    192.168.241.39:4444.iso.org.dod.internet.private.enterprises.cisco.ciscoExperiment.
    ciscoIetfDhcpSrvMIB.ciscoIetfDhcpv4SrvMIBObjects.cDhcpv4Counters.cDhcpv4CountDiscovers
CISCO-IETF-DHCP-SERVER-MIB::cDhcpv4CountDiscovers = Counter32: 0

C:\net-snmp5.2.2\bin>snmpget -m ALL -v 2c -c public
    192.168.241.39:4444 1.3.6.1.4.1.9.10.102.1.3.1
CISCO-IETF-DHCP-SERVER-MIB::cDhcpv4CountDiscovers = Counter32: 0
```

Both commands return the same results. The first one queries the full MIB attribute name, while the second one queries its OID equivalent (which can be less error prone). As previously described, the OID equivalents of the MIB attributes are located in the relevant files at the following URL:

ftp://ftp.cisco.com/pub/mibs/oid/

For example, the CISCO-IETF-DHCP-SERVER-MIB.oid file includes the following OID definition that corresponds to the previous query example:

```
"cDhcpv4CountDiscovers" "1.3.6.1.4.1.9.10.102.1.3.1"
```

Here are some possible SNMP query error conditions:

- The community string sent in the request PDU does not match what you configured.
- The version in the request PDU is not the same as the supported version (SNMPv2).
- If the object being queried does not have an instance in the server, the corresponding variable binding type field is set to SNMP_NOSUCHINSTANCE. With a GetNext, if there is no next attribute, the corresponding variable binding type field is set to SNMP_ENDOFMIBVIEW.

- If no match occurs for the OID, the corresponding variable binding type field is set to SNMP_NOSUCHOBJECT. With a GetNext, it is set to SNMP_ENDOFMIBVIEW.
- If there is a bad value returned by querying the attribute, the error status in the response PDU is set to SNMP_ERR_BAD_VALUE.

Default Ports for Network Registrar Services

Table 1-2 lists the default ports used for the Network Registrar services.

Table 1-2 Default Ports for Network Registrar Services

Port Number	Protocol	Service
22	TCP	SSH remote login (RIC server to router)
23	TCP	Telnet (RIC server to router)
53	TCP/UDP	DNS
67	UDP	DHCP client to server
68	UDP	DHCP server to client
69	UDP	TFTP (optional) client to server
162	TCP	SNMP traps server to server
389	TCP	DHCP server to LDAP server
546	UDP	DHCPv6 server to client
547	UDP	DHCPv6 client to server
647	UDP	DHCP failover server to server
653	TCP	High-Availability (HA) DNS server to server
1234	TCP	Local cluster CCM server to server
1244	TCP	Regional cluster CCM server to server
4444	TCP	SNMP client to server
8080	HTTP	Local cluster client to CCM server
8090	HTTP	Regional cluster client to CCM server
8443	HTTPS	Local cluster secure client to CCM server
8453	HTTPS	Regional cluster secure client to CCM server



CHAPTER 2

Network Registrar User Interfaces

Cisco Network Registrar provides a regional and a local web-based user interface (web UI) and a regional and local command line interface (CLI) to manage the DNS, DHCP, TFTP, and Central Configuration Management (CCM) servers:

- **Web UI for the regional cluster to access local cluster servers**—See the "Regional Cluster Web UI" section on page 2-11).
- **Web UI for the local cluster**—See the "Local Cluster Web UI" section on page 2-7).
- **CLI for the local clusters**—Open the **CLIContent.html** file in the installation **/docs** directory (see the "Command Line Interface" section on page 2-12).
- **CCM servers that provide the infrastructure to support these interfaces**—See the "Central Configuration Management Server" section on page 2-13).

This chapter describes the Network Registrar user interfaces and the services that the CCM servers provide. Read this chapter before starting to configure the Network Registrar servers so that you become familiar with each user interface capability.

Introduction to the Web-Based User Interfaces

The web UI provides granular access to configuration data through user roles and constraints. The web UI granularity is described in the following sections.

See Also

Supported Web Browsers, page 2-2
Access Security, page 2-2
Logging In to the Web UIs, page 2-2
Changing Passwords, page 2-3
Navigating the Web UIs, page 2-4
Waiting for Page Resolution Before Proceeding, page 2-4
Committing Changes in the Web UIs, page 2-4
Role and Attribute Visibility Settings, page 2-5
Displaying and Modifying Attributes, page 2-5
Help Pages, page 2-7
Logging Out, page 2-7

Supported Web Browsers

The minimum web browsers supported in Network Registrar are Internet Explorer 5.5 and Netscape 6.2.

Access Security

At Network Registrar installation, you can choose to configure HTTPS to support secure client access to the web UIs. You must specify the HTTPS port number and provide the keystore at that time. With HTTPS security in effect, the web UI Login page (see Figure 2-1 on page 2-2) indicates that the "Page is SSL Secure."

Note Do not use a dollar sign ($) symbol as part of a keystore password.

Logging In to the Web UIs

You can log in to the Network Registrar local or regional cluster web UIs either by HTTPS secure or HTTP nonsecure login. After installing Network Registrar, open one of the supported web browsers and specify the login location URL in the browser address or netsite field. Login is convenient and provides some memory features to increase login speed.

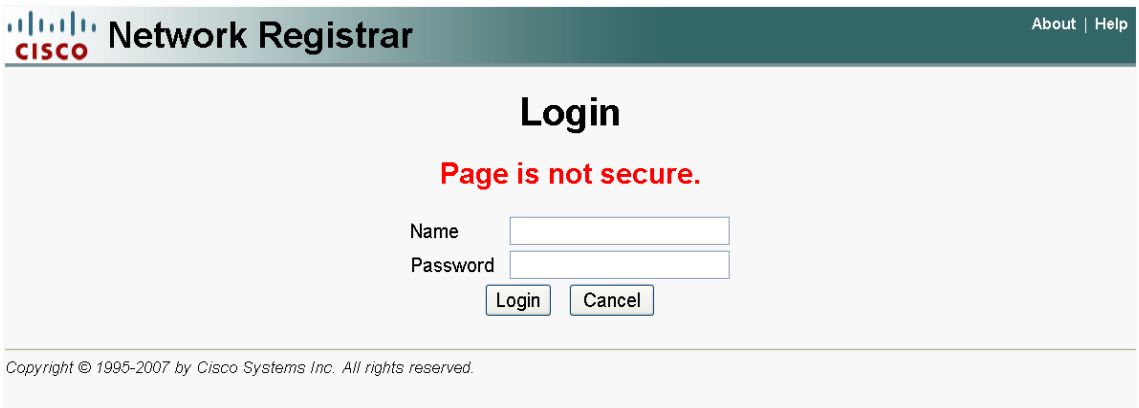
You can log in using a nonsecure login in two ways:

- On Windows, from the Start menu, choose **Start > Programs > Network Registrar 7.0 > Network Registrar 7.0 {local | regional} Web UI**, which opens the local or regional cluster web UI from your default web browser.
- Open the web browser and go to the web site. For example, if default ports were used during the installation, the URLs would be **http://***hostname***:8080** for the local cluster web UI, and **http://***hostname***:8090** for the regional cluster web UI.

This opens the Login page. With a conventional login, the page indicates "Page is not secure" (see Figure 2-1); with an SSL-secured login, the page indicates "Page is SSL Secure."

Note To prepare for an HTTPS-secured login, see the *Installation Guide*.

Figure 2-1 Web UI Login Page

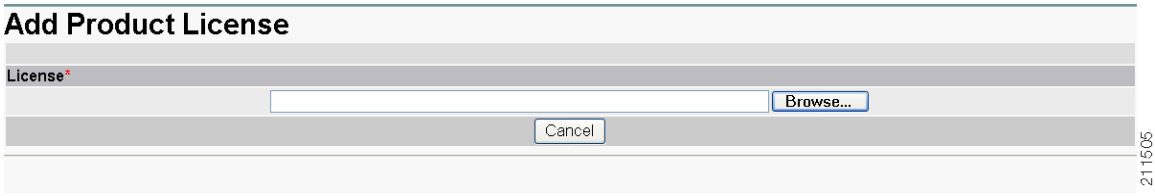


Enter the initial username **admin** and password **changeme**. The password is case sensitive. (Change this password as soon as you can; see the "Managing Passwords" section on page 5-7). Depending on how your browser is set up, you might be able to abbreviate the account name or choose it from a drop-down list. If the password is stored from a previous login, it might be entered automatically.

To log in, click **Login**.

If this is the first login after installing Network Registrar and you did not specify a license during installation, the Add Product License page appears. (See Figure 2-2 for the local cluster page; the appearance of the regional cluster page is essentially the same.)

Figure 2-2 Add Product License Page (Local)



Cisco will e-mail you one or more license files once you register the Network Registrar Product Authorization Key (PAK) on the web according to the Software License Claim Certificate shipped with the product. Cisco administers licenses through a FLEXlm system. Once you have the file or files:

1. Locate the license file or files in a directory (or on the desktop) that is easy to find.
2. On the Add Product License page, browse for each file by clicking the **Browse** button.
3. In the Choose file window, find the location of the initial license file, then click **Open**.
4. If the license key is acceptable, the Main Menu page appears immediately.
5. To add further licenses, click **Administration**, then **Licenses** to open the List/Add Product License page. Click **Browse** to open the Choose file window, locate the additional license file, then click **Open**. If the key in the file is acceptable, the key, type, count, and expiration date appear, along with whether it is an evaluation key. If the key is not acceptable, the page shows the license text along with an error message. For the list of license types, see the "Licensing" section on page 5-8.

   Under the table of licenses is a License Utilization area that, when expanded, shows the license types along with the total nodes that you can use and those actually used.

To reenter a previously active session, click **Reuse current session** (assuming that you did not remove cookies in the web browser).

Tip If you log back in to a previously active session without clicking **Reuse current session**, you could have two active sessions open, which can cause failures. For example, if your active session was the first one after an installation, when you enter the license key, you are prompted for it again indefinitely. To avoid this, click **Reuse current session**, or close and reopen the browser to initiate a new session.

Changing Passwords

Whenever you edit a password on a web UI page, it is displayed as a string of eight asterisks (********). The actual password value is never sent to the web browser. So, if you change the password, the field is automatically cleared. You must enter the new password value completely, exactly as you want it to be.

For details on changing administrator passwords at the local and regional cluster, see the "Managing Passwords" section on page 5-7.

Navigating the Web UIs

The web UI provides a hierarchy of pages based on the functionality you desire and the thread you are following as part of your administration tasks. The page hierarchy prevents you from getting lost easily.

Caution Do not use the Back button of the browser. Always use the navigation bar menu, or the **Cancel** button on the page to return to a previous page. Using the browser Back button can cause erratic failures.

A single sign-on feature is available to connect between the regional and local cluster web UIs. Many of the regional cluster web UI pages include the Go Local icon (), which you can click to connect to the local cluster associated with the icon. If you have single sign-on privileges to the local cluster, the connection takes you to the related local server management page (or a related page for related server configurations). If you do not have these privileges, the connection takes you to the login page for the local cluster. To return to the regional cluster, local cluster pages have the Go Regional icon () at the top right corner of the page.

Note Navigation bar items can vary based on if you have the role privileges for IPv4 or IPv6. For example, the **Address Space** menu bar can be **Address Space v4** and **Address Space v6** if you have the ipv6-management subrole of the addrblock-admin role assigned. In this *User Guide*, the convention is to use only the generic label of the menu bar in instructions. Similarly, **DHCP** can mean the **DHCP**, **DHCPv4**, or **DHCPv6** tabs.

Waiting for Page Resolution Before Proceeding

Operations performed in the web UI, such as resynchronizing or replicating data from server clusters, are synchronous in that they do not return control to the browser until the operation is completed. These operations display confirmation messages in blue text. Also, both the Netscape and IE browsers display a wait cursor while the operation is in progress.

Tip Wait for each operation in the web UI to finish before you begin a new operation. If the browser becomes impaired, close the browser, reopen it, then log in again.

Committing Changes in the Web UIs

You do not actually commit the page entries you make until you click **Add**... or **Modify**... on the page. You can delete items using the Delete icon (). To prevent unwanted deletions, a Confirm Delete page appears in many cases so that you have a chance to confirm or cancel the deletion.

Role and Attribute Visibility Settings

In Advanced user mode (see the "Local Advanced Main Menu Page" section on page 2-9), the Main Menu page shows the administrative roles assigned to the logged-in administrator. It also presents a choice of which visibility you want the configuration attributes to be in the web UI:

- To view the user groups and roles for the administrator, click the plus sign (+) next to the User Role and Group Data heading. Superuser is a special kind of administrator. (For details how to set up these administrator roles, see the "Create the Administrators" section on page 5-16.)
- To set the attribute visibility settings for this user session only, click the plus sign (+) next to the Session Settings heading (see Figure 2-6 on page 2-9). Pull down the choices next to the **Set Web UI Mode** button, choose a mode, then click **Set Web UI Mode**:
  - **Basic**—Basic user mode (the preset choice).
  - **Advanced**—Advanced user mode that exposes the normal attributes.
  - **Expert**—Expert user mode that exposes a set of attributes that are relevant for fine-tuning or troubleshooting the configuration. In most cases, you would accept the default values for these expert attributes and not change them without guidance from the Cisco Technical Assistance Center (TAC). Each Expert mode attribute is marked with a Warning icon () on the configuration pages. Each page is clearly marked as being in Expert mode.

    When you leave Expert mode (click **Basic** or **Advanced**), the **Expert** button disappears. To reenter Expert mode, you must set the web UI mode to Expert again.

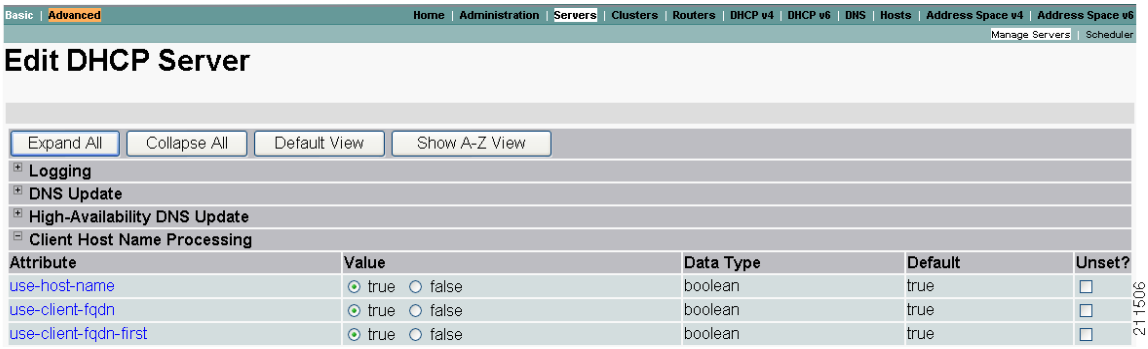
Displaying and Modifying Attributes

Many of the web UI pages, such as those for servers, zones, and scopes, include attribute settings that correspond to those you can set using the CLI. (The CLI name equivalents appear under the attribute name.) The attributes are categorized into groups by their function, with the more prominent attributes listed first and the ones less often configured nearer the bottom of the page.

Grouping and Sorting Attributes

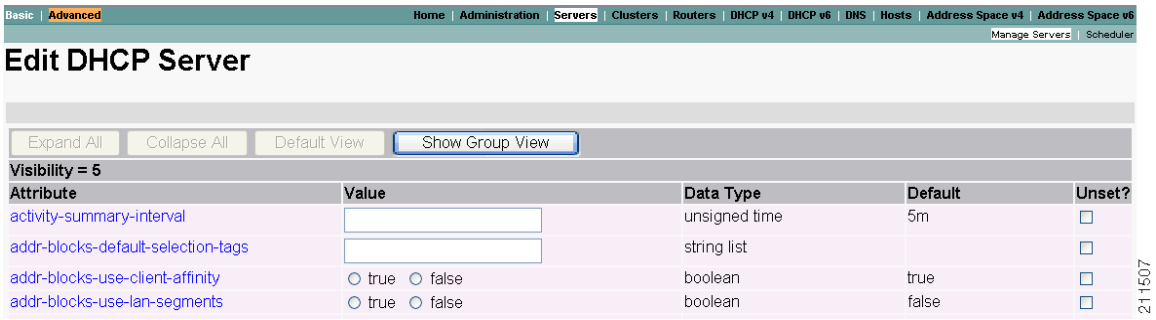
On many Advanced mode web UI pages, you can toggle between showing attributes in groups and in alphabetical order. These pages generally open by default in group view so that you can see the attributes in their respective categories. However, in the case of large numbers of attributes, you might want to see the attributes alphabetized. Click **Show A-Z View** to change the page to show the attributes alphabetically. Click **Show Group View** to change the page to show the attributes in groups. Figure 2-3 on page 2-6 and Figure 2-4 on page 2-6 show the difference between group and A-Z views for a sample web UI page.

Figure 2-3 Attributes in Group View (Local Advanced)



You can also expand or collapse the attribute groups in group view by clicking **Expand All** or **Collapse All**. Figure 2-4 on page 2-6 shows the alphabetized view with the example of being in Expert mode.

Figure 2-4 Attributes in Alphabetical View (Local Advanced)



Tip In Expert mode, the Expert mode attributes are alphabetized separately further down the page under the Visibility=3 heading and are all marked with the Warning icon ( ).

Modifying Attributes

You can modify attribute values and unset those for optional attributes. In many cases, these attributes have preset values, which are listed under the Default column on the page. The explicit value overrides the default one, but the default one is always the fallback. If there is no default value, unsetting the explicit value removes all values for that attribute.

Displaying Attribute Help

For contextual help for an attribute, click the name of the attribute to open a separate popup window.

Help Pages

The web UI provides a separate window that displays help text for each page. The Help pages provide:

- A context-sensitive help topic depending on which application page you have open.
- A clickable and hierarchical Contents and Index, and a Favorites setting, as tabs on a left-hand pane that you can show or hide.
- A Search facility that returns a list of topics containing the search string, ordered by frequency of appearance of the search string.
- Forward and backward navigation through the history of Help pages opened.
- A Print function.
- A Glossary.

Logging Out

Log out of the web UI by clicking **Logout** in the top right corner of any application page.

Local Cluster Web UI

The local cluster web UI provides concurrent access to Network Registrar user and protocol server administration and configuration. It provides granular administration across servers with permissions you can set on a per element or feature basis. The local cluster web UI is available in three user modes:

- **Basic mode**—Provides a more simplified configuration for the more frequently configured objects, such as DHCP scopes and DNS zones (see the "Local Basic Main Menu Page" section on page 2-8).
- **Advanced mode**—Provides the more advanced configuration method familiar to past users of the Network Registrar web UI, with some enhancements (see the "Local Advanced Main Menu Page" section on page 2-9).
- **Expert mode** (marked with the ⚠ icon)—For details on Expert mode, see the "Role and Attribute Visibility Settings" section on page 2-5.

Change to Basic or Advanced mode by clicking **Basic** or **Advanced** at the top left of the page. You can also add Expert mode by setting User Preferences (see the "Setting Local User Preferences" section on page 2-10.

Note If you change the IP address of your local cluster machine, see the Note in the "Configuring Clusters in the Local Web UI" section on page 2-11.

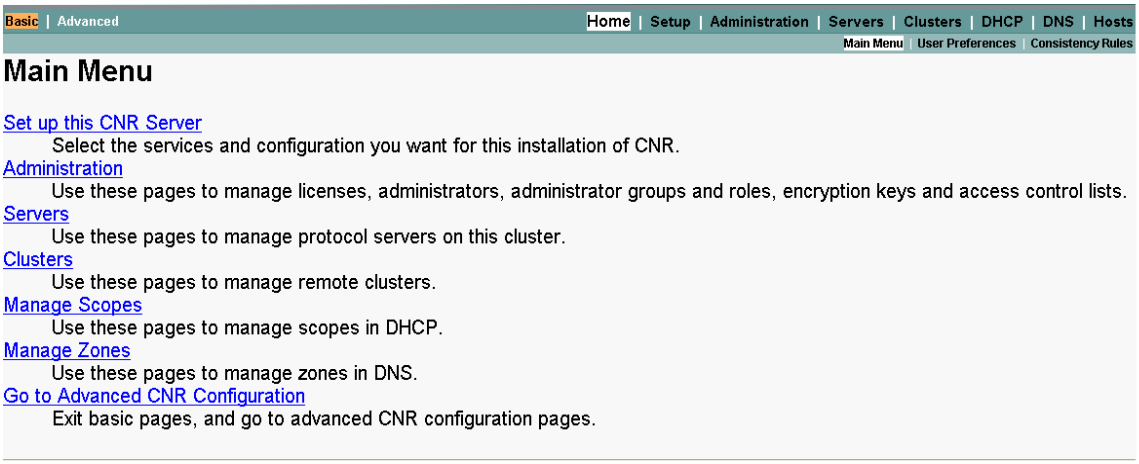
See Also

Introduction to the Web-Based User Interfaces, page 2-1
Regional Cluster Web UI, page 2-11

Local Basic Main Menu Page

Once you log in to the application, the Main Menu page is preset to open in Basic user mode (see Figure 2-5). You can tell that you are in Basic mode because the Basic tab is activated at the top left corner of the page. Otherwise, activate Basic user mode by clicking **Basic**.

Figure 2-5 Local Basic Main Menu Page



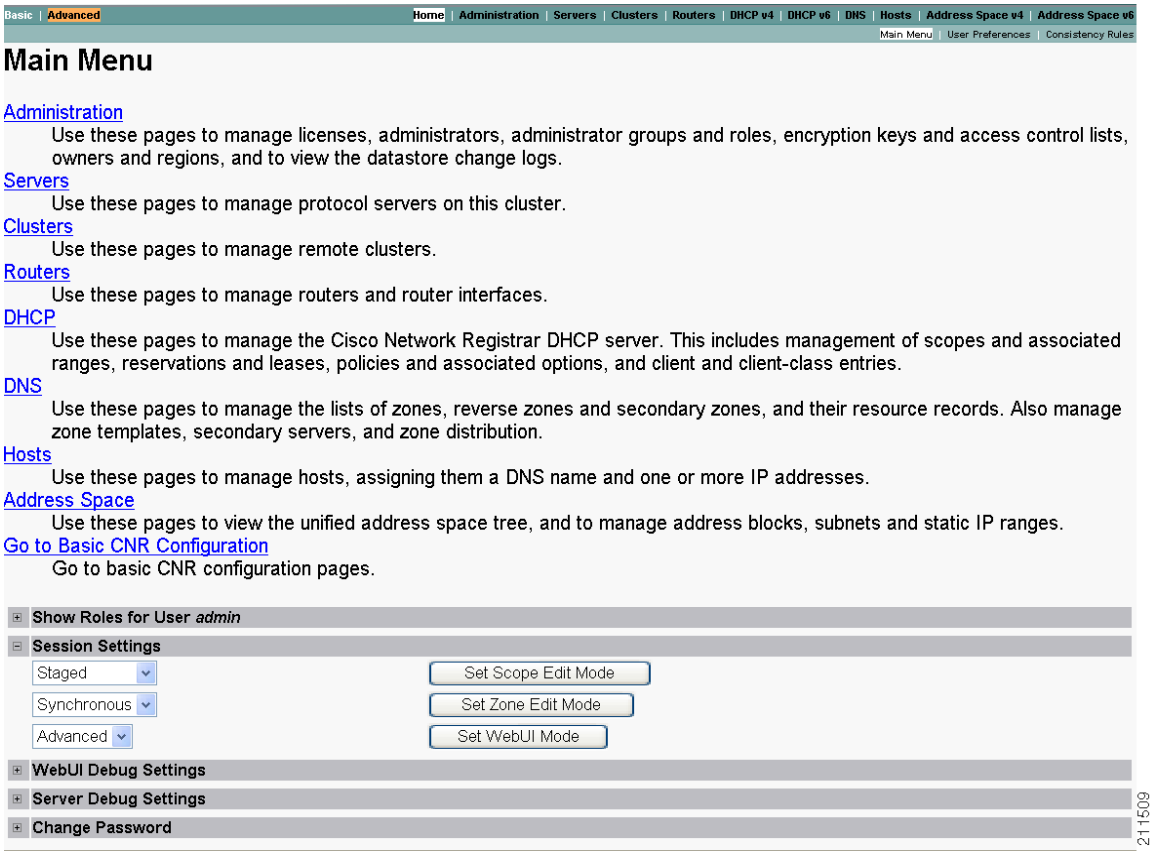
This page provides functions with which you can:

- **Open the dashboard to monitor system health**—Click **Dashboard** in the top right part of the page. See Chapter 3, "Server Status Dashboard."
- **Set up a basic configuration by using the Setup interview pages**—Click the **Setup** menu or Set up this CNR Server link on the page. See the *Quick Start Guide for Cisco Network Registrar*.
- **Administer licenses, users, groups, roles, encryption keys, and access control lists (ACLs)**—Click the **Administration** menu or link on the page. See Chapter 5, "Configuring Administrators."
- **Manage the Network Registrar protocol servers**—Click the **Servers** menu or link on the page. See Chapter 7, "Maintaining Servers and Databases."
- **Manage clusters**—Click the **Clusters** menu or link on the page. See the "Configuring Server Clusters" section on page 6-2.
- **Configure DHCP**—Click the **DHCP** menu or Manage Scopes link on the page. See Chapter 20, "Configuring Scopes and Networks."
- **Configure DNS**—Click the **DNS** menu or Manage Zones link on the page. See Chapter 15, "Managing Zones."
- **Manage hosts in zones**—Click the **Hosts** menu. See Chapter 10, "Managing Hosts."
- **Go to Advanced mode**—Click **Advanced** in the upper left part of the page or Go to Advanced CNR Configuration link on the page. See the "Local Advanced Main Menu Page" section.

Local Advanced Main Menu Page

You can switch to Advanced user mode on the Basic user Main Menu page (see Figure 2-5 on page 2-8) by clicking **Advanced** in the top left corner of the page. Doing so opens another Main Menu page, except that it shows the Advanced user mode functions (see Figure 2-6 for a view that shows the Session Settings). You can toggle back to Basic mode at any time by clicking **Basic** in the top left corner of the page.

Figure 2-6 Local Advanced Main Menu Page



The local Advanced mode Main Menu page includes advanced Network Registrar functions that are in addition to the ones in Basic mode:

- **Open the dashboard to monitor system health**—Click **Dashboard** in the top right part of the page. See Chapter 3, "Server Status Dashboard."
- **Administer licenses, users, groups, roles, encryption keys, owners, regions, and ACLs, plus view change logs and CCM and MCD tasks**—Click the **Administration** menu or link on the page. See Chapter 5, "Configuring Administrators."
- **Manage the Network Registrar protocol servers**—Click the **Servers** menu or link on the page. See Chapter 7, "Maintaining Servers and Databases."
- **Manage clusters**—Click the **Clusters** menu or link on the page. See the "Configuring Server Clusters" section on page 6-2.

- **Configure DHCPv4**—Click the **DHCPv4** or **DHCP** menu. (Clicking the DHCP link on the page opens the DHCP v4 view.) See Chapter 20, "Configuring Scopes and Networks."
- **Configure DHCPv6**—Click the **DHCPv6** menu. See Chapter 26, "Managing DHCPv6 Addresses."
- **Configure DNS**—Click the **DNS** menu or link on the page. See Chapter 15, "Managing Zones."
- **Manage hosts in zones**—Click the **Hosts** menu or link on the page. See Chapter 10, "Managing Hosts."
- **Manage IPv4 address space**—Click the **Address Space v4** or **Address Space** menu. (Clicking the Address Space link on the page opens the Address Space v4 view.) See Chapter 9, "Managing Address Space."
- **Configure IPv6 address space**—Click the **Address Space v6** menu. See Chapter 26, "Managing DHCPv6 Addresses."
- **Go to Basic mode**—Click **Basic** in the top left corner of the page or the Go to Basic CNR Configuration link on the page. See the "Local Basic Main Menu Page" section on page 2-8.

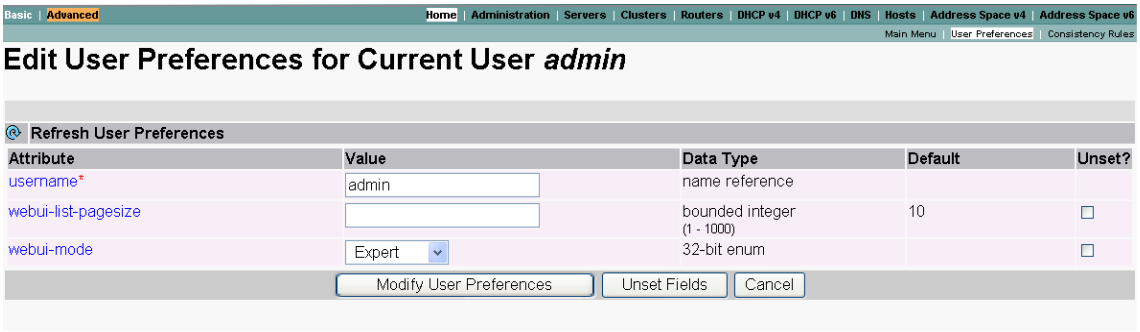
The Advanced user mode page provides additional functions:

- **View the user role and group data for the logged-in user**—See the "Role and Attribute Visibility Settings" section on page 2-5.
- **Set your preferred session settings**—See the "Role and Attribute Visibility Settings" section on page 2-5.
- **Set server debugging**—You can set debug flags for the protocol servers. Set these values only under diagnostic conditions when communicating with the Cisco Technical Assistance Center (TAC).
- **Change your login administrator password**—See the "Changing Passwords" section on page 2-3.

Setting Local User Preferences

As of Network Registrar 7.0, you can persist a short list of web UI settings through subsequent user sessions. In either Basic or Advanced user mode, click **Home**, then **User Preferences** to open the Edit User Preferences for Current User page (see Figure 2-7). The only difference between the Basic and Advanced or Expert mode user preference pages is that Advanced and Expert modes have additional columns listing the data types and defaults.

Figure 2-7 Edit User Preferences Page (Local Advanced)



The user preference attributes to set are:

- **Username**—Username string, with a preset value of admin. You cannot modify this field.
- **Web UI list page size**—Adjust the page size by the number of displayed lines in a list; the preset value is 10 lines.
- **Web UI mode**—User mode at startup: Basic, Advanced, or Expert (see the "Role and Attribute Visibility Settings" section on page 2-5). If unset, the mode defaults to the one set in the CCM server configuration (see the "Managing Servers" section on page 7-1).

You can unset the page size and web UI mode values by clicking a check mark in Unset? next to the attribute, then clicking **Unset Fields**. After making the user preference settings, click **Modify User Preferences**.

Configuring Clusters in the Local Web UI

You can define other local Network Registrar clusters in the local web UI. The local cluster on the current machine is called the **localhost** cluster. To set up other clusters, click **Clusters** to open the List Clusters page. Note that the **localhost** cluster has the IP address and SCP port of the local machine.

Click **Add Cluster** to open the Add Cluster page. At a minimum, you must enter the name and ipaddr of the remote local cluster. You should also enter the admin name and password, along with possibly the SCP port (if not 1234), of the remote cluster. If you want to use secure access mode, select use-ssl as disabled, optional, or required (optional is the preset value; you need the security library installed if you choose required). Click **Add Cluster**. To edit a cluster, click the cluster name on the List Clusters page to open the Edit Cluster page, make the changes, then click **Modify Cluster**.

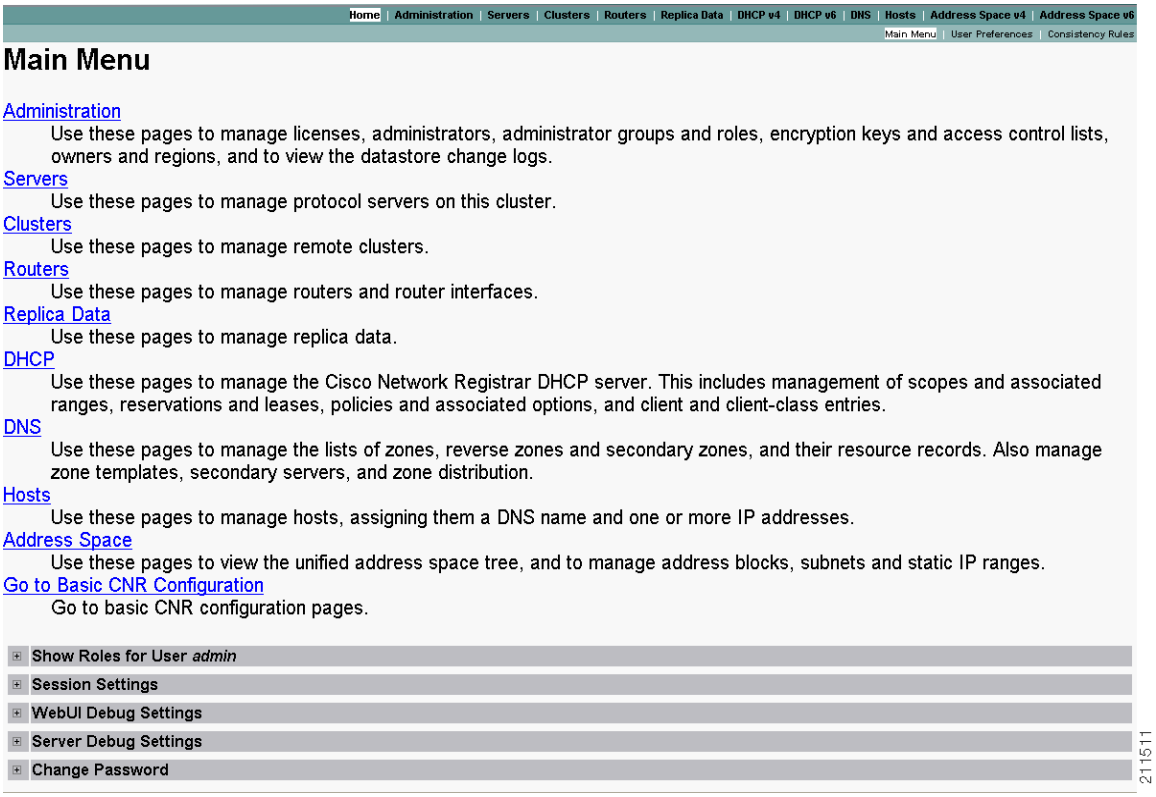


Note If you change the IP address of your local cluster machine, you must modify the **localhost** cluster to change the address in the ipaddr field. Avoid setting the value to the loopback address (127.0.0.1); if you do, you must also set the actual IP addresses of main and backup servers for DHCP failover and High-Availability (HA) DNS configurations.

Regional Cluster Web UI

The regional cluster web UI provides concurrent access to regional and central administration tasks. It provides granular administration across servers with permissions you can set on a per element or feature basis. Once you log in to the application, the Main Menu page appears (see Figure 2-8 on page 2-12). Regional cluster administration is described in Chapter 6, "Managing the Central Configuration."

Figure 2-8 Regional Main Menu Page



See Also

Introduction to the Web-Based User Interfaces, page 2-1
Local Cluster Web UI, page 2-7

Command Line Interface

Using the Network Registrar CLI (the **nrcmd** program), you can control your local cluster server operations. You can set all configurable options, as well as start and stop the servers.

Note The CLI provides concurrent access, by at most 14 simultaneous users and processes per cluster.

Tip See the **CLIContents.html** file in the **/docs** subdirectory of your installation directory for details.

The **nrcmd** program for the CLI is located on:

- **Windows**—In the *install-path*\bin directory.
- **Solaris and Linux**—In the *install-path*/usrbin directory.

On a local cluster, once you are in the appropriate directory, use the following command at the prompt:

nrcmd -C *clustername* **-N** *username* **-P** *password* [**-L** | **-R**]

- **–C** is the cluster name, preset value **localhost**.
- **–N** is the username, initially **admin**.
- **–P** is the user password, initially **changeme**.
- The local cluster (**–L**) is implied; use **–R** to open the regional cluster CLI.



Tip Change the initial password right away (see the "Managing Passwords" section on page 5-7). For additional command options, see the **CLIGuide.html** file in /docs.



Note If you change the IP address of your local cluster machine, you must modify the **localhost** cluster to change the address in the *ipaddress* attribute. Do not set the value to 127.0.0.1.

You can also pipe output to a file. For example:

```
nrcmd> dns getStats all > dnsstats.txt
```

To disconnect from the cluster, use **exit**:

```
nrcmd> exit
```



Tip The CLI operates on a coordinated basis with multiple user logins. If you receive a cluster lock message, determine who has the lock and discuss the issue with that person. You can override the lock by using the **force-lock** command, but not unless you are absolutely sure you would thereby not adversely affect someone else's work.

Central Configuration Management Server

The CCM servers at the local and regional clusters provide the infrastructure for Network Registrar operation and user interfaces. The web UIs operate with a combination of the legacy Network Registrar databases (MCD) and the additional CCM databases. The main purpose of the local cluster web UI infrastructure is to store and propagate data from the user to the protocol servers, and from the servers back to the user.

The change set is the fundamental unit of change to a data store. It sends incremental changes to a replicating server and provides an audit log for changes to the data store. Change sets consist of lists of change entries that are groups of one or more changes to a single network object. The web UI provides a view of the change sets for each data store.



CHAPTER 3

Server Status Dashboard

The Network Registrar server status dashboard in the web user interface (web UI) presents a graphical view of the system status, using graphs, charts, and tables, to help in tracking and diagnosis. These dashboard elements are designed to convey system information in an organized and consolidated way, and include:

- Significant protocol server and other metrics
- Alarms and alerts
- Database inventories
- Server health trends

The dashboard is best used in a troubleshooting desk context, where the system displaying the dashboard is dedicated for that purpose and might be distinct from the systems running the protocol servers. The dashboard system should point its browser to the system running the protocol servers.

You should interpret dashboard indicators in terms of deviations from your expected normal usage pattern. If you notice unusual spikes or drops in activity, there could be communication failures or power outages on the network that you need to investigate.

See Also

Opening the Dashboard
Display Types, page 3-2
Customizing the Display, page 3-6
Host Metrics, page 3-8
DHCP Metrics, page 3-10
DNS Metrics, page 3-15

Opening the Dashboard

Open the separate server status dashboard window by clicking **Dashboard** in the top right corner of the web UI page, between **Help** and **Logout**, in Basic, Advanced, and Expert mode.

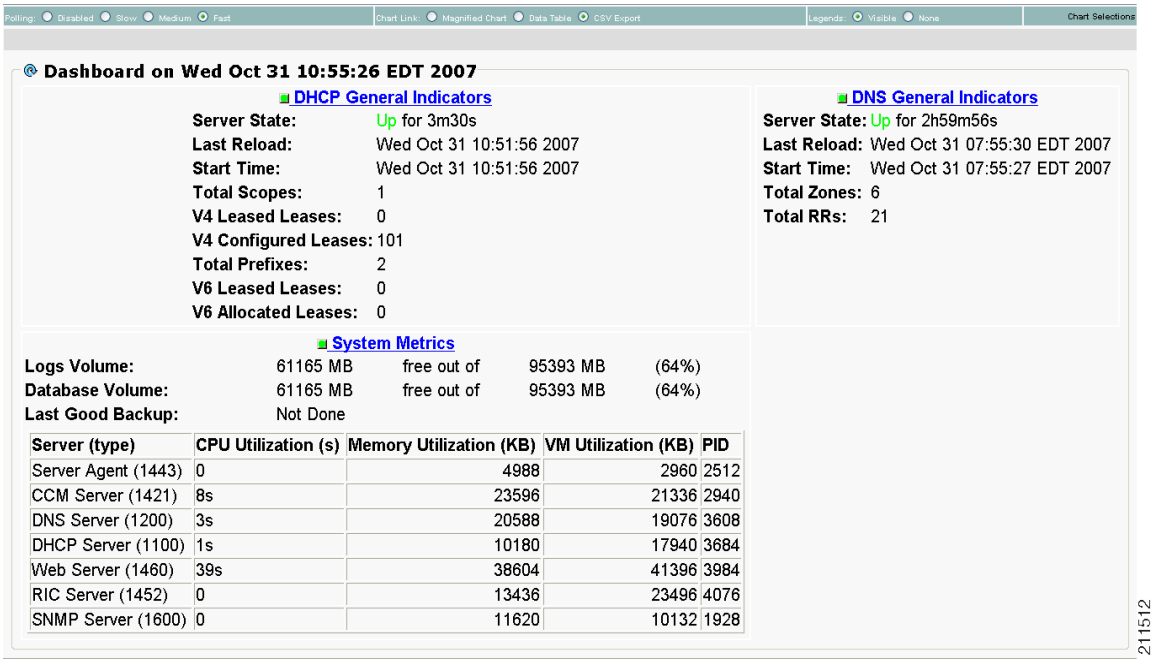
Display Types

Provided you have DHCP and DNS privileges through administrator roles assigned to you, the preset display of the dashboard consists of the following tables (see Figure 3-1 for an example):

- **System Metrics**—See the "System Metrics" section on page 3-9.
- **DHCP General Indicators**—See the "DHCP General Indicators" section on page 3-15.
- **DNS General Indicators**—See the "DNS General Indicators" section on page 3-19.

Tip These are just the preset selections. See the "Selecting Dashboard Elements to Include" section on page 3-7 for other dashboard elements you can select. The dashboard retains your selections from session to session.

Figure 3-1 Preset Dashboard Elements



Each dashboard element initially appears as a table or a specific chart type, depending on the element:

- **Table**—See the "Tables" section on page 3-3.
- **Line chart**—See the "Line Charts" section on page 3-3.
- **Stacked area chart**—See the "Stacked Area Charts" section on page 3-5.

General Status Indicators

Note the green box next to each dashboard element name in Figure 3-1 on page 3-2. This box indicates that the server sourcing the information is functioning normally. A yellow box indicates that server operation is less than optimum. A red box indicates that the server is down. These indicators are the same as for the server health on the Manage Servers page in the regular web UI.

Graphic Indicators for Levels of Alert

Graphed lines and stacked areas in the charts follow a standard color and visual coding so that you can immediately determine key diagnostic indicators at a glance. The charts use the following color and textural indicators:

- **High alerts or warnings**—Lines or areas in red, with a hatched texture.
- **All other indicators**—Lines or areas in various other colors distinguish the data elements. The charts do not use green or yellow.

Magnifying and Converting Charts

If Magnified Chart is the selected Chart Link (see Figure 3-5 on page 3-7), you can magnify a chart in a separate window by clicking the chart. In magnified chart view, you can choose an alternative chart type from the one that comes up initially (see the "Other Chart Types" section on page 3-5).

Note Automatic refresh is turned off for magnified charts (see the "Setting the Polling Interval" section on page 3-6). To get the most recent data, click the Refresh icon ( ) next to the word Dashboard at the top left of the page.

To convert a chart to a table, see the "Displaying Charts as Tables" section on page 3-7. You cannot convert tables to a graphic chart format.

Legends

Each chart initially includes a color-coded legend. To turn off the legend display on the main dashboard page, see the "Displaying or Hiding Chart Legends" section on page 3-7. Removing the legend renders the graphic chart size relatively larger, which can be helpful if you have many charts displayed. You cannot remove legends in magnified views.

Tables

Dashboard elements rendered as tables have data displayed in rows and columns. (See Figure 3-1 on page 3-2 for examples.) Note that if you view a table in Expert mode, additional data might appear. The following dashboard elements are preset to consist of (or include) tables:

- System Metrics
- DHCP Buffer Capacity
- DHCP DNS Updates
- DHCP Address Current Utilization
- DHCP Failover Status
- DHCP General Indicators
- DNS General Indicators

Line Charts

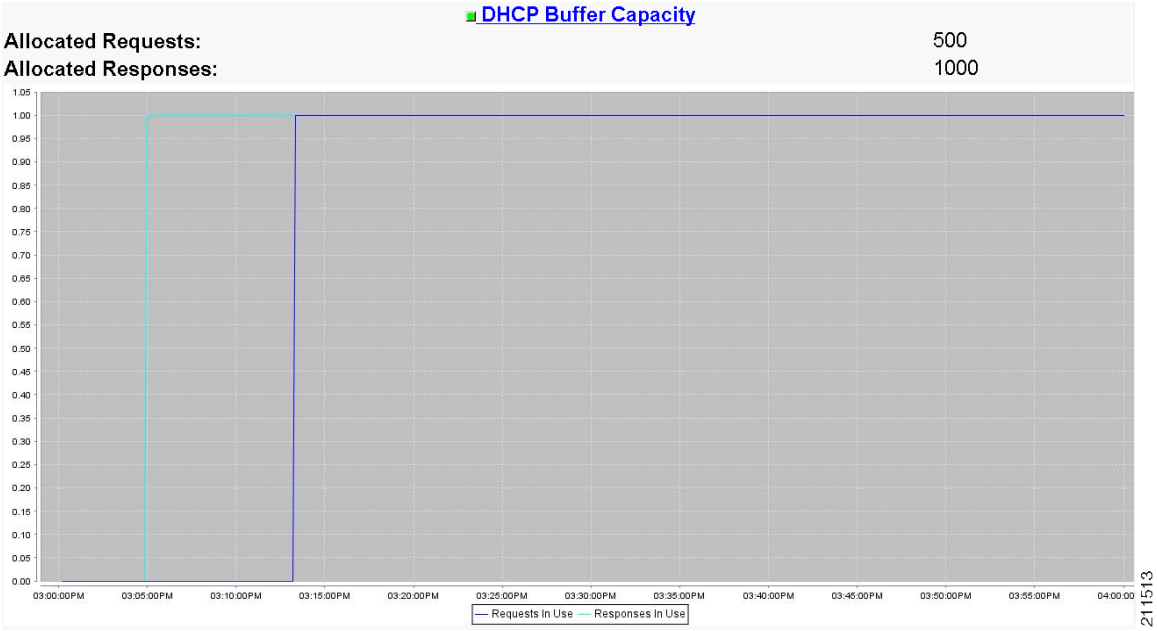
Dashboard elements rendered as line charts can include one or more lines plotted against the x and y axes. The three types of line charts are described in Table 3-1 on page 3-4.

Table 3-1 Line Chart Types

Type of Line Chart	Description	Dashboard Elements Rendered
Raw data line chart	Lines plotted against raw data.	• Java Virtual Machine (JVM) Memory Utilization (Expert mode only) • DHCP Buffer Capacity (see Figure 3-2) • DHCP Failover Status (two charts) • DNS Network Errors • DNS Related Servers Errors
Delta line chart	Lines plotted against the difference between two sequential raw data.	• DNS Inbound Zone Transfers • DNS Outbound Zone Transfers
Rate line chart	Lines plotted against the difference between two sequential raw data divided by the sample time between them.	• DHCP Server Request Activity • DHCP Server Response Activity • DHCP Response Latency • DNS Query Responses • DNS Forwarding Errors

Tip To get the raw data for a chart that shows delta or rate data, enter Expert mode, set the Chart Link to Data Table (see the "Displaying Charts as Tables" section on page 3-7), then click the chart. The Raw Data table is below the Chart Data table.

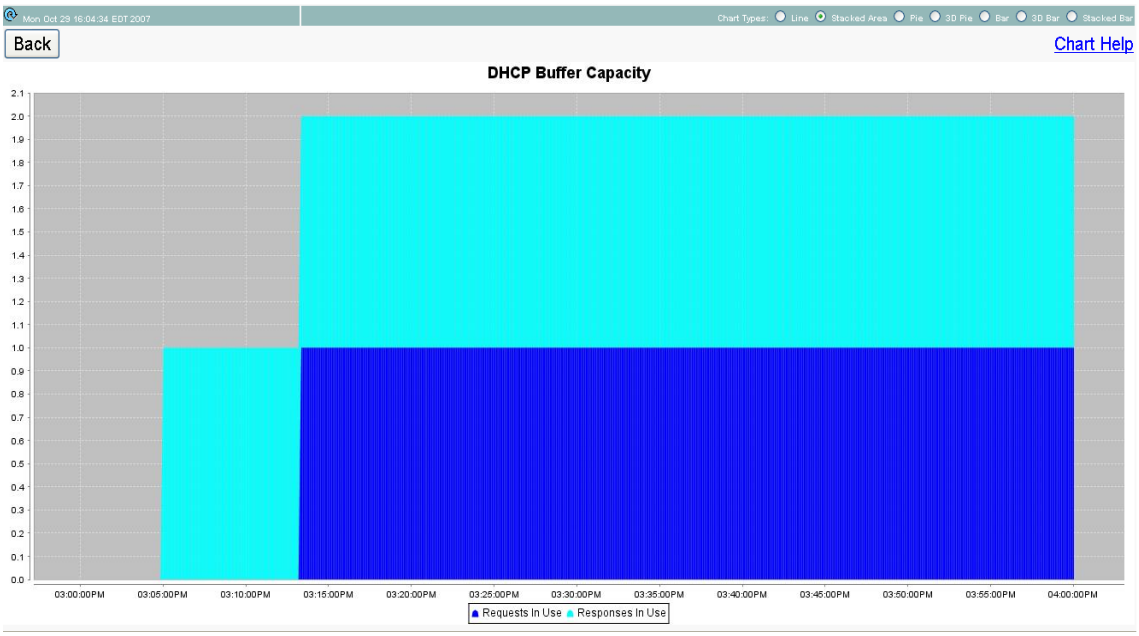
Figure 3-2 Line Chart Example



Stacked Area Charts

Dashboard elements rendered as stacked area charts have multiple related metrics plotted as trend charts, but stacked one on top of the other, so that the highest point represents a cumulative value. The values are independently shaded in contrasting colors. (See Figure 3-3 for an example of the DHCP Buffer Capacity chart shown in Figure 3-2 on page 3-4 rendered as a stacked area chart.)

Figure 3-3 Stacked Area Chart Example



The areas are stacked in the order listed in the legend, the left-most legend item appearing at the bottom of the stack and the right-most legend item at the top of the stack. In Figure 3-3, for example, the Requests in Use appear at the bottom of the stack and the Responses in Use appear at the top of the stack.

The dashboard elements that are preset to stacked area charts are:

- DHCP Server Request Activity
- DHCP Server Response Activity
- DHCP Response Latency

Other Chart Types

In the magnified chart view, you can convert the preset charts for each dashboard element into other chart types.

Tip Each chart type shows the data in distinct ways and in different interpretations. You can decide which type best suits your needs.

At the top of the page are radio buttons where you can choose from the following chart types:

- **Line**—One of the line charts described in Table 3-1 on page 3-4.
- **Stacked Area**—Charts described in the "Stacked Area Charts" section on page 3-5.
- **Pie**—Shows a single percentage pie chart of the data averaged over the time sampled.
- **3D Pie**—Three-dimensional rendering of the pie chart.
- **Bar**—Multiple related current value metrics plotted side by side as groups of bars that show the actual data sampled.
- **3D Bar**—Three-dimensional rendering of the bar chart.
- **Stacked Bar**—Addition total of the actual samples. This chart shows more distinct data points than the stacked area chart.

Getting Help for the Dashboard Elements

You can open a help window for each dashboard element by clicking the title of the element. In magnified view, click **Chart Help**.

Customizing the Display

To customize the dashboard display, you can:

- Refresh the data and set an automatic refresh interval.
- Expand a chart and render it in a different format.
- Convert a graphic chart to a table.
- Download data to comma-separated value (CSV) output.
- Display or hide chart legends.

Refreshing Displays

Refresh each display so that it picks up the most recent polling by clicking the Refresh icon ( ).

Setting the Polling Interval

You can set how often to poll for data. In the upper-left corner of the dashboard display are the controls for the polling interval of the cached data, which polls the protocol servers for updates. (See Figure 3-4.)

Figure 3-4 Setting the Chart Polling Interval



You can set the cached data polling (hence, automatic refresh) interval to:

- **Disabled**—Does not poll, therefore does not automatically refresh the data.
- **Slow**—Refreshes the data every 30 seconds.
- **Medium**—Refreshes the data every 20 seconds.
- **Fast** (the preset value)—Refreshes the data every 10 seconds.

Displaying Charts as Tables

You can choose to display a graphic chart as a table when you magnify the chart by clicking it (see the "Magnifying and Converting Charts" section on page 3-3). At the middle of the top of the dashboard display are the controls for the chart links (see Figure 3-5).

Figure 3-5 Specifying Chart Conversion to Table Format



Click the **Data Table** radio button. When you click the chart itself, it opens as a table. The preset display format is Magnified Chart.

Exporting to CSV Format

You can dump the chart data to a comma-separated value (CSV) file (such as a spreadsheet) when you magnify the chart by clicking it. In the Chart Link controls at the top of the page (see Figure 3-5), click the **CSV Export** radio button, then click the chart. A Save As window appears, where you can specify the name and location of the CSV file.

Displaying or Hiding Chart Legends

You can include or exclude the color-coded legends for charts on the main dashboard page. You might want to remove the legends as you become more familiar with the data and track it on a slightly larger chart display. In the upper-right of the dashboard display are the controls for the legend display (see Figure 3-6). The preset value is Visible.

Figure 3-6 Displaying or Hiding Chart Legends and Selecting Chart



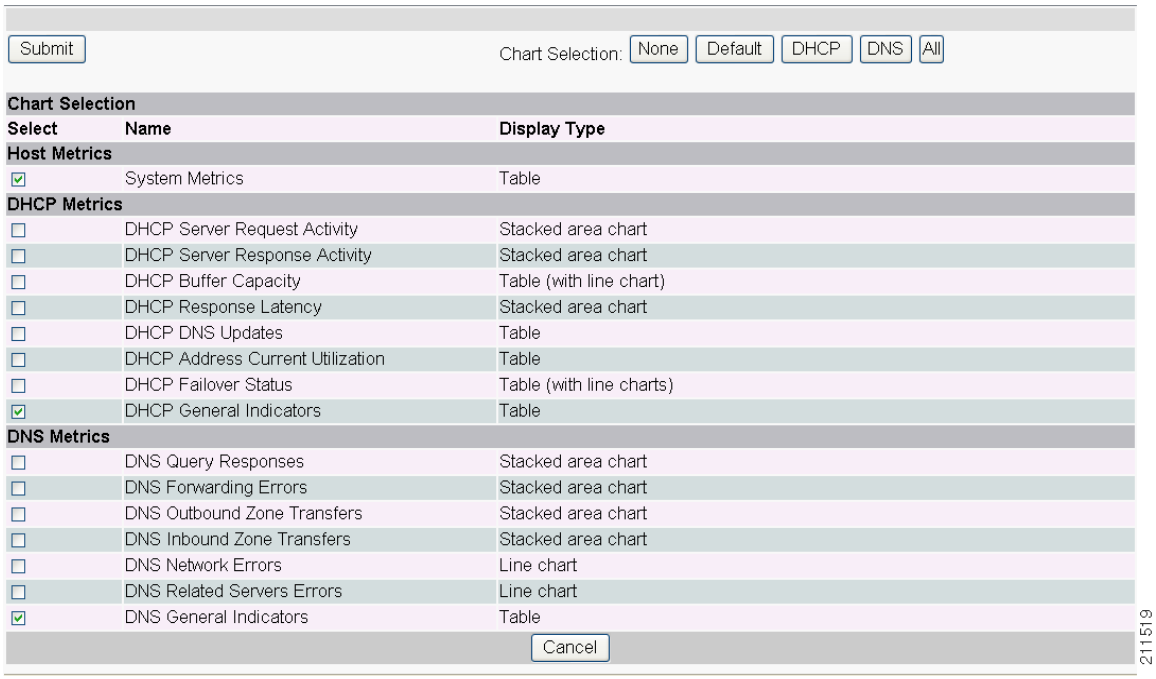
Selecting Dashboard Elements to Include

You can decide how many dashboard elements you want to display on the page. At times, you might want to focus on one server activity only, such as for the DHCP server, and exclude all other metrics for the other servers. In this way, the dashboard becomes less crowded, the elements are larger and more readable. At other times, you might want an overview of all server activities, with a resulting smaller element display.

You can select the dashboard elements to display from the main Dashboard page by clicking **Chart Selections** in the upper right corner of the page (see Figure 3-6). Clicking the link opens the Chart Selection page (see Figure 3-7 on page 3-8).

Tip The order in which the dashboard elements appear in the Chart Selection list does not necessarily determine the order in which the elements will appear on the page. An algorithm that considers the available space determines the order and size in a grid layout. The layout might be different each time you submit the dashboard element selections.

Figure 3-7 Selecting Dashboard Elements



To change selections, click a check mark in the box next to the dashboard element that you want to display. Specific group controls are available in the upper-right corner of the page. To:

- Remove all check marks, click **None**.
- Revert to the preset selections, click **Default**. The preset dashboard elements for administrator roles supporting DHCP and DNS are:
  - Host Metrics: System Metrics (see the "Host Metrics" section on page 3-8)
  - DHCP Metrics: General Indicators
  - DNS Metrics: General Indicators
- Choose the DHCP metrics only, click **DHCP** (see the "DHCP Metrics" section on page 3-10).
- Choose the DNS metrics only, click **DNS** (see the "DNS Metrics" section on page 3-15).
- Choose all the dashboard elements, click **All**.

Click **Submit** at the top left of the page to submit your choices, or **Cancel** at the bottom of the page.

Host Metrics

Host metrics comprise two charts:

- **System Metrics**—See the "System Metrics" section on page 3-9.
- **JVM Memory Utilization** (available in Expert mode only)—See the "JVM Memory Utilization" section on page 3-10.

System Metrics

The System Metrics dashboard element shows the free space on the disk volumes where the Network Registrar logs and database directories are located, the date and time of the last server backup, and CPU and memory usage for the various servers. System metrics are available if you choose **Host Metrics**: **System Metrics** in the Chart Selection list.

The resulting table shows:

- **Logs Volume**—Current free space out of the total space on the disk drive where the logs directory is located, with the equivalent percentage of free space.
- **Database Volume**—Current free space out of the total space on the disk drive where the data directory is located, with the equivalent percentage of free space.
- **Last Good Backup**—Date and time when the last successful shadow database backup occurred (or Not Done if it did not yet occur) since the server agent was last started.
- **CPU Utilization** (in seconds), **Memory Utilization** (in kilobytes), and (in Expert mode only) the **VM Utilization** (in kilobytes) and Process ID (**PID**) for the:
  - Network Registrar server agent
  - CCM server
  - DNS server
  - DHCP server
  - Web server
  - Router Interface Configuration (RIC) server
  - SNMP server

How to Interpret the Data

The System Metrics data shows how full your disk volumes are getting based on the available free space for the Network Registrar logs and data volumes. It also shows if you had a last successful backup of the data files and when that occurred. Finally, it shows how much of the available CPU and memory the Network Registrar servers are using. The difference in the memory and VM utilization values is:

- **Memory Utilization**—Physical memory that a process uses, or roughly equivalent to the Resident Set Size (RSS) value in UNIX **ps** command output, or to the Task Manager Mem Usage value in Windows: the number of pages the process has in real memory minus administrative usage. This value includes only the pages that count toward text, data, or stack space, but not those demand-loaded in or swapped out.
- **VM Utilization**—Virtual memory that a process uses, or roughly equivalent to the SZ value in UNIX **ps** command output, or to the Task Manager VM Size value in Windows: the in-memory pages plus the page files and demand-zero pages, but not usually the memory-mapped files. This value is useful in diagnosing how large a process is and if it continues to grow.

Troubleshooting Based on the Results

If you notice the free disk space decreasing for the logs or data directory, you might want to consider increasing the disk capacity or look at the programs you are running concurrently with Network Registrar.

JVM Memory Utilization

The Java Virtual Machine (JVM) Memory Utilization dashboard element is available only when you are in Expert mode. It is rendered as a line trend chart that traces the Unused Maximum, Free, and Used bytes of JVM memory. The chart is available if you choose **Host Metrics**: **JVM Memory Utilization** in the Chart Selection list when you are in Expert mode.

How to Interpret the Data

The JVM Memory Utilization data shows how much memory applies to running the dashboard in your browser. If you see the Used byte data spiking, dashboard elements might be using too much memory.

Troubleshooting Based on the Results

If you see spikes in Used memory data, check your browser settings or adjust the polling interval to poll for data less frequently.

DHCP Metrics

These DHCP metric elements are available in the dashboard:

- **DHCP Server Request Activity**—See the "DHCP Server Request Activity" section.
- **DHCP Server Response Activity**—See the "DHCP Server Response Activity" section on page 3-11.
- **DHCP Buffer Capacity**—See the "DHCP Buffer Capacity" section on page 3-12.
- **DHCP Response Latency**—See the "DHCP Response Latency" section on page 3-12.
- **DHCP DNS Update Activity**—See the "DHCP DNS Updates" section on page 3-13.
- **DHCP Address Utilization**—See the "DHCP Address Current Utilization" section on page 3-13.
- **DHCP Failover Status**—See the "DHCP Failover Status" section on page 3-14.
- **DHCP General Indicators**—See the "DHCP General Indicators" section on page 3-15.

DHCP Server Request Activity

The DHCP Server Request Activity dashboard element rendered as a stacked area chart traces the totals in the change rate of incoming DHCP packet activity. The chart is available if you choose **DHCP Metrics**: **DHCP Server Request Activity** in the Chart Selection list.

The resulting stacked area chart plots the following trends:

- **V4 Discovers**—Number of DHCPv4 discover packets.
- **V4 Requests**—Number of DHCPv4 request packets.
- **V4 Other**—Number of DHCPv4 release, decline, or info-request packets.
- **V4 Lease Queries**—Number of DHCPv4 lease query packets.
- **V6 Solicits**—Number of DHCPv6 solicit packets.
- **V6 Requests/Renews/Rebinds**—Number of DHCPv6 request, renew, and rebind packets.
- **V6 Other**—Number of DHCPv6 release, decline, or information-request packets.

- **V6 Lease Queries**—Number of DHCPv6 lease query packets.
- **Invalid Packets**—Combined number of invalid DHCPv4 and DHCPv6 packets.

How to Interpret the Data

The DHCP Server Request Activity data shows the pattern of server traffic based on incoming DHCP requests. The trend should be fairly consistent, with spikes in the number of Invalid packets being a sign that there is some misconfigured data on the network. Note that DHCPv4 and DHCPv6 invalid packet activity is grouped together.

Troubleshooting Based on the Results

Check your DHCP server configurations if there is a sudden spike in activity, especially in the number of invalid request packets. Set your server logging to report where the activity is occurring. Spikes or drops in activity can indicate network or power outages that are worth investigating. Spikes in activity can also indicate a faulty client, malicious client activity, or a recovery after a power failure or outage that results in pent-up requests.

DHCP Server Response Activity

The DHCP Server Response Activity dashboard element rendered as a stacked area chart traces the totals in the change rate of outgoing DHCP packet activity. The chart is available if you choose **DHCP Metrics**: **DHCP Server Response Activity** in the Chart Selection list.

The resulting stacked area chart plots the following trends:

- **V4 Offers**—Number of DHCPv4 offer packets.
- **V4 Acks**—Number of DHCPv4 acknowledgment packets.
- **V4 Other Client**—Number of other outgoing DHCPv4 client packets.
- **V4 Lease Queries**—Number of outgoing DHCPv4 lease query packets.
- **V6 Advertises**—Number of DHCPv6 advertise packets.
- **V6 Replies**—Number of DHCPv6 reply packets.
- **V6 Reconfigures**—Number of DHCPv6 reconfigure packets.
- **V6 Lease Query Replies**—Number of DHCPv6 lease query reply packets.
- **Total Dropped**—Combined number of dropped DHCPv4 and DHCPv6 packets.

How to Interpret the Data

The DHCP Server Response Activity data shows the pattern of server traffic to answer DHCP requests. The trend should be fairly consistent, with spikes in the number of Total Dropped packets being a sign that there is some misconfigured data on the network. Note that DHCPv4 and DHCPv6 dropped packet activity is grouped together.

Troubleshooting Based on the Results

Check your DHCP server configurations if there is a sudden spike in activity, especially in the number of total dropped response packets. The response activity should match the request activity, except for the normal time shift, and the same diagnostics apply.

DHCP Buffer Capacity

The DHCP Buffer Capacity dashboard element rendered as a table shows the number of allocated requests and responses, and a line chart that plots the number of requests and responses in use. The element is available if you choose **DHCP Metrics: DHCP Buffer Capacity** in the Chart Selection list.

The resulting table and line chart plots:

- **Requests in Use**—Trend in the number of in-use request buffers.
- **Responses in Use**—Trend in the number of in-use response buffers.

How to Interpret the Data

The DHCP Buffer Capacity data shows the pattern in the use of DHCP request and response buffers. If the buffers begin to increase in an abnormal pattern, there are measures you can take without trying to compensate by increasing the number of allocated buffers.

Troubleshooting Based on the Results

If you see increasing and consistent exceeding of the buffer threshold, find the reason why the server is running slowly. Possible reasons include high degrees of logging, slow DHCP extensions or LDAP servers, or overload, such as with chatty clients or frequent rebooting of cable modem termination systems (CMTSs). You might need to increase the buffer sizes.

DHCP Response Latency

The DHCP Response Latency dashboard element rendered as a stacked area chart shows the trend in the response packet latency (the time interval between the request packet and its ensuing response). The chart is available if you choose **DHCP Metrics: DHCP Response Latency** in the Chart Selection list.



Tip You must also set the *collect-sample-counters* DHCP server attribute for this data to display, with the *enhanced-sample-counters* attribute also set for further granularity. These attribute values are preset. If you are concerned about achieving maximum performance, unset these attributes. (See the "Displaying Statistics" section on page 7-11.)

The resulting stacked area chart plots response latencies at the intervals:

- Less than 50 milliseconds
- 50 to 200 milliseconds
- 200 to 500 milliseconds
- 500 to 1000 milliseconds (note that if the *enhanced-sample-counters* attribute is not set, all values below 1 second appear in this grouping)
- 1 to 2 seconds
- 2 to 3 seconds
- 3 to 4 seconds
- More than 4 seconds

How to Interpret the Data

The chart shows the trend in response packet latency as an indicator of how long it takes to respond to incoming packets. The gradations in the latency periods are stacked.

Troubleshooting Based on the Results

High response packet latency is similar to high buffer usage for troubleshooting purposes. Look for slow LDAP servers or DHCP extensions, high levels of logging, or disk I/O bottlenecks.

DHCP DNS Updates

The DHCP DNS Updates dashboard element rendered as a table shows the related DNS server and its current state, and how many pending DNS updates are occurring between it and the DHCP server. The table is available if you choose **DHCP Metrics: DHCP DNS Updates** in the Chart Selection list.

The resulting table shows:

- **Server**—Related DNS server IP address
- **State**—Related DNS server state
- **Pending Updates**—Total number of pending updates

How to Interpret the Data

A high level of pending updates to a specific DNS server indicates that the server is unreachable or unavailable, or its address is wrong.

Troubleshooting Based on the Results

Check into the reachability of the associated DNS servers if the pending update rate spikes, or ensure that the address of the associated server is correct.

DHCP Address Current Utilization

The DHCP Address Current Utilization dashboard element rendered as a table shows the DHCPv4 address utilization (how many assigned addresses exist) for a particular address aggregation, which can be a scope, network, or network plus selection tag. The table is available if you choose **DHCP Metrics: DHCP Address Current Utilization** in the Chart Selection list.

The resulting table shows:

- **Name**—Aggregation name (or address).
- **In Use**—Number of in-use addresses.
- **Total**—Total number of addresses.
- **Utilization**—Percentage of utilized addresses.
- **Mode** (appears in Expert mode only)—Aggregation mode (scope, network, or selection-tags).

How to Interpret the Data

The chart shows a table with four columns: the scope name, its in-use and total addresses, and the percentage of address utilization based on the previous two columns. The chart is available only if the DHCP server *enhanced-sample-counters* attribute is enabled.

Note If an SNMP trap configuration in scope mode applies, the Name column shows the scope name, otherwise it shows the network IP address. If traps are not enabled (or if the DHCP server *default-free-address-config* attribute is not set), the network address is appended with an asterisk (*). If a scope-selection tag applies, its name is also appended. (See the "Handling SNMP Notification Events" section on page 1-7 for details on SNMP traps.)

Troubleshooting Based on the Results

If the percentage of utilized addresses is high, the addresses reached a saturation point. It might be necessary to reassign addresses from a different scope.

DHCP Failover Status

The DHCP Failover Status dashboard element rendered as two parallel trend charts that show the current and partner server state and the binding updates and acknowledgments sent and received between the two failover partners. The charts are available if you choose **DHCP Metrics: DHCP Failover Status** in the Chart Selection list.

Note The failover status is only for the first failover pair in the related servers list.

The display is a table along with two rate line trend charts that shows the failover status for the first failover pair for the related servers:

- **Local State**—Local DHCP server failover state along with when it occurred.
- **Partner State**—Partner server failover state along with when it occurred.
- **DHCP Failover Status Updates Received**—The first trend chart shows a comparison of the number of binding updates received and binding acknowledgments sent.
- **DHCP Failover Status Updates Sent**—The second trend chart shows a comparison of the number of binding updates sent and binding acknowledgments received.

How to Interpret the Data

Along with some state data, the display is split into two line trend charts that are inverses of each other. Each chart compares the binding updates with the acknowledgments. The top chart pairs the binding updates received with the acknowledgments sent; the bottom chart pairs the binding updates sent with the acknowledgments received.

Troubleshooting Based on the Results

If the Partner State value is other than 10, check the configuration of the partner server. The updates sent and received data should also be fairly level.

DHCP General Indicators

The DHCP General Indicators dashboard element rendered as a table shows the server state, reload data, and lease counts. The table is available if you choose **DHCP Metrics: DHCP General Indicators** in the Chart Selection list.

The resulting table shows:

- **Server State**—Up or Down (based on whether statistics are available) and its duration.
- **Last Reload**—Date and time of the last server reload.
- **Start Time**—Date and time of the last server process (Network Registrar server agent) startup.
- **Total Scopes**—Total number of configured DHCPv4 scopes.
- **V4 Leased Leases**—Number of active DHCPv4 leases, including reservations.
- **V4 Configured Leases**—Number of configured DHCPv4 leases, including reservations and ranges.
- **Total Prefixes**—Number of configured DHCPv6 prefixes.
- **V6 Leased Leases**—Number of active DHCPv6 leases, including reservations and delegated prefixes (which each count as one lease).
- **V6 Allocated Leases**—Number of allocated DHCPv6 leases, including reservations and delegated prefixes (which each count as one lease).

How to Interpret the Data

The table indicates the server state, process start time (via the Network Registrar server agent), and reload data, and also provides lease statistics. The top set of data compares the DHCPv4 leases actually in effect with those configured; the bottom set of data does the same for DHCPv6 leases.

Time of last reload is important for determining if recent changes to the server configuration occurred from a reload operation. It can also help pinpoint when server changes were last applied, if other indicators show a marked, unexpected behavioral change. Be sure to preserve log files since the last reload.

Troubleshooting Based on the Results

A drop or increase in leases might indicate a power or network outage, but it can also indicate a normal variation depending on lease times and usage patterns. The number of scopes or prefixes indicated might also require some evaluation and possible reconfiguration. If the server state is Down, all the DHCP chart indicators show a red status box, so no data will be available. In the case of a server that is down, restart the server.

DNS Metrics

These DNS metric elements are available in the dashboard:

- **DNS Query Responses**—See the "DNS Query Responses" section on page 3-16.
- **DNS Forwarding Errors**—See the "DNS Forwarding Errors" section on page 3-16.
- **DNS Outbound Zone Transfers**—See the "DNS Outbound Zone Transfers" section on page 3-17.
- **DNS Inbound Zone Transfers**—See the "DNS Inbound Zone Transfers" section on page 3-17.
- **DNS Network Errors**—See the "DNS Network Errors" section on page 3-18.

- **DNS Related Servers Errors**—See the "DNS Related Servers Errors" section on page 3-18.
- **DNS General Indicators**—See the "DNS General Indicators" section on page 3-19.

DNS Query Responses

The DNS Query Responses dashboard element rendered as a stacked area chart traces the totals in the change rate of the number of responses answered from memory and the cache database, and forwarded to other DNS servers. The display is available if you choose **DNS Metrics: DNS Query Responses** in the Chart Selection list.

The resulting stacked area chart plots the following trends:

- **Memory**—Number of DNS queries answered entirely from in-memory cache.
- **Cache DB**—Number of queries answered from the cache database.
- **Forwarding**—Number of queries answered by having been forwarded to other servers.

How to Interpret the Data

This chart shows how the local DNS server responds to incoming queries, whether from memory or the cache database, or if the server forwards the queries. The rates of change of these values are identified separately, but stacked as a total in the chart. Because the responses from memory are the fastest, you should see a steady state or even increasing rate in the number of queries that the server responds to from memory. The queries answered from the cache database on disk, and especially those forwarded to other servers, result in less efficient responses, so the Cache DB and Forwarding rates should not be increasing.

Troubleshooting Based on the Results

The server should use memory cache as much as possible in responding to queries. Increasing rates of cache database reads and forwarded queries can indicate that not enough space is available in memory to store the cached queries for more efficient responses.

DNS Forwarding Errors

The DNS Forwarding Errors dashboard element rendered as a stacked area chart traces timeouts, lame delegations, and dropped forwarded queries. The chart is available if you choose **DNS Metrics: DNS Forwarding Errors** in the Chart Selection list.

The resulting stacked area chart plots the following trends:

- **Timeouts**—Number of queries timed out with no responses.
- **Lame Delegations**—Number of queries encountering lame delegation from remote servers.
- **Dropped**—Number of dropped forwarded queries.

How to Interpret the Data

This chart indicates the health of any forwarded queries as consolidated stacked data. The most significant data is the number of dropped forwarded queries. Also significant, although not as critical, is the number of timed-out forwarded queries. Lame delegations are relatively easy to fix.

Troubleshooting Based on the Results

Check your DNS configuration if spikes or increases occur in timeouts, dropped packets, or lame delegations.

DNS Outbound Zone Transfers

The DNS Outbound Zone Transfers dashboard element rendered as a stacked area chart tracks the rate of change in full and incremental outbound zone transfer responses, and any associated errors. The chart is available if you choose **DNS Metrics: DNS Outbound Zone Transfers** in the Chart Selection list.

The resulting stacked area chart plots the following trends:

- **Full Responses**—Number of full outbound zone transfers (AXFRs out).
- **Incremental Responses**—Number of incremental outbound zone transfers (IXFRs out).
- **Authorization Errors**—Number of unauthorized (refused) zone transfer requests.
- **Exceed Max Transfers Out**—Number of failed outbound transfers that exceed the maximum limit.
- **Other Errors**—Number of other outbound transfer errors that are not authorization errors.

How to Interpret the Data

This chart is useful in gauging if outbound zone transfers to a secondary DNS server are occurring as predicted and if there are any authorizations or failed transfer attempts in the process. The most significant indicator is the trend in the number of outbound zone transfers denied for lack of permission or for not being authorized for the zone.

Troubleshooting Based on the Results

Check the primary and secondary server configurations if there are errors or exceeded limits in the outbound zone transfers.

DNS Inbound Zone Transfers

The DNS Inbound Zone Transfers dashboard element rendered as a stacked area chart tracks the rate of change in full and incremental inbound zone transfer responses, and any associated errors. The chart is available if you choose **DNS Metrics: DNS Inbound Zone Transfers** in the Chart Selection list.

The resulting stacked area chart plots the following trends:

- **Full Responses**—Number of full inbound zone transfers (AXFRs in).
- **Incremental Responses**—Number of incremental inbound zone transfers (IXFRs in).
- **Authorization Errors**—Number of refused responses (xfer-in-auth-errors).
- **Failed Attempts**—Number of failures other than refusals (xfer-failed-attempts).
- **Exceed Max Transfers In**—Number of times that the concurrent inbound transfers reach the maximum limit.

How to Interpret the Data

This chart is useful in gauging if inbound zone transfers to a secondary DNS server are occurring as predicted and if there are any authentication or failed transfer attempts in the process. The most significant indicator is the trend in the number of inbound zone transfers denied for lack of permission, for not being authorized for the zone, or for other reasons.

Troubleshooting Based on the Results

Check the primary and secondary server configurations if there are errors or exceeded limits in the inbound zone transfers.

DNS Network Errors

The DNS Network Errors dashboard element rendered as a line chart tracks the rate of change in DNS server network errors. The chart is available if you choose **DNS Metrics: DNS Network Errors** in the Chart Selection list.

The resulting line chart plots the following trends:

- **Query Error Packets/Query Responses**—Ratio of query error packets over responses. Responses consist of:
  - Authoritative
  - Authoritative no-such-name
  - Authoritative no-such-data
  - Nonauthoritative
  - Nonauthoritative no-such-data
  - Requests refused
- **Non Error Dropped Packets/Query Responses**—Ratio of nonerror dropped packets (queries dropped) over responses.
- **Update Errors/Updates**—Ratio of DNS Update errors over total updates.

How to Interpret the Data

This chart indicates query and response errors as an indication of the health of the server.

Troubleshooting Based on the Results

Check the DNS server network configuration if errors are increasing.

DNS Related Servers Errors

The DNS Related Servers Errors dashboard element rendered as a line chart tracks the rate of change in DNS related server errors. The chart is available if you choose **DNS Metrics: DNS Related Servers Errors** in the Chart Selection list.

The resulting line chart plots the following trends:

- **Referral Timeouts/Referrals**—Ratio of referral timeouts over referrals.
- **Failed Responses/Total Incoming Zone Transfer Requests**—Ratio of failed responses over incoming zone transfer requests.
- **TSIG Errors/TSIG Attempts**—Ratio of transaction signature (TSIG) errors (bad times, keys, or signatures) over total TSIG attempts (successfully received packets).

How to Interpret the Data

This chart indicates the health of connections and data transfers with related DNS servers. All three chart lines can have diagnostic significance.

Troubleshooting Based on the Results

Check the configurations and connectivity of the related servers in HA DNS relationships if errors are increasing.

DNS General Indicators

The DNS General Indicators dashboard element rendered as a table shows the server state, its last and startup reload time, the number of zones per server, and the total resource record (RR) count. The table is available if you choose **DNS Metrics: DNS General Indicators** in the Chart Selection list.

The resulting table shows:

- **Server State**—Up or Down (based on whether statistics are available), and how long the server has been in this state.
- **Last Reload**—How long since the last server reload.
- **Start Time**—Date and time of the last server process (Network Registrar server agent) startup.
- **Total Zones**—Number of configured zones.
- **Total RRs**—Number of resource records.

How to Interpret the Data

The data in this chart shows general server health and operational duration. The objective is to make decisions about the server, such as whether it might be time for another reload, perhaps warranted by the number of configured zones.

Troubleshooting Based on the Results

If the server state is Down, all the DNS chart indicators show a red status box, so no data will be available. In the case of a server that is down, restart the server. The number of zones indicated might also require some evaluation and possible reconfiguration.



CHAPTER 4

Deploying Network Registrar

Cisco Network Registrar is a full featured, scalable Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), and Trivial File Transfer Protocol (TFTP) implementation for medium to large IP networks. It provides the key benefits of stabilizing the IP infrastructure and automating networking services, such as configuring clients and provisioning cable modems. This provides a foundation for policy-based networking. Service provider and enterprise users can better manage their networks to integrate with other network infrastructure software and business applications.

See Also

Target Users
Regional and Local Clusters, page 4-2
Deployment Scenarios, page 4-2
Configuration and Performance Guidelines, page 4-5
Interoperability with Earlier Releases, page 4-6

Target Users

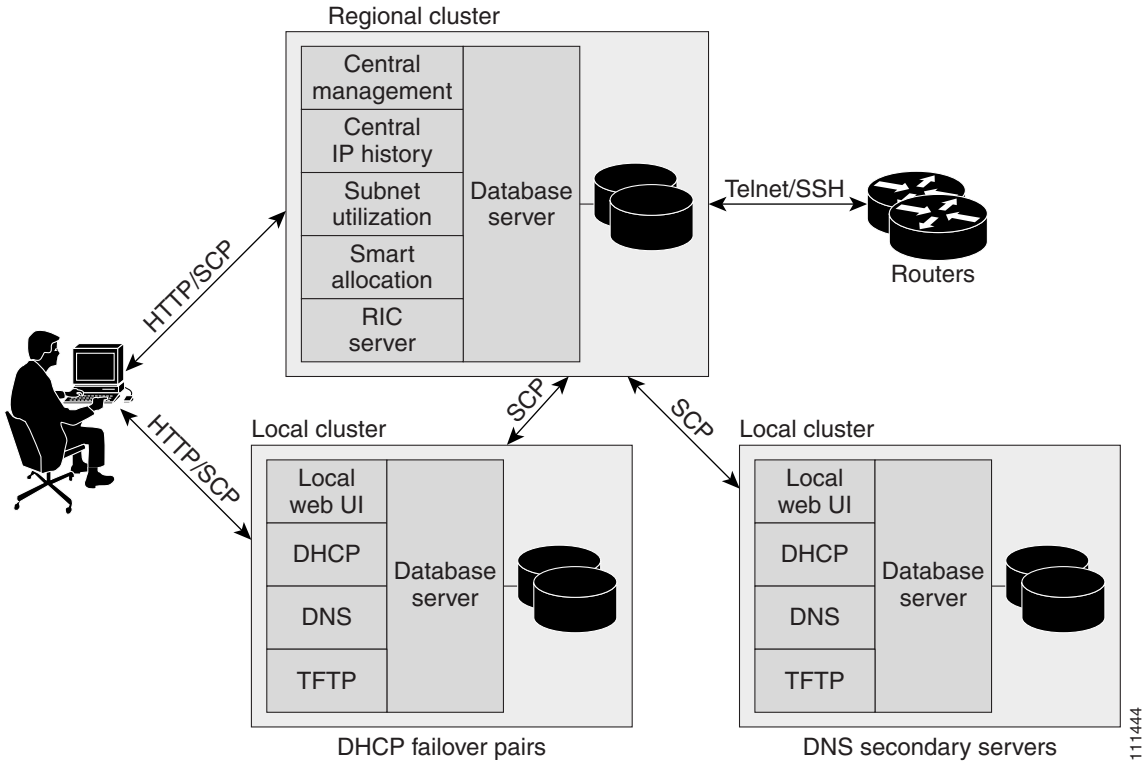
Network Registrar is designed for these users:

- **Internet service providers (ISPs)**—Helps ISPs drive the cost of operating networks that provide leased line, dialup, and DSL (Point-to-Point over Ethernet and DHCP) access to customers.
- **Multiple service operators (MSOs)**—Helps MSOs provide subscribers with Internet access using cable or wireless technologies. MSOs can benefit from services and tools providing reliable and manageable DHCP and DNS services that meet the Data Over Cable Service Interface Specification (DOCSIS). Network Registrar provides policy-based, robust, and scalable DNS and DHCP services that form the basis for a complete cable modem provisioning system.
- **Enterprises**—Helps meet the needs of single- and multisite enterprises (small-to-large businesses) to administer and control network functions. Network Registrar automates the tasks of assigning IP addresses and configuring the Transport Control Protocol/Internet Protocol (TCP/IP) software for individual network devices. Forward-looking enterprise users can benefit from class-of-service and other features that help integrate with new or existing network management applications, such as user registration.

Regional and Local Clusters

Network Registrar 6.2 extended the work of building on the local address server and address management architecture of earlier releases by providing additional features at the regional cluster (see Figure 4-1). This regional cluster acts as an aggregate management system for up to a hundred local clusters. Address and server administrators interact at the regional and local clusters through the regional and local web-based user interfaces (web UIs), and local cluster administrators can continue to use the command line interface (CLI) at the local cluster. The regional cluster consists of a Central Configuration Management (CCM) server, Router Interface Configuration (RIC) server, Tomcat web server, servlet engine, and server agent.

Figure 4-1 Network Registrar User Interfaces and Server Clusters



A typical deployment is one regional cluster at a customer network operation center (NOC), the central point of network operations for an organization. Each division of the organization includes a local address management server cluster responsible for managing a part of the network. The System Configuration Protocol (SCP) communicates the configuration changes between the servers.

The regional and local cluster can also manage a RIC server responsible for end point cable modem termination systems (CMTSs). (See Chapter 11, "Managing Router Interface Configurations.")

Deployment Scenarios

The Network Registrar regional cluster web UI provides a single point to manage any number of local clusters hosting DNS, DHCP, or TFTP servers. The regional and local clusters also provide administrator management so that you can assign administrative roles to users logged in to the application.

This section describes two basic administrative scenarios and the hardware and software deployments for two different types of installations—a small-to-medium local area network (LAN), and a large-enterprise or service-provider network with three geographic locations.

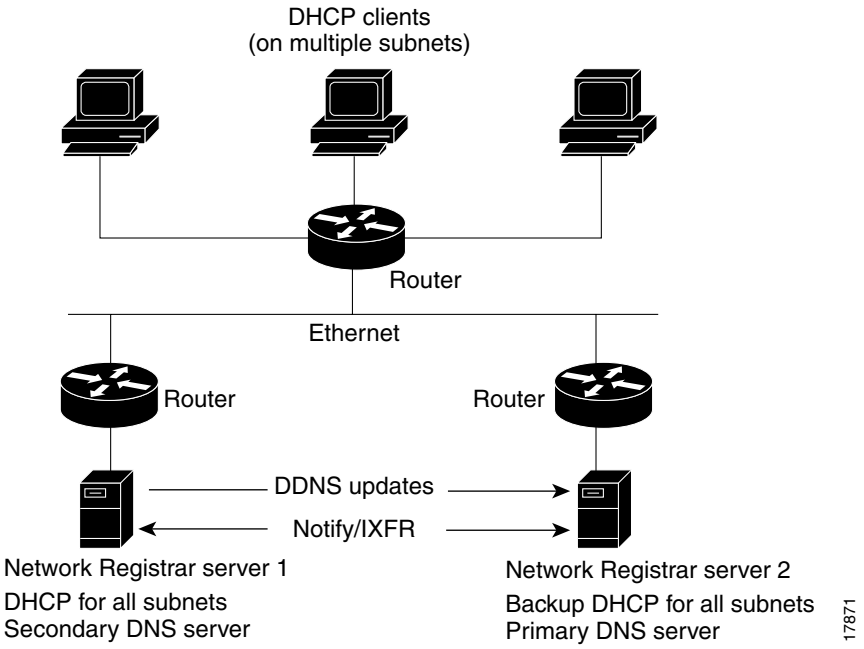
See Also

Small-to-Medium-Size LANs, page 4-3
Large Enterprise and Service Provider Networks, page 4-3

Small-to-Medium-Size LANs

In a small-to-medium LAN serving fewer than 50,000 DHCP clients, you can deploy Network Registrar without a regional cluster component. In this scenario, low-end Windows, Solaris, or Linux servers are acceptable. You can also use systems with EIDE disks, although Cisco recommends Ultra-SCSI disks for dynamic DNS update. Figure 4-2 shows a configuration that would be adequate for this network.

Figure 4-2 Small-to-Medium LAN Configuration

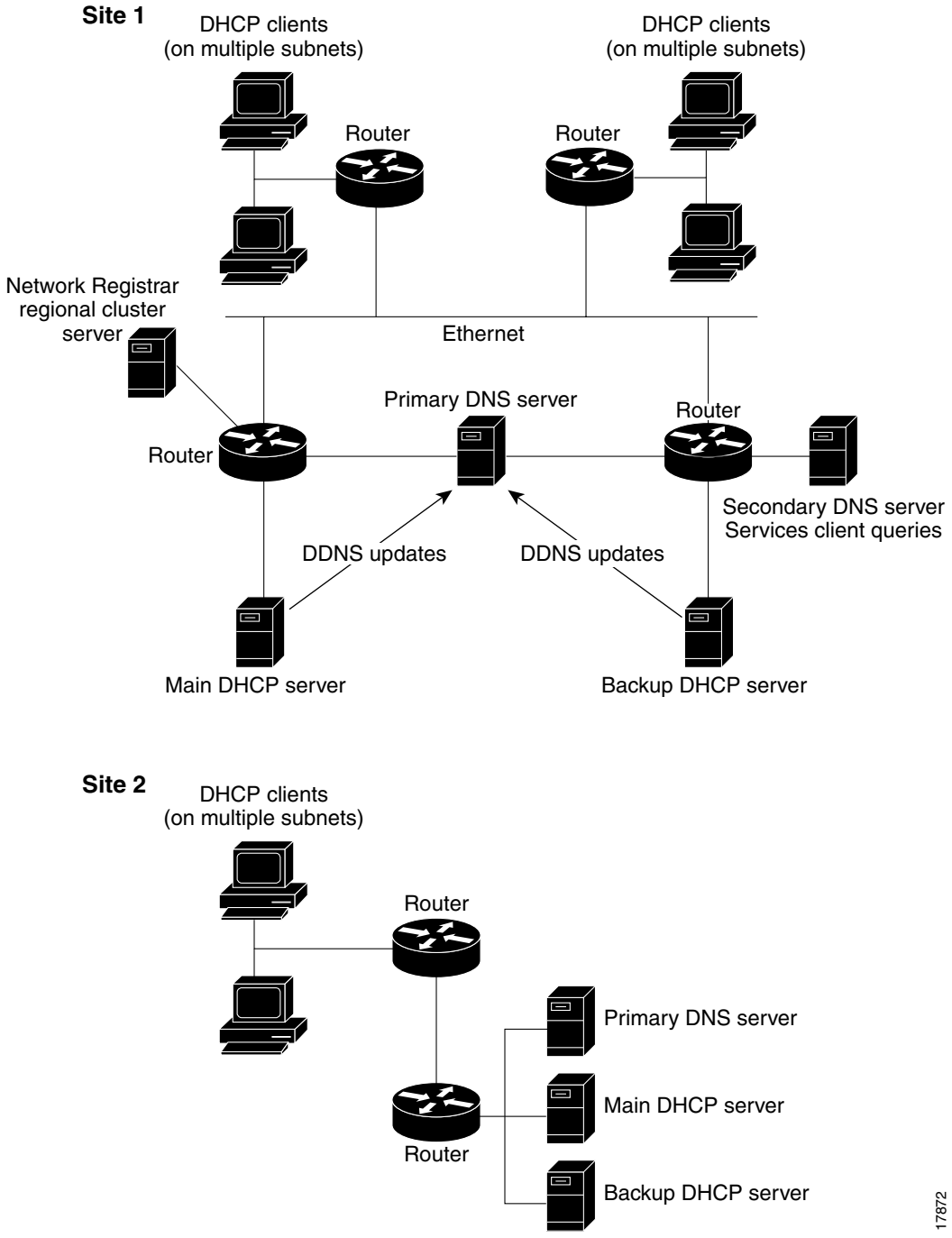


Large Enterprise and Service Provider Networks

In a large enterprise or service provider network serving over 500,000 DHCP clients, use mid-range Sun, Windows, or Linux servers. Put DNS and DHCP servers on different systems. Figure 4-3 on page 4-4 shows the hardware that would be adequate for this network.

When supporting geographically dispersed clients, locate DHCP servers at remote locations to avoid disrupting local services if wide-area connections fail. Install the Network Registrar regional cluster to centrally manage the distributed clusters.

Figure 4-3 Large Enterprise or Service Provider Network Configuration



Configuration and Performance Guidelines

Network Registrar is an integrated DHCP, DNS, and TFTP server cluster capable of running on a Windows 2003, Solaris, or Linux workstation or server.

Because of the wide range of network topologies for which you can deploy Network Registrar, you should first consider the following guidelines. These guidelines are very general and cover most cases. Specific or challenging implementations could require additional hardware or servers.

See Also

General Configuration Guidelines
Special Configuration Cases, page 4-6

General Configuration Guidelines

The following suggestions apply to most Network Registrar deployments:

- Configure a separate DHCP server to run in remote segments of the wide area network (WAN).

  Ensure that the DHCP client can consistently send a packet to the server in under a second. The DHCP protocol dictates that the client receive a response to a DHCPDISCOVER or DHCPREQUEST packet within four seconds of transmission. Many clients (notably early releases of the Microsoft DHCP stack) actually implement a two-second timeout.

- In large deployments, separate the secondary DHCP server from the primary DNS server used for dynamic DNS updates.

  Because lease requests and dynamic DNS updates are persisted to disk, server performance is impacted when using a common disk system. So that the DNS server is not adversely affected, run it on a different cluster than the DHCP server.

- Include a time server in your configuration to deal with time differences between the local and regional clusters so that aggregated data at the regional server appears in a consistent way. See the "Polling Subnet Utilization and Lease History Data" section on page 6-9.

- Set DHCP lease times in policies to four to ten days.

  To prevent leases from expiring when the DHCP client is turned off (overnight or over long weekends), set the DHCP lease time longer than the longest period of expected downtime, such as seven days. See Chapter 22, "Managing Leases."

- Locate backup DNS servers on separate network segments.

  DNS servers are redundant by nature. However, to minimize client impact during a network failure, ensure that primary and secondary DNS servers are on separate network segments.

- If there are high dynamic DNS update rates in the network, configure separate DNS servers for forward and reverse zones.

- Use NOTIFY/IXFR.

  Secondary DNS servers can receive their data from the primary DNS server in two ways: through a full zone transfer (AXFR) or an incremental zone transfer (NOTIFY/IXFR, as described in RFCs 1995 and 1996). Use NOTIFY/IXFR in environments where the name space is relatively dynamic. This reduces the number of records transferred from the primary to the secondary server. See the "Enabling Incremental Zone Transfers (IXFR)" section on page 17-9.

Special Configuration Cases

The following suggestions apply to some special configurations:

- When using dynamic DNS updates for large deployments or very dynamic networks, divide primary and secondary DNS and DHCP servers across multiple clusters.

  Dynamic DNS updates generate an additional load on all Network Registrar servers as new DHCP lease requests trigger dynamic DNS updates to primary servers that update secondary servers through zone transfers.

- During network reconfiguration, set DHCP lease renewal times to a small value.

  Do this several days before making changes in network infrastructure (such as to gateway router and DNS server addresses). A renewal time of eight hours ensures that all DHCP clients receive a changed DHCP option parameter within one working day. See Chapter 22, "Managing Leases."

Interoperability with Earlier Releases

Table 4-1 shows the interoperability of Network Registrar features on the regional CCM server with versions of the local cluster.

The following Red Hat (RH) Linux compatibility issues exist among Network Registrar releases:

- Release 6.1.*x* supports RH 7.3, RH ES 2.1, and RH ES 3.0:
  - The linux download kit supports RH 7.3 and RH ES 2.1
  - The linux3 download kit supports RH ES 3.0 for Release 6.1.2 and later
- Release 6.2.*x* supports RH ES 3.0 and RH ES 4.0:
  - The linux3 download kit supports RH ES 3.0
  - The linux4 download kit supports RH ES 4.0
- Release 6.3 and later supports RH ES 4.0 only.

Table 4-1 CCM Regional Feature Interoperability with Server Versions

	Local Cluster Version					
Feature	**6.0**	**6.1**	**6.1.1**	**6.2**	**6.3**	**7.0**
Push and pull:						
Address space	x	x	x	x	x	x
IPv6 address space						x
Scope templates, policies, client-classes	x	x	x	x	x	x
IPv6 prefix and link templates						x
Zone data and templates	x	x	x	x	x	x
Groups, owners, regions	x	x	x	x	x	x
Resource records (RRs)	x	x	x	x	x	x
Local cluster restoration			x	x	x	x
Host administration			x	x	x	x
Extended host administration				x	x	x
Administrators and roles			x	x	x	x

Table 4-1 CCM Regional Feature Interoperability with Server Versions

	Local Cluster Version					
Feature	**6.0**	**6.1**	**6.1.1**	**6.2**	**6.3**	**7.0**
Administrator:						
Single sign-on		x	x	x	x	x
Password change			x	x	x	x
IP history reporting:						
Lease history		x	x	x	x	x
Detail lease history			x	x	x	x
Utilization reporting:						
Subnet utilization history		x	x	x	x	x
Subnet and scope utilization			x	x	x	x
IPv6 prefix utilization						x



PART 2

Local and Regional Administration



CHAPTER 5

Configuring Administrators

This chapter explains how to set up network administrators at the local and regional clusters. The chapter also includes local and regional cluster tutorials for many of the administration features.

Administrators, Groups, and Roles

The types of functions that network administrators can perform in Network Registrar are based on the roles assigned to them. Local and regional administrators can define these roles to provide granularity for the network administration functions. Network Registrar predefines a set of base roles that segment the administrative functions. From these base roles you can define further constrained roles that are limited to administering particular addresses, zones, and other network objects.

The mechanism to associate administrators with their roles is to place the administrators in groups that include these roles.

See Also

How Administrators Relate to Groups and Roles
Administrator Types, page 5-2
Roles, Subroles, and Constraints, page 5-2
Groups, page 5-4
Managing Administrators, page 5-5
Managing Groups, page 5-6
Managing Passwords, page 5-7

How Administrators Relate to Groups and Roles

There are three administrator objects in Network Registrar—administrator, group, and role:

- **Administrator**—An account that logs in and that, through its association with one or more administrator groups, can perform certain functions based on its assigned role or roles. At the local cluster, these functions are administering the local Central Configuration Management (CCM) server and databases, hosts, zones, address space, and DHCP. At the regional cluster, these functions administer the regional CCM server and databases, central configuration, and regional address space. An administrator must be assigned to at least one group to be effective.

  Adding administrators is described in the "Managing Administrators" section on page 5-5.

- **Group**—A grouping of roles. You must associate one or more groups with an administrator, and a group must be assigned at least one role to be usable. The predefined groups that Network Registrar provides map each role to a unique group.

  Adding groups is described in the "Managing Groups" section on page 5-6.

- **Role**—Defines the network objects that an administrator can manage and the functions that an administrator can perform. A set of predefined roles are created at installation, and you can define additional constrained roles. Some of the roles include subroles that provide further functional constraints.

  Adding roles is described in the "Managing Roles" section on page 5-6.

Administrator Types

There are two basic types of administrators: superusers and specialized administrators:

- **Superuser**—Administrator with unrestricted access to the web UI, CLI, and all features. This administrator type should be restricted to a few individuals. Network Registrar includes one default superuser administrator (admin) with the password changeme. It appears by default in the list of administrators in the web UI (see Figure 5-2 on page 5-9). The superuser privileges of an administrator override all its other roles.

Tip Immediately change the default password after installing Network Registrar.

- **Specialized**—Administrator created by name to fulfill specialized functions, for example, to administer a specific DNS forward or reverse zone, based on the administrator assigned role (and subrole, if applicable). Specialized administrators, like the superuser, require a password, but must also be assigned at least one administrator group that defines the relevant roles. The entry fields in the web UI appear in Figure 5-2 on page 5-9. The CLI provides the **admin** command.

  For an example of creating a local zone or host administrator, see the "Create the Administrators" section on page 5-16.

Roles, Subroles, and Constraints

You can limit an administrator role by applying constraints. For example, you can use the host-admin base role to create a host administrator, named 192.168.50-host-admin, who is constrained to the 192.168.50.0 subnet. The administrator assigned a group that includes this role then logs in with this constraint in effect. Adding roles and subroles is described in the "Managing Roles" section on page 5-6.

You can further limit the constraints on roles to read-only access. An administrator can be allowed to read any of the data for that role, but not modify it. However, if read-write mode applies from any associated role, it supersedes the read-only constraints from all the other roles.

Tip An example of adding role constraints is in the "Create a Host Administrator Role with Constraints" section on page 5-21.

The interplay between DNS and host administrator role assignments is such that you can combine an unconstrained dns-admin role with any host-admin role in a group. For example, combining the dns-admin-readonly role and a host-admin role in a group (and naming the group host-rw-dns-ro) provides full host access and read-only access to zones and RRs. However, if you assign a constrained

dns-admin role along with a host-admin role to a group and then to an administrator, the constrained dns-admin role takes precedence, and the administrator privileges at login will preclude any host administration.

Certain roles provide subroles with which you can further limit the role functionality. For example, the local ccm-admin or regional-admin, with just the owner-region subrole applied, can manage only owners and regions. By default, all the possible subroles apply when you create a constrained role.

The predefined roles are described in Table 5-1 (local), and Table 5-2 on page 5-4 (regional).

Table 5-1 Local Cluster Administrator Predefined and Base Roles

Local Role	Subroles and Active Functionality
addrblock-admin	Core functionality: Manage address block, subnets, and reverse DNS zones (also requires dns-admin); and notify of scope activity. • *ric-management*: Push to, and reclaim subnets from, DHCP failover pairs and routers. • *ipv6-management*: Manage IPv6 prefixes, links, options, leases, and reservations.
ccm-admin	Core functionality: Manage licenses, access control lists (ACLs), and encryption keys. • *authentication*: Manage administrators. • *authorization*: Manage roles and groups. • *owner-region*: Manage owners and regions. • *database*: View database change entries and trim the CCM change sets.
cfg-admin	Core functionality: Manage clusters. • *ccm-management*: Manage the CCM server configuration. • *dhcp-management*: Manage the DHCP server configuration. • *dns-management*: Manage the DNS server configuration. • *ric-management*: Manage routers. • *snmp-management*: Manage the SNMP server configuration. • *tftp-management*: Manage the TFTP server configuration.
dhcp-admin	Core functionality: Manage DHCP scopes and templates, policies, clients, client-classes, options, leases, and reservations. • *server-management*: Manage the DHCP server configuration, failover pairs, LDAP servers, extensions, and statistics. • *ipv6-management*: Manage IPv6 prefixes, links, options, leases, and reservations.
dns-admin	Core functionality: Manage DNS zones and templates, resource records, secondary servers, and hosts. • *security-management*: Manage DNS update policies, ACLs, and encryption keys. • *server-management*: Manage DNS server configurations and zone distributions, synchronize zones and HA server pairs, and push update maps. • *ipv6-management*: Manage IPv6 zones and hosts.

Table 5-1 Local Cluster Administrator Predefined and Base Roles (continued)

Local Role	Subroles and Active Functionality
host-admin	Core functionality: Manage DNS hosts. (Note that if an administrator is also assigned a constrained dns-admin role that overrides the host-admin definition, the administrator is not assigned the host-admin role.)

Table 5-2 Regional Cluster Administrator Predefined and Base Roles

Regional Role	Subroles and Active Functionality
central-cfg-admin	Core functionality: Manage clusters and view replica data. • *dhcp-management*: Manage DHCP scope templates, policies, client-classes, failover pairs, virtual private networks (VPNs), and options; modify subnets; and replicate data. • *ric-management*: Manage routers and router interfaces, and pull replica router data.
central-dns-admin	Core functionality: Manage DNS zones and templates, hosts, resource records, and secondary servers; and create subzones and reverse zones. • *security-management*: Manage DNS update policies, ACLs, and encryption keys. • *server-management*: Synchronize DNS zones and HA server pairs, manage zone distributions, pull replica zone data, and push update maps.
central-host-admin	Core functionality: Manage DNS hosts. (Note that if an administrator is also assigned a constrained central-dns-admin role that overrides the central-host-admin definition, the administrator is not assigned the central-host-admin role.)
regional-admin	Core functionality: Manage licenses and encryption keys. • *authentication*: Manage administrators. • *authorization*: Manage roles and groups. • *owner-region*: Manage owners and regions. • *database*: View database change entries and trim the CCM change sets.
regional-addr-admin	Core functionality: Manage address blocks, subnets, and address ranges; generate allocation reports; and pull replica address space data. • *dhcp-management*: Push and reclaim subnets; and add subnets to, and remove subnets from, DHCP failover pairs. • *lease-history*: Query, poll, and trim lease history data. • *subnet-utilization*: Query, poll, trim, and compact subnet utilization data.

Groups

Administrator groups are the mechanism used to assign roles to administrators. Hence, a group must consist of one or more administrator roles to be usable. When you first install Network Registrar, a predefined group is created to correspond to each predefined role.

When a group is assigned multiple roles, constraints are defined as a sum of all the roles. For example, if one of the roles is assigned unconstrained read-write privileges, the group is assigned unconstrained read-write privileges, even though other roles might be assigned read-only privileges. Therefore, to limit the read-write privileges of a user while allowing read-only access to all data, create a group that includes the unconstrained read-only role along with a constrained read-write role. (See the "Roles, Subroles, and Constraints" section on page 5-2 for the implementation of host-admin and dns-admin roles combined in a group.)

Note Upgrading from Network Registrar 6.0 or 6.1 does not create groups for each predefined role. However, groups are created for administrators that had direct role assignments in the earlier releases. These group names are the original role names appended with **–group** (and a number if there already is a group by that name).

Managing Administrators

The Network Registrar web UIs have only one predefined administrator, the admin account. This superuser can exercise all the functions of the web UI and usually adds the other key administrators. However, ccm-admin and regional-admin administrators can also add, edit, and delete administrators. Creating an administrator requires:

- Adding its name.
- Adding a password.
- Specifying if the administrator should have superuser privileges (usually assigned on an extremely limited basis).
- If not creating a superuser, specifying the group or groups to which the administrator should belong. These groups should have the appropriate role (and possibly subrole) assignments, thereby setting the proper constraints.

Tip If you accidentally delete all the roles by which you can log in to Network Registrar (those having superuser, ccm-admin, or regional-admin privileges), you can recover by creating a username/password pair in the *install-path*/conf/priv/local.superusers file. You must create this file, have write access to it, and include a line in it with the format *username password*. After creating the file, stop and restart the Network Registrar server agent. Use this username and password for the next login session. Note, however, that using the local.superusers file causes reduced security. Therefore, use this file only in emergencies such as when temporarily losing all login access. Once logged in, create a superuser account in the usual way, then delete the local.superusers file or its contents.

Local and Regional Web UI

Click **Administration**, then **Administrators**. This opens the List/Add Administrators page (see the "Create the Administrators" section on page 5-16 for an example). Enter a name and password, and choose one or more existing groups from the drop-down list (or whether the administrator should be a superuser), then click **Add Administrator**.

Edit an administrator by clicking its name on the List/Add Administrators page, then modifying the name, password, superuser status, or group membership on the Edit Administrator page. The active group or groups should be in the Selected list. Click **Modify Administrator**.

To delete an administrator, click the Delete icon ( ) next to the name, then confirm or cancel the deletion.

CLI Commands

List the administrators by using **admin list** or **admin listnames**. Add administrators by using **admin** *name* **create** (see the **admin** command in the CLIGuide.html file in the /docs directory for syntax and attribute descriptions). Delete an administrator by using **admin** *name* **delete**.

Managing Groups

A superuser, ccm-admin, or regional-admin can create, edit, and delete administrator groups. Creating an administrator group involves:

- Adding its name.
- Adding an optional description.
- Choosing an associated role.

Local Advanced and Regional Web UI

Click **Administration**, then **Groups**. This opens the List/Add Administrator Groups page (see the "Create a Group to Assign to the Host Administrator" section on page 5-23 for an example). (You cannot manage groups in local Basic mode.) Enter a name and optional description, and choose one or more existing roles from the drop-down list, then click **Add Group**.

Edit a group by clicking its name on the List/Add Administrator Groups page, then modifying the name, description, or role membership on the Edit Administrator Group page. The active roles should be in the Selected list. Click **Modify Group**.

To delete a group, click the Delete icon () next to the name, then confirm or cancel the deletion.

CLI Commands

To add groups, use **group** *name* **create** (see the **group** command in the CLIGuide.html file in the /docs directory for syntax and attribute descriptions).

Managing Roles

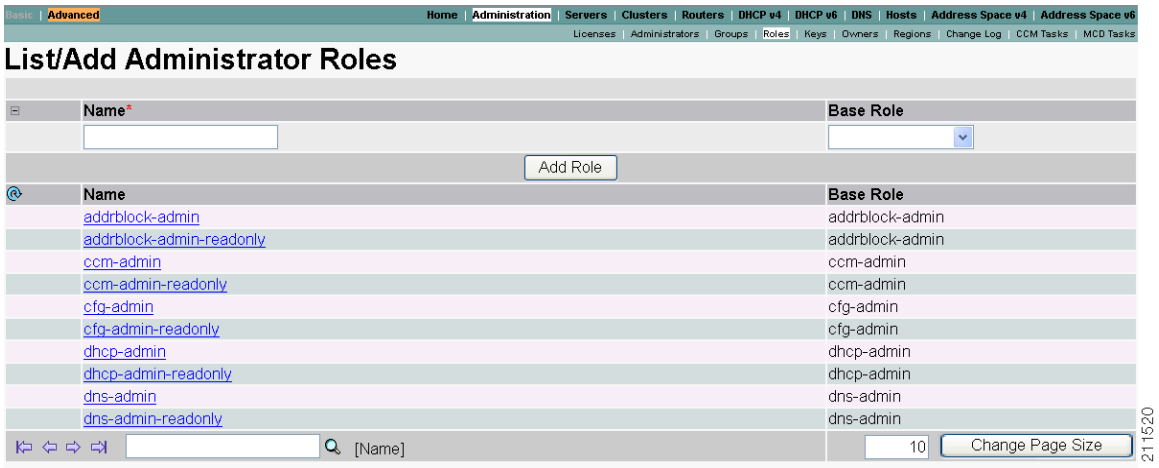
A superuser, ccm-admin, or regional-admin administrator can create, edit, and delete administrator roles. Creating an administrator role involves:

- Adding its name.
- Choosing a base role.
- Possibly specifying if the role should be constrained, unconstrained, or read-only.
- Possibly adding constraints.
- Possibly assigning groups.

Local Advanced and Regional Web UI

Click **Administration**, then **Roles**. This opens the List/Add Administrator Roles page (see Figure 5-1 on page 5-7). (You cannot manage roles in Basic mode.)

Figure 5-1 List/Add Administrator Roles Page (Local Advanced)



Enter a name and choose a base role from the drop-down list for the role, then click **Add Role**. The next page that opens varies based on the base role you choose for the role (for an example, see the "Create a Host Administrator Role with Constraints" section on page 5-21). On the Add *xxx* Administrator Role page, specify any role constraints, subrole restrictions, or group selections, then click **Add Role**.

Edit a role by clicking its name on the List/Add Administrator Roles page, then modifying the name or any constraints, subrole restrictions, or group selections on the Edit *xxx* Administrator Role page. The active subroles or groups should be in the Selected list. Click **Modify Role**.

To delete a role, click the Delete icon ( ) next to the name, then confirm or cancel the deletion.

CLI Commands

To add and edit administrator roles, use **role** *name* **create** *base-role* (see the **role** command in the CLIGuide.html file in the /docs directory for syntax and attribute descriptions). The base roles have default groups associated with them. To add other groups, set the *groups* attribute (a comma-separated string value).

Managing Passwords

Passwords are key to administrator access to the web UI and CLI. In the web UI, you enter the password on the Login page. In the CLI, you enter the password when you first invoke the **nrcmd** program. The local or regional CCM administrator or superuser can change any administrator password.

You can prevent exposing a password on entry. In the web UI, logging in or adding a password never exposes it on the page, except as asterisks. In the CLI, you can prevent exposing the password by creating an administrator, omitting the password, then using **admin** *name* **enterPassword**, where the prompt displays the password as asterisks. You can do this instead of the usual **admin** *name* **set password** command that exposes the password as plain text.

Administrators can change their own passwords on clusters. If you want the password change propagated from the regional to local clusters, you must change the password at the regional cluster, then use the regional administrator push function to push the new setting to the other clusters (see the "Pushing and Pulling Administrators" section on page 5-9). You must have regional CCM administrator or superuser privileges; otherwise, you must have the regional administrator do it for you.

Licensing

Regional and local cluster operations require a feature **ip-node** license and possibly incremental additional node licenses. See the "Logging In to the Web UIs" section on page 2-2 for entering license data the first time you try to log in. You can add the additional ip-node licenses once you log in.

Local and Regional Web UI

Click **Administration**, then **Licenses** to open the List/Add Product Licenses page. Click **Browse** to locate the license file, click the file, then click **Open**. If the license ID in the file is valid, the license key appears in the list of licenses with the message "Successfully added license file "*filename*." If the ID is not valid, the License field shows the contents of the file and the message "Object is invalid" appears.

The License Utilization section at the bottom of the page lists the type of license, the number of nodes allowed for the license, and the actual number of nodes used. Expand the section by clicking the plus (+) sign.

CLI Commands

Use **license** *file* **create** to create a license. The file referenced should include its absolute path or path relative to where you execute the commands. For example:

```
nrcmd> license "C:\licenses\ipnode-1.lic" create
```

Use **license list** to list the properties of all the created licenses (identified by key), and **license listnames** to list just the keys. Use **license** *key* **show** to show the properties of a specific license key.

Centrally Managing Administrators

As a regional or local CCM administrator, you can:

- Create and modify local and regional cluster administrators, groups, and roles.
- Push administrators, groups, roles, owners, and regions to local clusters.
- Pull local cluster administrators, groups, roles, owners, and regions to the central cluster.

Each of these functions involves having at least one regional CCM administrator subrole defined. Table 5-3 describes the subroles required for these operations.

Table 5-3 Subroles Required for Central Administrator Management

Central Administrator Management Action	Required Regional Subroles
Create, modify, push, pull, or delete administrators	authentication
Create, modify, push, pull, or delete groups or roles	authorization
Create, modify, pull, push, or delete owners or regions	owner-region
Create, modify, push, pull, or delete groups or roles with associated owners or regions	authorization owner-region

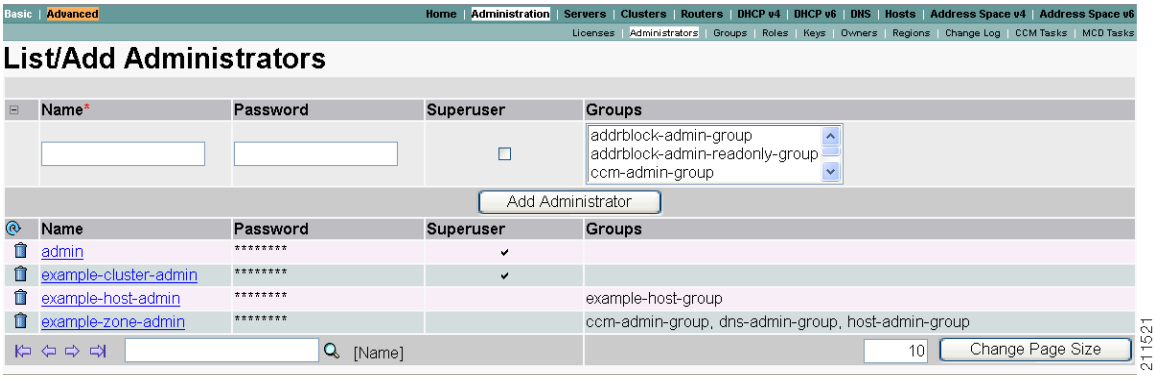
See Also

Pushing and Pulling Administrators
Pushing and Pulling Groups, page 5-11
Pushing and Pulling Roles, page 5-12
Pushing and Pulling Owners or Regions, page 5-14

Pushing and Pulling Administrators

You can push administrators to, and pull administrators from, local clusters on the List/Add Administrators page in the regional cluster web UI (see Figure 5-2).

Figure 5-2 List/Add Administrators Page (Regional)



You can create administrators with both local and regional roles at the regional cluster. However, you can push or pull only associated local roles, because local clusters do not recognize regional roles.

See Also

Pushing Administrators to Local Clusters
Pulling Administrators from the Replica Database, page 5-10

Pushing Administrators to Local Clusters

Pushing administrators to local clusters involves choosing one or more clusters and a push mode.

Regional Web UI

Step 1 Click **Administration**, then **Administrators**.

Step 2 On the List/Add Administrators Page, click **Push All Administrators** to push all the administrators listed on the page, or **Push Admin** next to an individual administrator. This opens the Push Administrator Data to Local Clusters page.

Step 3 Choose a push mode by clicking one of the Data Synchronization Mode radio buttons. If you are pushing all the admiLnistrators, you can choose Ensure, Replace, or Exact. If you are pushing a single administrator, you can choose Ensure or Replace. In both cases, Ensure is the default mode. You would

choose Replace only if you want to replace the existing administrator data at the local cluster. You would choose Exact only if you want to create an exact copy of the administrator database at the local cluster, thereby deleting all administrators that are not defined at the regional cluster.

Step 4 Choose one or more local clusters in the Available field of the Destination Clusters and move it or them to the Selected field.

Step 5 Click **Push Data to Clusters**.

Step 6 On the View Push Administrator Data Report page, view the push details, then click **OK** to return to the List/Add Administrators page.

Step 7 To confirm that administrators are pushed successfully, click **Clusters**, then **Cluster Tree** to open the View Tree of Server Clusters page. Click the Go Local icon () next to the cluster name to open the web UI for the local cluster, then verify that the administrator or administrators are added to the cluster.

Pulling Administrators from the Replica Database

Pulling administrators from the local clusters is mainly useful only in creating an initial list of administrators that can then be pushed to other local clusters. The local administrators are not effective at the regional cluster itself, because these administrators do not have regional roles assigned to them.

When you pull an administrator, you are actually pulling it from the regional cluster replica database. Creating the local cluster initially replicates the data, and periodic polling automatically updates the replication. However, to ensure that the replica data is absolutely current with the local cluster, you can force an update before pulling the data.

Regional Web UI

Step 1 Click **Administration**, then **Administrators**.

Step 2 On the List/Add Administrators Page, click **Pull Replica Administrators**. This opens the Select Replica Administrator Data to Pull page.

Step 3 Click the Replicate icon () in the Update Replica Data column for the cluster. (For the automatic replication interval, see the "Replicating Local Cluster Data" section on page 6-7.)

Step 4 Choose a replication mode using one of the Mode radio buttons. In most cases, you would leave the default Replace mode enabled, unless you want to preserve any existing administrator properties already defined at the regional cluster by choosing Ensure, or create an exact copy of the administrator database at the local cluster by choosing Exact (not recommended).

Step 5 Click **Pull All Administrators** next to the cluster, or expand the cluster name and click **Pull Administrator** to pull an individual administrator in the cluster.

Step 6 On the Report Pull Replica Administrators page, view the pull details, then click **Run**.

Step 7 On the Run Pull Replica Administrators page, view the change set data, then click **OK**. You return to the List/Add Administrators page with the pulled administrators added to the list.

Note If you do not have a regional cluster and would like to copy administrators, roles, or groups from one local cluster to another, you can export them and then reimport them at the target cluster by using the **cnr_exim** tool (see the "Using the cnr_exim Data Import and Export Tool" section on page 8-11).

However, the tool does not preserve the administrator passwords, and you must manually reset them at the target cluster. It is implemented this way to maintain password security. The export command is:

```
cnr_exim -c admin -x -e outputfile.txt
```

Pushing and Pulling Groups

Pushing and pulling groups is vital in associating administrators with a consistent set of roles at the local clusters. You can push groups to, and pull groups from, local clusters on the List/Add Administrator Groups page in the regional cluster web UI (see Figure 5-10 on page 5-23).

See Also

Pushing Groups to Local Clusters
Pulling Groups from the Replica Database, page 5-12

Pushing Groups to Local Clusters

Pushing groups to local clusters involves choosing one or more clusters and a push mode.

Regional Web UI

Step 1 Click **Administration**, then **Groups**.

Step 2 On the List/Add Administrator Groups page, click **Push All Groups** to push all the groups listed on the page, or **Push Group** next to an individual group. This opens the Push Group Data to Local Clusters page.

Step 3 Choose a push mode using one of the Data Synchronization Mode radio buttons. If you are pushing all the groups, you can choose Ensure, Replace, or Exact. If you are pushing a single group, you can choose Ensure or Replace. In both cases, Ensure is the default mode. You would choose Replace only if you want to replace the existing group data at the local cluster. You would choose Exact only if you want to create an exact copy of the group data at the local cluster, thereby deleting all groups that are not defined at the regional cluster.

Step 4 By default, the associated roles and owners are pushed along with the group. Roles are pushed in Replace mode and owners in Ensure mode. To disable pushing the associated roles or owners, remove the respective check mark.

Step 5 Choose one or more local clusters in the Available field of the Destination Clusters and move it or them to the Selected field.

Step 6 Click **Push Data to Clusters**.

Step 7 On the View Push Group Data Report page, view the push details, then click **OK** to return to the List/Add Administrator Groups page.

Step 8 To confirm that groups are pushed successfully, click **Clusters**, then **Cluster Tree** to open the View Tree of Server Clusters page. Click the Go Local icon () next to the cluster name to open the web UI for the local cluster, then verify that the group or groups are added to the cluster.

Pulling Groups from the Replica Database

Pulling administrator groups from the local clusters is mainly useful only in creating an initial list of groups that can then be pushed to other local clusters. The local groups are not useful at the regional cluster itself, because these groups do not have regional roles assigned to them.

When you pull a group, you are actually pulling it from the regional cluster replica database. Creating the local cluster initially replicates the data, and periodic polling automatically updates the replication. However, to ensure that the replica data is absolutely current with the local cluster, you can force an update before pulling the data.

Regional Web UI

Step 1 Click **Administration**, then **Groups**.

Step 2 On the List/Add Administrator Groups page, click **Pull Replica Groups**. This opens the Select Replica Group Data to Pull page.

Step 3 Click the Replica icon ( ) in the Update Replica Data column for the cluster. (For the automatic replication interval, see the "Replicating Local Cluster Data" section on page 6-7.)

Step 4 Choose a replication mode using one of the Mode radio buttons. In most cases, you would leave the default Replace mode enabled, unless you want to preserve any existing group properties at the local cluster by choosing Ensure, or create an exact copy of the group data at the local cluster by choosing Exact (not recommended).

Step 5 Click **Pull All Groups** next to the cluster, or expand the cluster name and click **Pull Group** to pull an individual group in the cluster.

Step 6 On the Report Pull Replica Groups page, view the pull details, then click **Run**.

Step 7 On the Run Pull Replica Groups page, view the change set data, then click **OK**. You return to the List/Add Administrator Groups page with the pulled groups added to the list.

Pushing and Pulling Roles

You can push roles to, and pull roles from, local clusters on the List/Add Administrator Roles page in the regional cluster web UI (see Figure 5-1 on page 5-7). You can also push associated groups and owners, and pull associated owners, depending on your subrole permissions (see Table 5-3 on page 5-8).

See Also

Pushing Roles to Local Clusters, page 5-13
Pulling Roles from the Replica Database, page 5-13

Pushing Roles to Local Clusters

Pushing administrator roles to local clusters involves choosing one or more clusters and a push mode.

Regional Web UI

Step 1 Click **Administration**, then **Roles**.

Step 2 On the List/Add Administrator Roles page, click **Push All Roles** to push all the roles listed on the page, or **Push Role** next to an individual role. This opens the Push Role Data to Local Clusters page.

Step 3 Choose a push mode using one of the Data Synchronization Mode radio buttons. If you are pushing all the roles, you can choose Ensure, Replace, or Exact. If you are pushing a single role, you can choose Ensure or Replace. In both cases, Ensure is the default mode. You would choose Replace only if you want to replace the existing role data at the local cluster. You would choose Exact only if you want to create an exact copy of the role data at the local cluster, thereby deleting all roles that are not defined at the regional cluster.

Step 4 By default, the associated groups and owners are pushed along with the role. Groups are pushed in Replace mode and owners in Ensure mode. To disable pushing the associated roles or owners, remove the respective check mark:

- If you disable pushing associated groups and the group does not exist at the local cluster, a group based on the name of the role is created at the local cluster.
- If you disable pushing associated owners and the owner does not exist at the local cluster, the role will not be configured with its intended constraints. You must separately push the group to the local cluster, or ensure that the regional administrator assigned the owner-region subrole has pushed the group before pushing the role.

Step 5 Choose one or more local clusters in the Available field of the Destination Clusters and move it or them to the Selected field.

Step 6 Click **Push Data to Clusters**.

Step 7 On the View Push Role Data Report page, view the push details, then click **OK** to return to the List/Add Administrator Roles page.

Step 8 To confirm that roles are pushed successfully, click **Clusters**, then **Cluster Tree** to open the View Tree of Server Clusters page. Click the Go Local icon () next to the cluster name to open the web UI for the local cluster, then verify that the role or roles are added to the cluster.

Pulling Roles from the Replica Database

Pulling administrator roles from the local clusters is mainly useful only in creating an initial list of roles that can then be pushed to other local clusters. The local roles are not useful at the regional cluster itself.

When you pull a role, you are actually pulling it from the regional cluster replica database. Creating the local cluster initially replicates the data, and periodic polling automatically updates the replication. However, to ensure that the replica data is absolutely current with the local cluster, you can force an update before pulling the data.

Regional Web UI

Step 1 Click **Administration**, then **Roles**.

Step 2 On the List/Add Administrator Roles page, click **Pull Replica Roles**. This opens the Select Replica Role Data to Pull page.

Step 3 Click the Replicate icon ( ) in the Update Replica Data column for the cluster. (For the automatic replication interval, see the "Replicating Local Cluster Data" section on page 6-7.)

Step 4 Choose a replication mode using one of the Mode radio buttons. In most cases, you would leave the default Replace mode enabled, unless you want to preserve any existing role properties at the local cluster by choosing Ensure, or create an exact copy of the role data at the local cluster by choosing Exact (not recommended).

Step 5 If you have the owner-region subrole permission, you can decide if you want to pull all the associated owners with the role, which is always in Ensure mode. This choice is enabled by default.

Step 6 Click **Pull All Roles** next to the cluster, or expand the cluster name and click **Pull Role** to pull an individual role in the cluster.

Step 7 On the Report Pull Replica Roles page, view the pull details, then click **Run**.

Step 8 On the Run Pull Replica Roles page, view the change set data, then click **OK**. You return to the List/Add Administrator Roles page with the pulled roles added to the list.

Pushing and Pulling Owners or Regions

You can push owners or regions to, and pull them from, local clusters on the List/Add Owners page or List/Add Regions page, respectively, in the regional cluster web UI.

See Also

Pushing Owners or Regions to Local Clusters
Pulling Owners and Regions from the Replica Database, page 5-15

Pushing Owners or Regions to Local Clusters

Pushing owners or regions to local clusters involves choosing one or more clusters and a push mode.

Regional Web UI

Step 1 Click **Administration**, then **Owners** or **Regions**.

Step 2 On the List/Add Owners or List/Add Regions page, click **Push All Owners** or **Push All Regions** to push all the owners or regions listed on the page, or **Push Owner** or **Push Region** next to an individual owner or region. This opens the Push Owner Data to Local Clusters or Push Owner Data to Local Clusters page.

Step 3 Choose a push mode using one of the Data Synchronization Mode radio buttons. If you are pushing all the owners or regions, you can choose Ensure, Replace, or Exact. If you are pushing a single owner or region, you can choose Ensure or Replace. In both cases, Ensure is the default mode. You would choose Replace only if you want to replace the existing owner or region data at the local cluster. You would choose Exact only if you want to create an exact copy of the owner or region data at the local cluster, thereby deleting all owners or regions that are not defined at the regional cluster.

Step 4 Choose one or more local clusters in the Available field of the Destination Clusters and move it or them to the Selected field.

Step 5 Click **Push Data to Clusters**.

Step 6 On the View Push Owner Data Report or View Push Region Data Report page, view the push details, then click **OK** to return to the List/Add Owners or List/Add Regions page.

Step 7 To confirm that owners or regions are pushed successfully, click **Clusters**, then **Cluster Tree** to open the View Tree of Server Clusters page. Click the Go Local icon () next to the cluster name to open the web UI for the local cluster, then verify that the owners or regions are added to the cluster.

Pulling Owners and Regions from the Replica Database

When you pull an owner or region, you are actually pulling it from the regional cluster replica database. Creating the local cluster initially replicates the data, and periodic polling automatically updates the replication. However, to ensure that the replica data is absolutely current with the local cluster, you can force an update before pulling the data.

Regional Web UI

Step 1 Click **Administration** in the regional cluster web UI, then **Owners** or **Regions**.

Step 2 On the List/Add Owners or List/Add Regions page, click **Pull Replica Owners** or **Pull Replica Regions**. This opens the Select Replica Owner Data to Pull or Select Replica Region Data to Pull page.

Step 3 Click the Replicate icon () in the Update Replica Data column for the cluster. (For the automatic replication interval, see the "Replicating Local Cluster Data" section on page 6-7.)

Step 4 Choose a replication mode using one of the Mode radio buttons. In most cases, you would leave the default Replace mode enabled, unless you want to preserve any existing owner properties at the local cluster by choosing Ensure, or create an exact copy of the owner data at the local cluster by choosing Exact (not recommended).

Step 5 Click **Pull All Owners** or **Pull All Regions** next to the cluster, or expand the cluster name and click **Pull Owner** or **Pull Region** to pull an individual owner or region in the cluster.

Step 6 On the Report Pull Replica Owners or Report Pull Replica Regions page, click **Run**.

Step 7 On the Run Pull Replica Owners or Run Pull Replica Region page, view the change set data, then click **OK**. You return to the List/Add Owners or List/Add Regions page with the pulled owners or regions added to the list.

Local Cluster Management Tutorial

This tutorial describes a basic scenario on a local cluster of the Example Company. Administrators at the cluster are responsible for users, zone data, DHCP data, address space data, and the servers in general. The task is to set up two zones (example.com and boston.example.com), hosts in the zones, and a subnet. The local cluster must also create a special administrator account so that the regional cluster in San Jose can perform the central configuration and replicate the local cluster administrators and address space at another cluster, as described in the "Regional Cluster Management Tutorial" section on page 5-25.

See Also

Administrator Responsibilities and Tasks
Create the Administrators
Create the Address Infrastructure, page 5-18
Create the Zone Infrastructure, page 5-19
Create a Host Administrator Role with Constraints, page 5-21
Create a Group to Assign to the Host Administrator, page 5-23
Test the Host Address Range, page 5-24

Administrator Responsibilities and Tasks

The local cluster administrators have the following responsibilities and tasks:

- **example-cluster-admin**—Created by the superuser:
  - At the Boston cluster, creates the other local administrators (example-zone-admin and example-host-admin).
  - Creates the basic network infrastructure for the local clusters.
  - Constrains the example-host-role to an address range in the boston.example.com zone.
  - Creates the example-host-group (defined with the example-host-role) that the example-zone-admin will assign to the example-host-admin.
- **example-zone-admin**:
  - Creates the example.com and boston.example.com zones, and maintains the latter zone.
  - Assigns the example-host-group to the example-host-admin.
- **example-host-admin**—Maintains local host lists and IP address assignments.

Create the Administrators

For this example, the superuser in Boston creates the local cluster, zone, and host administrators, as described in the "Administrator Responsibilities and Tasks" section.

Local Basic Web UI

Step 1 At the Boston local cluster, log in as superuser (usually **admin**).

Step 2 In Basic mode, click **Administration**, then **Administrators**.

Step 3 Add the local cluster administrator (with superuser access)—On the List/Add Administrators page:

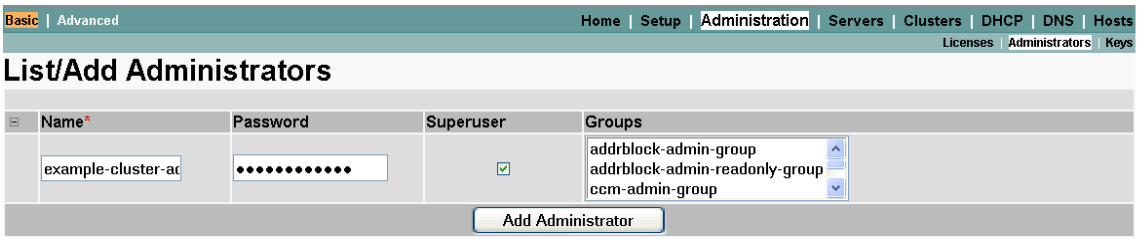
a. Enter **example-cluster-admin** in the Name field. Tab to the next field.

b. Enter **exampleadmin** in the Password field.

c. Click the Superuser check box to add a check mark.

d. Do not choose a group from the Groups list (see Figure 5-3 on page 5-17).

Figure 5-3 Adding a Local Cluster Administrator (Local Basic)



e. Click **Add Administrator**.

Step 4 Add the local zone administrator on the same page:

a. Enter **example-zone-admin** in the Name field, then **examplezone** in the Password field.

b. Multiselect **ccm-admin-group**, **dns-admin-group**, and **host-admin-group** in the Groups drop-down list. The dns-admin-group is already predefined with the dns-admin role to administer DNS zones and servers. The ccm-admin-group guarantees that the example-zone-admin can set up the example-host-admin with a constrained role later on. The host-admin-group is mainly to test host creation in the zone.

c. Click **Add Administrator**.

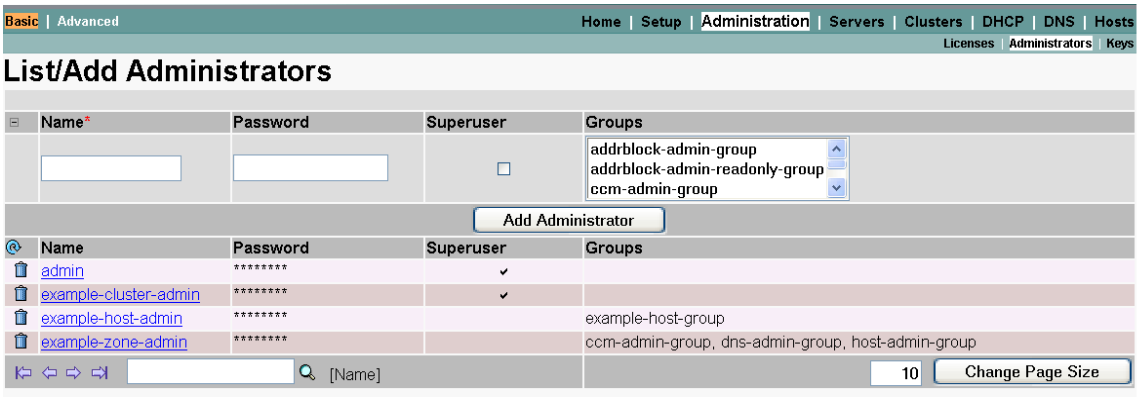
Step 5 Add the local host administrator on the same page:

a. Enter **example-host-admin** in the Name field, then **examplehost** in the Password field.

b. Do not choose a group at this point. (The example-zone-admin will later assign example-host-admin to a group with a constrained role.)

c. Click **Add Administrator**. The page should now appear as in Figure 5-4.

Figure 5-4 Listing Local Administrators (Local Basic)



Note For a description on how to apply constraints to the administrator, see the "Create a Host Administrator Role with Constraints" section on page 5-21.

Create the Address Infrastructure

A prerequisite to managing the zones and hosts at the clusters is to create the underlying network infrastructure. The network configuration often already exists and was imported. However, this tutorial assumes that you are starting with a clean slate.

The local example-cluster-admin next creates the allowable address ranges for the hosts in the boston.example.com zone that will be assigned static IP addresses. These addresses are in the 192.168.50.0/24 subnet with a range of hosts from 100 through 200.

Local Advanced Web UI

Step 1 At the local cluster, log out as superuser, then log in as the **example-cluster-admin** user with password **exampleadmin**. Because the administrator is a superuser, all features are available.

Step 2 Click **Advanced** to go to Advanced mode, then **Address Space**, then **Subnets**.

Step 3 On the List/Add Subnets page (see Figure 15-4 on page 15-13), enter the boston.example.com subnet address:

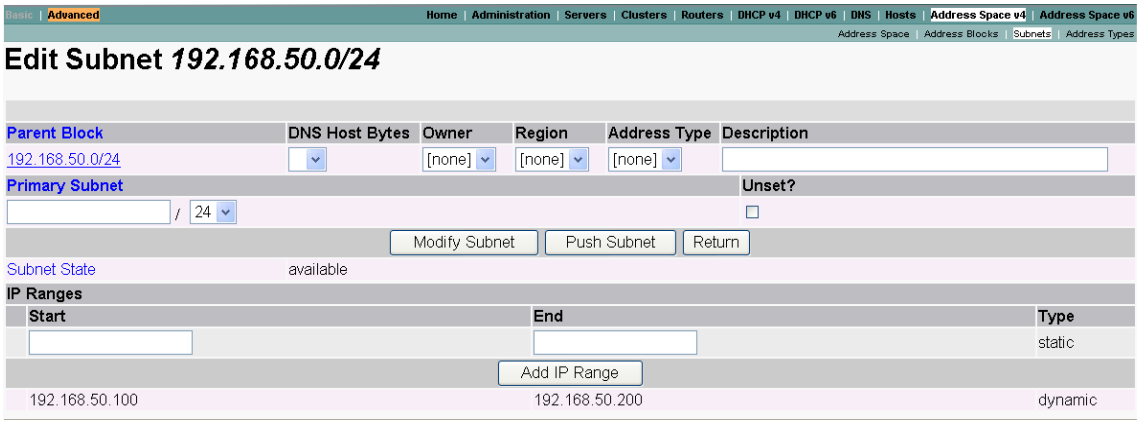
- **a.** In the Address/Mask field, enter **192.168.50**.
- **b.** Choose **24** in the mask drop-down list—This subnet will be a normal Class C network.
- **c.** Leave the Owner, Region, and Address Type fields as is. Add descriptive text if desired.
- **d.** Click **Add Subnet**.

Step 4 Click the 192.168.50.0/24 address to open the Edit Subnet page.

Step 5 In the IP Ranges fields, enter the static address range:

- **a.** Enter **100** in the Start field. Tab to the next field.
- **b.** Enter **200** in the End field.
- **c.** Click **Add IP Range**. The address range appears under the fields (see Figure 5-5). (Note that in the figure the IP address range shows as dynamic because in this example a scope with that address range was already defined; otherwise, it would show as a static range.)

Figure 5-5 Editing a Subnet to Add an IP Range (Local Advanced)



Step 6 Click **Modify Subnet**.

Step 7 Click **Address Space** to open the View Unified Address Space page. The 192.168.50.0/24 subnet should appear in the list. If not, click the Refresh icon ( ).

Create the Zone Infrastructure

For this scenario, example-cluster-admin must create the Example Company zones locally, including the example.com zone and its subzones. The example-cluster-admin also adds some initial host records to the boston.example.com zone.

See Also

Create the Forward Zones
Create the Reverse Zones, page 5-20
Create the Initial Hosts, page 5-21

Create the Forward Zones

First, create the example.com and boston.example.com forward zones.

Local Basic Web UI

Step 1 At the local cluster, log in as the **example-zone-admin** user with password **examplezone**.

Step 2 On the Main Menu page, in Basic mode, click the Manage Zones link to open the List/Add Zones page.

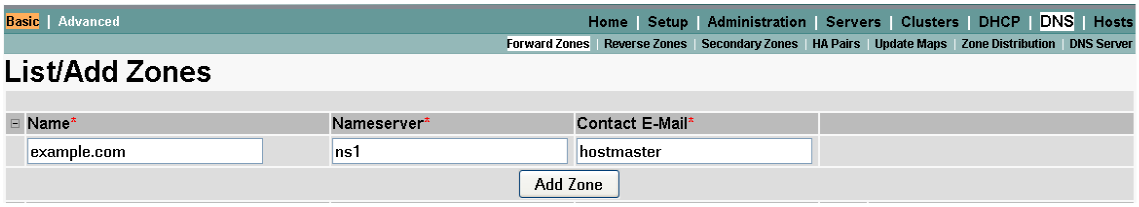
Step 3 Create the example.com zone (tab from field to field):

a. In the Name field, enter **example.com**.

b. In the Nameserver field, enter **ns1**.

c. In the Contact E-Mail field, enter **hostmaster** (see Figure 5-6).

Figure 5-6 Creating a Zone (Local Basic)



d. Click **Add Zone**.

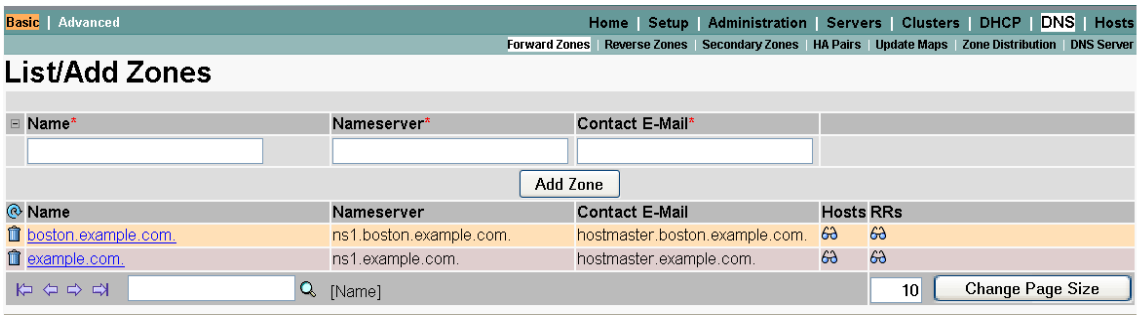
Step 4 Create the **boston.example.com** zone in the same way, using the same values as in the previous steps:

a. Creating a zone with a prefix added to an existing zone opens the Create Subzone in Parent Zone page, because the zone can be a potential subzone. Because you do want to create this zone as a subzone to example.com, click **Create as Subzone** on the Create Subzone in Parent Zone page.

b. Because nameservers are different in each zone, you must create a glue Address (A) record to tie the zones together. Enter 192.168.50.1 in the A record field, then click **Specify Glue Records**. Then click **Report**, **Run**, and **Return**.

c. The List/Add Zones page should now list example.com and boston.example.com (see Figure 5-7).

Figure 5-7 Viewing Created Zones (Local Basic)



Step 5 Click **Advanced**, then **Show Forward Zone Tree** to show the hierarchy of the zones. Return to list mode by clicking **Show Forward Zone List**.

Create the Reverse Zones

Next, create the reverse zones for example.com and boston.example.com. This way you can add reverse address pointer (PTR) records for each added host. The reverse zone for example.com is based on the 192.168.50.0 subnet; the reverse zone for boston.example.com is based on the 192.168.60.0 subnet.

Local Basic Web UI

Step 1 At the local cluster, you should be logged in as the example-zone-admin user, as in the previous section.

Step 2 In DNS mode, click the **Reverse Zones** submenu.

Step 3 On the List/Add Reverse Zones page, enter **50.168.192.in-addr.arpa** in the Name field. (There is already a reverse zone for the loopback address, 127.in-addr.arpa.)

Step 4 Click **Add Zone** to open the Add Reverse Zone page.

Step 5 Enter the required fields to create the reverse zone, using the forward zone values:

a. **Nameserver**—Enter **ns1.example.com.** (be sure to include the trailing dot).

b. **Contact E-Mail**—Enter **hostmaster.example.com.** (be sure to include the trailing dot).

Step 6 Click **Add Zone** to add the zone and return to the List/Add Reverse Zones page.

Step 7 Do the same for the boston.example.com zone, using **60.168.192.in-addr.arpa** as the zone name and the same nameserver and contact e-mail values as in Step 5. (You can cut and paste the values from the table.)

Create the Initial Hosts

As a confirmation that hosts can be created at the Boston cluster, the example-zone-admin tries to create two hosts in the example.com zone.

Local Advanced Web UI

Step 1 As the example-zone-admin user, click **Advanced** to enter Advanced mode.

Step 2 Click Hosts to open the List Zones page. You should see boston.example.com and example.com.

Step 3 Click example.com. in the list of zones.

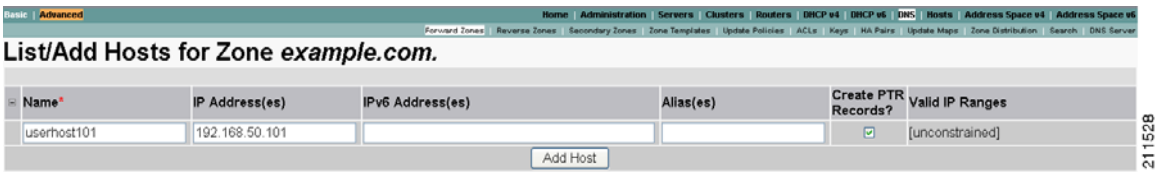
Step 4 On the List/Add Hosts for Zone page, add the first static host with address 192.168.50.101:

a. Enter **userhost101** in the Name field.

b. Enter the complete address **192.168.50.101** in the IP Address(es) field. Leave the IPv6 Address(es) and Alias(es) field blank.

c. Leave the check mark in the Create PTR Records? box (see Figure 5-8).

Figure 5-8 Adding Hosts to a Zone (Local Advanced)



d. Click **Add Host**.

Step 5 Add the second host, **userhost102**, with address **192.168.50.102**, in the same way. The two hosts should now appear along with the nameserver host on the List/Add Hosts for Zone page.

Create a Host Administrator Role with Constraints

In this part of the tutorial, the Boston example-cluster-admin creates the example-host-role with address constraints in the boston.example.com zone.

Local Advanced Web UI

Step 1 Log out as the example-zone-admin user and log in as the **example-cluster-admin** user (with password **exampleadmin**).

Step 2 Click **Advanced** to enter Advanced mode.

Step 3 Click **Administration**, then **Roles** to open the List/Add Administrator Roles page.

Step 4 Add the example-host-role:

a. Enter **example-host-role** in the Name field.

b. From the Base Role drop-down list, choose **host-admin**.

c. Click **Add Role** to open the Add Host Administrator Role page.

Step 5 Add the constraint for the role:

a. Click **Add Constraint**.

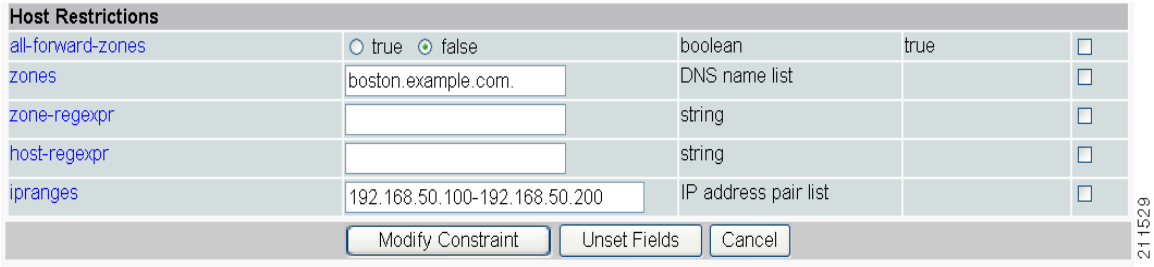
b. On the Add Role Constraint for Role page, scroll down to Host Restrictions.

c. For the *all-forward-zones* attribute, click the **false** radio button.

d. For the *zones* attribute, enter **boston.example.com**.

e. For the *ipranges* attribute, enter the range **192.168.50.101–192.168.50.200** (see Figure 5-9 for the Host Restrictions section of the Add Role Constraint for Role page).

Figure 5-9 Setting Host Restrictions for a Role (Local Advanced)



f. The *zone-regexpr* and *host-regexpr* attribute fields are for entering regular expressions to match zones and hosts, respectively, in regex syntax. (See Table 5-4 for the commonly used regex values.)

Table 5-4 Common Regex Values

Value	Matches
. (dot)	Any character (a wildcard). Note that to match a literal dot character (such as in a domain name), you must escape it by using a backslash (\), such that **\.com** matches .com.
char	Literal character (*char*) that follows, or the *char* has special meaning. Used especially to escape metacharacters such as the dot (**.**) or another backslash. Special meanings include **\d** to match decimal digits, **\D** for nondigits, **\w** for alphanumerics, and **\s** for whitespace.
*char***?**	Preceding *char* once or not at all, as if the character were optional. For example, **example\.?com** matches example.com or examplecom.
*char******	Preceding *char* zero or more times. For example, **ca*t** matches ct, cat, and caaat. This repetition metacharacter does iterative processing with character sets (see **[***charset***]**).
*char***+**	Preceding *char* one or more times. For example, **ca+t** matches cat and caaat (but not ct).

Table 5-4 Common Regex Values (continued)

Value	Matches
[*charset*]	Any of the characters enclosed in the brackets (a character set). You can include character ranges such as **[a–z]** (which matches any lowercase character). With the * repetition metacharacter applied, the search engine iterates through the set as many times as necessary to effect a match. For example, **a[bcd]*b** will find abcbd (by iterating through the set a second time). Note that many of the metacharacters (such as the dot) are inactive and considered literal inside a character set.
[^*charset*]	Anything but the *charset*, such that **[^a-zA-Z0-9]** matches any nonalphanumeric character (which is equivalent to using **\W**). Note that the caret outside a character set has a different meaning.
^	Beginning of a line.
$	End of a line.

g. Click **Add Constraint**. The constraint should have an index number of 1.

Step 6 Click **Add Role**. The example-host-role should now appear in the list of roles on the List/Add Administrator Roles page.

Create a Group to Assign to the Host Administrator

The Boston example-cluster-admin next creates an example-host-group that includes the example-host-role so that the example-zone-admin can assign this group to the example-host-admin.

Local Advanced Web UI

Step 1 As example-cluster-admin, still in Advanced mode, click the **Groups** submenu to open the List/Add Administrator Groups page.

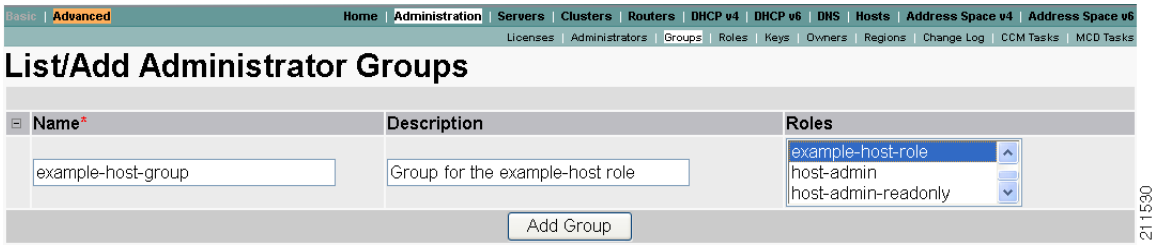
Step 2 Create the example-host-group and assign the example-host-role to it:

a. Enter **example-host-group** in the Name field.

b. Add a description such as **Group for the example-host-role**.

c. From the Roles drop-down list, choose **example-host-role** (see Figure 5-10).

Figure 5-10 Creating a Group (Local Advanced)



d. Click **Add Group**.

e. Change the page size at the bottom of the page to **20**, then click **Change Page Size**, so that the newly created group appears in the list.

Step 3 Log out as example-cluster-admin, then log in as the **example-zone-admin** user (with password **examplezone**).

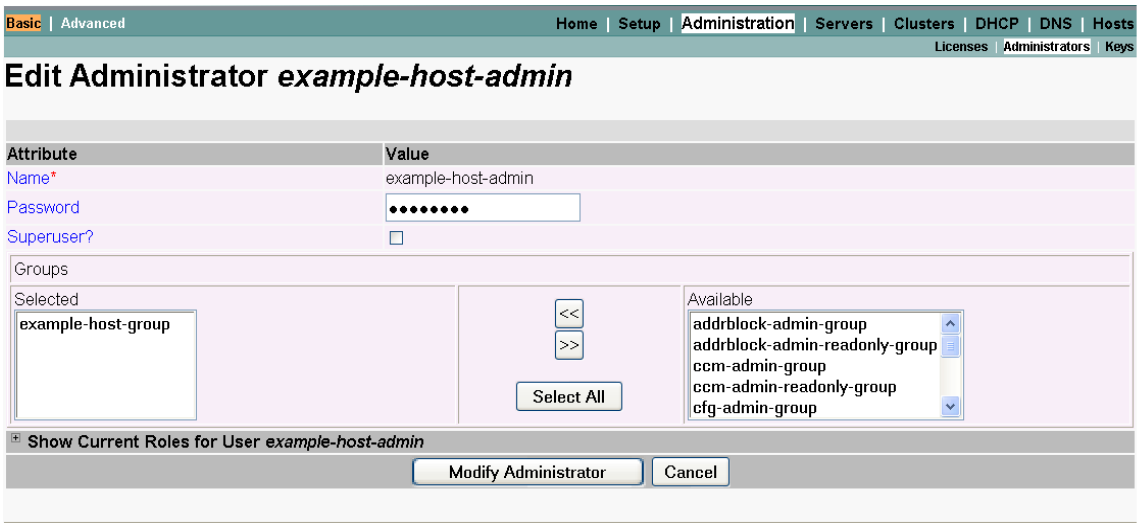
Step 4 As example-zone-admin, assign the example-host-group to the example-host-admin:

a. In Basic mode, click **Administration**, then **Administrators**.

b. On the List/Add Administrators page, click example-host-admin to edit the administrator.

c. On the Edit Administrator page, choose **example-host-group** in the Available list, then click **<<** to move it to the Selected list (see Figure 5-11).

Figure 5-11 Assigning a Group to an Administrator (Local Basic)



d. Click **Modify Administrator**. The example-host-admin should now show the example-host-group in the Groups column on the List/Add Administrators page.

Test the Host Address Range

The example-host-admin next tests an out-of-range address and then adds an acceptable one.

Local Advanced Web UI

Step 1 At the local cluster, log out as example-zone-admin, then log in as **example-host-admin** (with password **examplehost**).

Step 2 Click **Advanced** to enter Advanced mode.

Step 3 On the Main Menu page, click **Hosts**.

Step 4 On the List/Add Hosts for Zone page, try to enter an out-of-range address (note the range of valid addresses in the Valid IP Ranges field):

a. Enter **userhost3** in the Name field.

b. Deliberately enter an out-of-range address (**192.168.50.3**) in the IP Address(es) field.

c. Click **Add Host**. You should get an error message.

Step 5 Enter a valid address:

a. Enter **userhost103**.

b. Enter **192.168.50.103** in the IP Address(es) field.

c. Click **Add Host**. The host should now appear with that address in the list.

Regional Cluster Management Tutorial

This tutorial is an extension of the scenario described in the "Local Cluster Management Tutorial" section on page 5-15. In the regional cluster tutorial, San Jose has two administrators—a regional cluster administrator and a central configuration administrator. Their goal is to coordinate activities with the local clusters in Boston and Chicago so as to create DNS zone distributions, router configurations, and DHCP failover configurations using the servers at these clusters. The configuration consists of:

- One regional cluster machine in San Jose.
- Two local cluster machines, one in Boston and one in Chicago.
- One Cisco uBR7200 router in Chicago.

See Also

Administrator Responsibilities and Tasks, page 5-26
Create the Regional Cluster Administrator, page 5-26
Create the Central Configuration Administrator, page 5-26
Create the Local Clusters, page 5-27
Add a Router and Modify an Interface, page 5-28
Add Zone Management to the Configuration Administrator, page 5-29
Create a Zone for the Local Cluster, page 5-30
Pull Zone Data and Create a Zone Distribution, page 5-30
Create a Subnet and Pull Address Space, page 5-31
Push a DHCP Policy, page 5-31
Create a Scope Template, page 5-33
Create and Synchronize the Failover Pair, page 5-33

Administrator Responsibilities and Tasks

The regional administrators have the following responsibilities and tasks:

- **example-regional-admin**—Created by the superuser at the San Jose regional cluster, who creates the example-cfg-admin.
- **example-cfg-admin**:
  - Defines the Boston and Chicago clusters and checks connectivity with them.
  - Adds a router and modifies a router interface.
  - Pulls zone data from the local clusters to create a zone distribution.
  - Creates a subnet and policy, and pulls address space, to configure DHCP failover pairs in Boston and Chicago.

Create the Regional Cluster Administrator

The regional superuser first creates the example-regional-administrator, defined with groups, to perform cluster and user administration.

Regional Web UI

Step 1 Log in to the regional cluster as superuser (usually **admin**).

Step 2 Click **Administration**, then **Administrators** to open the List/Add Administrators page (see Figure 5-3 on page 5-17 for the local cluster version of this page, which is essentially identical).

Step 3 Enter **example-regional-admin** in the Name field, then **examplereg** in the Password field.

Step 4 Multiselect **central-cfg-admin-group** (for cluster administration) and **regional-admin-group** (for user administration) in the Groups drop-down list.

Step 5 Click **Add Administrator**.

Create the Central Configuration Administrator

As part of this tutorial, the example-regional-admin next logs in to create the example-cfg-admin, who must have regional configuration and address management capabilities.

Regional Web UI

Step 1 Log out as superuser, then log in as **example-regional-admin** with password **examplereg**. Note that the administrator has all but host and address space administration privileges.

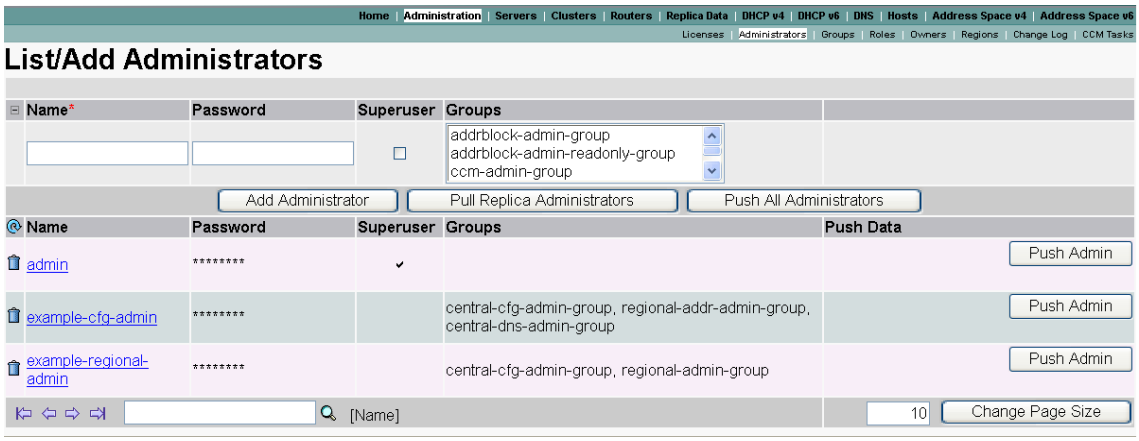
Step 2 Click **Administration**, then **Administrators** to open the List/Add Administrators page.

Step 3 Enter **example-cfg-admin** in the Name field, then **cfgadmin** in the Password field.

Step 4 Multiselect **central-cfg-admin-group** and **regional-addr-admin-group** in the Groups drop-down list.

Step 5 Click **Add Administrator**. The example-cfg-admin now appears with the two groups assigned (see Figure 5-12 on page 5-27).

Figure 5-12 Adding a Regional Administrator (Regional)



You can also add constraints for the administrator. Click **Add Constraint** and, on the Add Role Constraint for Role page, choose the read-only, owner, or region constraints, then click **Add Constraint**.

Create the Local Clusters

The example-cfg-admin next creates the two local clusters for Boston and Chicago.

Regional Web UI

Step 1 Log out as example-regional-admin, then log in as **example-cfg-admin** with password **cfgadmin**.

Step 2 Click **Clusters**, then **Cluster List**.

Step 3 On the List Server Clusters page, click **Add Cluster**.

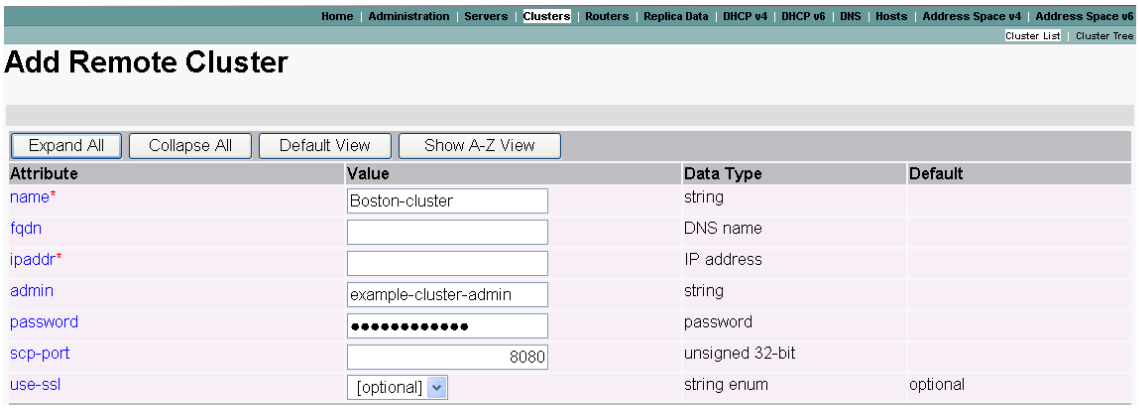
Step 4 On the Add Server Cluster page, create the Boston cluster based on data provided by its administrator:

- **a.** Enter **Boston-cluster** in the name field.
- **b.** Enter the IP address of the Boston server in the ipaddr field.
- **c.** Enter **example-cluster-admin** in the admin field, then **exampleadmin** in the password field.
- **d.** Enter in the scp-port field the SCP port to access the cluster as set at installation (**1234** is the preset value).
- **e.** Enter in the http-port field the HTTP port to access the cluster (**8080** is the preset value). (See Figure 5-13 on page 5-28.)
- **f.** Click **Add Cluster**.

Step 5 Create the Chicago cluster in the same way, except use **Chicago-cluster** in the name field, enter the remaining values based on data provided by the Chicago administrator, then click **Add Cluster**. The two clusters should now appear on the List Server Clusters page.

Step 6 Confirm the cluster connectivity. Click **Cluster Tree**. The created server clusters should appear with their servers listed on the View Tree of Server Clusters page.

Figure 5-13 Adding a Server Cluster (Regional)



Step 7 Connect to the Boston cluster. Click the Go Local icon ( ) next to Boston-cluster. If this opens the local cluster Manage Servers page, this confirms the administrator connectivity to the cluster. To return to the regional cluster web UI, click the Go Regional icon ( ).

Step 8 Connect to the Chicago cluster to confirm the connectivity in the same way.

Step 9 Confirm that you can replicate data for the two forward zones from the Boston cluster synchronization:

a. Click **Replica Data**.

b. On the View Replica Class List page, click Boston-cluster in the Select Cluster list.

c. In the Select Class list, click **Forward Zones**.

d. Click the Replicate icon ( ) in the Replicate Data column.

e. Click **View Replica Class List**. On the List Replica Forward Zones for Cluster page, you should see the boston.example.com and example.com zones.

Add a Router and Modify an Interface

The example-cfg-admin next takes over at the regional cluster to add a router and modify one of its interfaces to configure the DHCP relay agent. Adding the router pulls in the subnets already defined in the router configuration. This should occur now to prevent overlapping subnets and router synchronization errors when you add additional address space. (Because the routers define the physical network, it is preferable to save these definitions as opposed to saving those possibly conflicting definitions present in the DHCP configuration.)

Regional Web UI

Step 1 As example-cfg-admin, on the Main Menu page, click **Routers**, then **Router List**.

Step 2 On the List Routers page, click **Add Router**.

Step 3 On the Add Router page, add the router based on data from its administrator:

a. Give the router a distinguishing name in the name field. For this example, enter **router-1**.

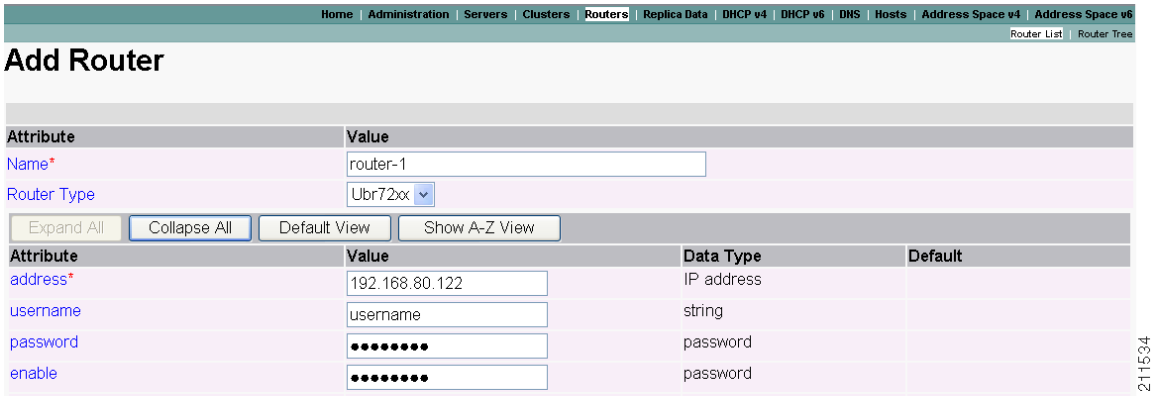
b. Because this router is a Cisco uBR7200 router, choose **Ubr72xx** in the Router Type drop-down list.

c. Enter the router IP address in the address field.

d. Enter the router administrator username in the username field.

e. Enter the router administrator enable password in the enable field (see Figure 5-14).

Figure 5-14 Adding a Router (Regional)



f. Click **Add Router**. The router should now appear on the List Routers page.

Step 4 Confirm that the router is created. Click **Router Tree** to view the hierarchy of router interfaces for router-1 on the View Tree of Routers page.

Step 5 Configure a DHCP relay agent for the router:

a. Click one of the interface names on the View Tree of Routers page to open the Edit Router Interface page. (Alternatively, from the List Routers page, click the Interfaces icon ( ) associated with the router, then click the interface name on the List Router Interfaces for Router page.)

b. On the Edit Router Interface page, enter the IP address of the DHCP server in the ip-helper field.

c. Click **Modify Router Interface** at the bottom of the page.

Step 6 Confirm with the router administrator that the DHCP relay agent was successfully added.

Add Zone Management to the Configuration Administrator

Because there are no zones set up at the Chicago cluster, the example-cfg-admin can create a zone at the regional cluster to make it part of the zone distribution. However, the example-regional-admin must first modify the example-cfg-admin to be able to create zones.

Regional Web UI

Step 1 Log out as example-cfg-admin, then log in as **example-regional-admin**.

Step 2 Click **Administration**, then **Administrators**.

Step 3 On the List/Add Administrators page, click example-cfg-admin.

Step 4 On the Edit Administrator page, click central-dns-admin-group in the Groups Available list, then move it (using **<<**) to the Selected list. The Selected list should now have central-cfg-admin-group, regional-addr-admin-group, and central-dns-admin-group.

Step 5 Click **Modify Administrator**. The change should be reflected on the List/Add Administrators page.

Create a Zone for the Local Cluster

The example-cfg-admin next creates the chicago.example.com zone for the zone distribution with the Boston and Chicago zones.

Regional Web UI

Step 1 Log out as example-regional-admin, then log in as **example-cfg-admin**.

Step 2 On the Main Menu page, click **DNS**, then **Forward Zones**.

Step 3 On the List Forward Zones page, enter **chicago.example.com** in the Name field.

Step 4 Click **Add Zone**.

Step 5 On the Add Zone page, enter:

a. **Serial Number—1**.

b. **Nameserver—ns1**.

c. **Contact E-mail—hostmaster**.

d. **Nameservers—ns1** (click **Add Nameserver**).

e. Click **Add Zone**.

Step 6 Click the **Reverse Zones** submenu.

Step 7 On the List Reverse Zones page, create the **60.168.192.in-addr.arpa** reverse zone for the Chicago zone, with the proper attributes set.

Pull Zone Data and Create a Zone Distribution

The example-cfg-admin next pulls zone data from Boston and Chicago and creates a zone distribution.

Regional Web UI

Step 1 As example-cfg-admin, click **Zone Distributions**.

Step 2 On the List/Add Zone Distributions page, pull the zone from the replica database:

a. Click **Pull Replica Zone Data**.

b. On the Select Pull Replica Zone Data page, leave the Data Synchronization Mode defaulted as Update, then click **Report** to open the Report Pull Replica Zone Data page.

c. Notice the change sets of data to pull, then click **Run**.

d. On the Run Pull Replica Zone Data page, click **OK**.

Step 3 On the List/Add Zone Distributions page, notice that the Boston cluster zone distribution is assigned an index number (**1**) in the Name column. Click the number.

Step 4 On the Edit Zone Distribution page, in the Primary Server field, click Boston-cluster. (The IP address of the Boston-cluster becomes the first master server in the Master Servers list.)

Step 5 Because we want to make the Chicago-cluster DNS server a secondary server for the Boston-cluster:

a. Click **Add Server** in the Secondary Servers area.

b. On the Add Zone Distribution Secondary Server page, choose **Chicago-cluster** in the Secondary Server drop-down list.

c. Click **Add Secondary Server**.

Step 6 On the Edit Zone Distribution page, in the Forward Zones area, move **chicago.example.com** to the Selected list.

Step 7 In the Reverse Zones area, move **60.168.192.in-addr.arpa** to the Selected list.

Step 8 Click **Modify Zone Distribution**.

Create a Subnet and Pull Address Space

The example-cfg-admin next creates a subnet at the regional cluster. This subnet will be combined with the other two pulled subnets from the local clusters to create a DHCP failover server configuration.

Regional Web UI

Step 1 As example-cfg-admin, click **Address Space**, then **Subnets** to open the List/Add Subnets page (see Figure 15-4 on page 15-13). You should see the subnets created by adding the router (in the "Add a Router and Modify an Interface" section on page 5-28).

Step 2 Create an additional subnet, 192.168.70.0/24:

a. Enter **192.168.70** (the abbreviated form) as the subnet network address in the Address/Mask field.

b. Leave the **24** (255.255.255.0) selected as the network mask.

c. Click **Add Subnet**.

Step 3 Click **Address Space** to confirm the subnet you created.

Step 4 On the View Unified Address Space page, click **Pull Replica Address Space**.

Step 5 On the Select Pull Replica Address Space page, leave everything defaulted, then click **Report**.

Step 6 The Report Pull Replica Address Space page should show the change sets for the two subnets from the clusters. Click **Run**.

Step 7 Click **OK**. The two pulled subnets appear on the List/Add Subnets page.

Push a DHCP Policy

The example-cfg-admin next creates a DHCP policy, then pushes it to the local clusters.

Regional Web UI

Step 1 As example-cfg-admin, click **DHCP**, then **Policies**.

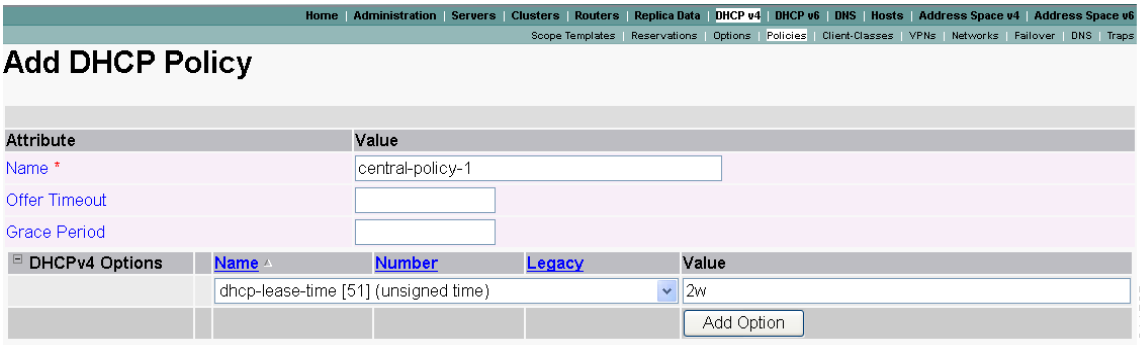
Step 2 On the List DHCP Policies page, click **Add Policy**.

Step 3 On the Add DHCP Policy page, create a central policy for all the local clusters:

a. Enter **central-policy-1** in the Name field. Leave the offer timeout and grace period values as is.

b. Enter a lease period. In the DHCPv4 Options drop-down list, choose **dhcp-lease-time [51] (unsigned time)**, then enter **2w** (two weeks) for the lease period in the Value field (see Figure 5-15).

Figure 5-15 Adding a DHCP Policy (Regional)



c. Click **Add Option**.

d. Click **Add Policy**. The central-policy-1 should appear on the List DHCP Policies page.

Step 4 Push the policy to the local clusters:

a. Click **Push Policy** next to central-policy-1.

b. On the Push DHCP Policy Data to Local Clusters page, leave the Data Synchronization Mode as **Ensure**. This ensures that the policy is replicated at the local cluster, but does not replace its attributes if a policy by that name already exists.

c. Click **Select All** in the Destination Clusters section of the page.

d. Click **<<** to move both clusters to the Selected field.

e. Click **Push Data to Clusters**.

f. View the push operation results on the View Push DHCP Policy Data Report page, then click **OK**.

Step 5 Confirm that the policy now exists at the local cluster:

a. Click **Clusters** to open the View Tree of Server Clusters page.

b. Next to Boston-cluster, click the Go Local icon ( ) in the Connect column.

c. In the Boston cluster web UI, click **DHCP**, then **Policies** to open the List DHCP Policies page. The central-policy-1 should appear.

d. Click the policy name to confirm the attributes set for it.

e. Click the Go Regional icon ( ) at the top right corner of the page to return to the regional cluster.

Create a Scope Template

The example-cfg-admin next creates a DHCP scope template to handle failover server pair creation.

Regional Web UI

Step 1 As the example-cfg-admin user, click **DHCP**, then **Scope Templates**.

Step 2 On the List DHCP Scope Templates page, click **Add Scope Template**.

Step 3 Set the basic properties for the scope template—Enter or choose the following values in the fields:

a. **Name**—Enter **scope-template-1**.

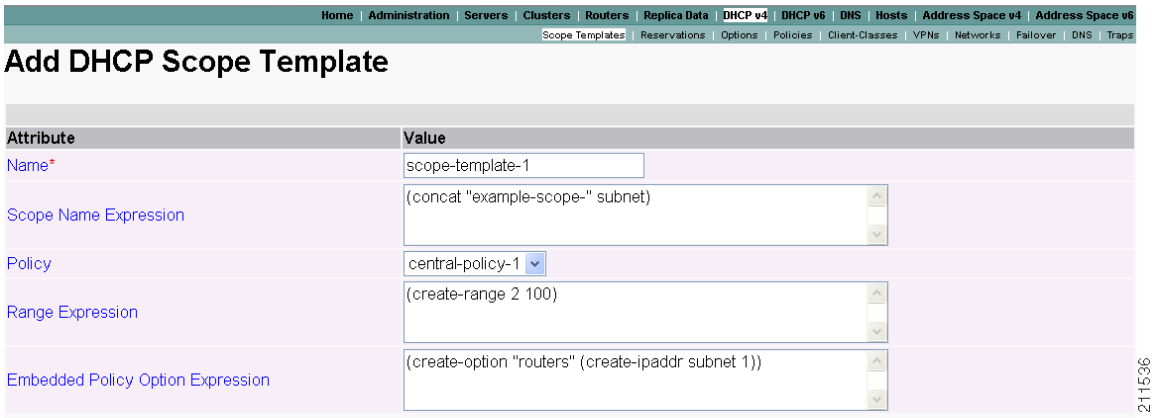
b. **Scope Name Expression**—To autogenerate names for the derivative scopes, concatenate the example-scope string with the subnet defined for the scope. To do this, enter **(concat "example-scope-" subnet)** in the field (including the parentheses).

c. **Policy**—Choose **central-policy-1** in the drop-down list.

d. **Range Expression**—Create an address range based on the remainder of the subnet (the second through last address) by entering **(create-range 2 100)**.

e. **Embedded Policy Option Expression**—Define the router for the scope in its embedded policy and assign it the first address in the subnet by entering **(create-option "routers" (create-ipaddr subnet 1))** (see Figure 5-16).

Figure 5-16 Adding a Scope Template (Regional)



Step 4 Click **Add Scope Template**. The template should appear on the List DHCP Scope Templates page.

Create and Synchronize the Failover Pair

The example-cfg-admin next creates the failover server pair relationship and synchronizes the failover pair. The DHCP server at Boston becomes the main, and the server at Chicago becomes the backup.

Regional Web UI

Step 1 As the example-cfg-admin user, click the **Failover** submenu.

Step 2 On the List DHCP Failover Pairs page, click **Add Failover Pair**.

Step 3 On the Add DHCP Failover Pair page, enter or choose the following values:

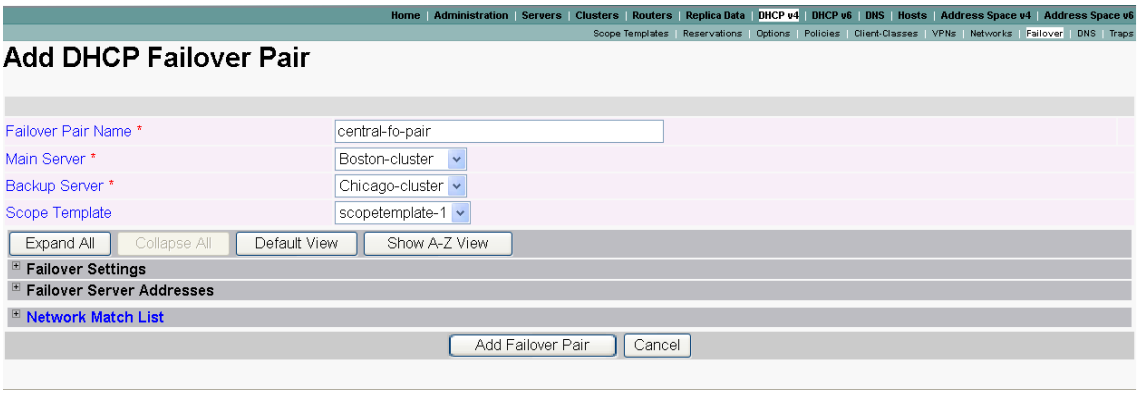
a. **Failover Pair Name**—Enter **central-fo-pair**.

b. **Main Server**—Click **Boston-cluster**.

c. **Backup Server**—Click **Chicago-cluster**.

d. **Scope Template**—Click **scopetemplate-1** (see Figure 5-17).

Figure 5-17 Adding a Failover Pair (Regional)



e. Click **Add Failover Pair**.

Step 4 Synchronize the failover pair with the local clusters:

a. On the List DHCP Failover Pairs page, click the Report icon ( ) in the Synchronize column.

b. On the Report Synchronize Failover Pair page, accept **Local Server** as the source of network data.

c. Accept **Main to Backup** as the direction of synchronization.

d. Accept the operation **Update**.

e. Click **Report** at the bottom of the page.

f. On the View Failover Pair Sync Report page, click **Run Update**.

g. Click **Return**.

Step 5 Confirm the failover configuration and reload the server at the Boston cluster:

a. On the List DHCP Failover Pairs page, click the Go Local icon ( ) next to Boston-cluster.

b. On the Manage DHCP Server page, click the Reload icon ( ).

c. Click the Go Regional icon ( ) at the top of the page to return to the regional cluster.

Step 6 Confirm the failover configuration and reload the server at the Chicago cluster in the same way.



CHAPTER 6

Managing the Central Configuration

This chapter explains how to manage the central configuration at the Cisco Network Registrar regional cluster.

Central Configuration Tasks

Central configuration management at the regional cluster can involve:

- Setting up server clusters, replicating their data, and polling subnet utilization and lease history data from them.
- Setting up routers (see Chapter 11, "Managing Router Interface Configurations").
- Managing network objects such as DHCP scope templates, policies, client-classes, options, networks, and virtual private networks (VPNs).
- Managing DHCP failover server pairs.

These functions are available only to administrators assigned the central-cfg-admin role. (The full list of functions for the central-cfg-admin are listed in Table 5-2 on page 5-4.) Note that central configuration management does not involve setting up administrators and checking the status of the regional servers. These functions are performed by the regional administrator, as described in the "Licensing" section on page 5-8 and "Managing Servers" section on page 7-1.

See Also

Configuring Server Clusters, page 6-2
Managing DHCP Scope Templates, page 6-12
Managing DHCP Policies, page 6-13
Managing DHCP Client-Classes, page 6-15
Managing Virtual Private Networks, page 6-16
Managing DHCP Failover Pairs, page 6-17
Managing Lease Reservations, page 6-18

Configuring Server Clusters

Server clusters are groupings of CCM, DNS, DHCP, and TFTP servers at local cluster locations. For example, an organization might have Boston and Chicago clusters of DNS and DHCP servers. A central administrator might want to affect how addresses are allocated at these clusters, or poll subnet utilization or lease history data from them. The central administrator might even want to connect to those local clusters, if the required permissions exist, to view changes there or restart the servers.

View the created clusters on the View Tree of Cluster Servers page. To get there, click **Clusters**. Once the page is populated with clusters, it shows some rich information and provides some useful functions. The Go Local icon () allows single sign-on to a local cluster web UI, if an equivalent administrator account exists at the local cluster.

The View Tree of Clusters page might have been populated by manually adding clusters on the List Server Clusters page, or automatically when adding and synchronizing with routers, which also creates server clusters. The cluster names are links that you can click to edit the cluster information. The resynchronization, replication, and polling functions are described further on in this chapter.

The DHCP server may have the Related Servers icon () next to the DHCP server for the cluster. Click this icon to open the List Related Servers for DHCP Server page (see the "Listing Related Servers for DHCP Servers" section on page 6-4). These servers can be DNS, TFTP, or DHCP failover servers.

See Also

Adding Local Clusters
Editing Local Clusters, page 6-3
Listing Related Servers for DHCP Servers, page 6-4
Connecting to Local Clusters, page 6-6
Synchronizing with Local Clusters, page 6-7
Replicating Local Cluster Data, page 6-7
Viewing Replica Data, page 6-7
Deactivating, Reactivating, and Recovering Data for Clusters, page 6-8
Polling Subnet Utilization and Lease History Data, page 6-9
Enabling Subnet Utilization Collection, page 6-11Enabling Lease History Collection, page 6-12

Adding Local Clusters

Adding local clusters to the regional cluster is the core functionality of the central-cfg-admin role.

To enable subnet utilization and lease history data collection, see the "Polling Subnet Utilization and Lease History Data" section on page 6-9.

The minimum required values to add a cluster are its name, IP address of the machine, administrator username, and password. The cluster name must be unique and its IP address must match that of the host where the CCM database is located. Obtain the SCP and HTTP ports, username, and password from the local cluster administrator. The preset value at Network Registrar installation for the SCP port is 1234 and the HTTP port is 8080.

You can also set whether you want outbound connections to local servers to be secure by setting the *use-ssl* attribute to optional or required. It is set to optional by default, and it requires the Network Registrar Communications Security Option installed to be effective.

Regional Web UI

View local clusters on the List Server Clusters page, which is similar to the View Tree of Server Clusters page, except that you cannot expand the clusters to show their servers. However, you can add server clusters on the List Server Clusters page, which you cannot do on the View Tree of Server Clusters page. Both pages provide the following functions:

- Connect to a local cluster web UI for local administration.
- Resynchronize with a local cluster to reconcile updates there.
- Pull data over to a regional cluster replica database.
- Query subnet utilization data from a local cluster. This function appears only if you are assigned the regional-addr-admin role with at least the subnet-utilization subrole.
- Query lease history data from a local cluster. This function appears only if you are assigned the regional-addr-admin role with at least the lease-history subrole.

To add a cluster, click **Add Cluster**. This opens the Add Server Cluster page. For an example of adding a local cluster, see the "Create the Local Clusters" section on page 5-27. Click **Add Cluster** to return to the List Server Clusters page.

Local Web UI

You can also manage clusters in the local web UI. See the "Configuring Clusters in the Local Web UI" section on page 2-11 for details.

CLI Commands

To add a cluster, use **cluster** *name* **create** *address* to give the cluster a name and address and set the important attributes. For example:

```
nrcmd> cluster example-cluster create 192.168.100.101 admin=admin password=changeme
```

Note that the administrator must be a superuser to fully synchronize at the local cluster.

Editing Local Clusters

Editing local clusters at the regional cluster is the core functionality of the central-cfg-admin role.

Regional Web UI

To edit a local cluster, click its name on the View Tree of Server Clusters page or List Server Clusters page to open the Edit Server Cluster page. This page is essentially the same as the Add Server Cluster page, except for an additional attribute unset function. Make your changes, then click **Modify Cluster**.

Local Web UI

You can also edit clusters in the local web UI. See the "Configuring Clusters in the Local Web UI" section on page 2-11 for details.

CLI Commands

To edit a local cluster, use **cluster** *name* **set** *attribute* to set or reset the attributes. For example:

```
nrcmd> cluster Example-cluster set poll-replica-interval=8h
```

Listing Related Servers for DHCP Servers

If you have related DNS, TFTP, or DHCP failover servers (see the "Creating and Synchronizing Failover Server Pairs" section on page 27-4), you can access the attributes for these servers.

Regional Web UI

On the View Tree of Server Clusters page, click the Related Servers icon ( ) next to the DHCP server for the cluster to open the List Related Servers for DHCP Server page. This page shows the communication and failover states the servers are in. Table 6-1 describes the attributes on this page. (For this page to appear, you must be assigned the central-cfg-admin role with the dhcp-management subrole.)

Table 6-1 Attributes for Related Servers

Related Server Attribute	Description
Related Server Type	Type of related server: DHCP, DNS, or LDAP.
Related Server IP Address	IP address of the related server. For DHCP failover partners, click this link to open the View Failover Related Server page (see Table 6-2 on page 6-5).
Communications	State of the communication—None, OK, or Interrupted.
Requests	Applies to DNS or LDAP related servers only, the number of requests from these servers.
State	For DHCP failover—None, Startup, Normal, Communications-interrupted, Partner-down, Potential-conflict, Recover, Paused, Shutdown, or Recover-done. For High-Availability (HA) DNS—Send-Update, Probe, or ha-state-unknown. Only the server that is successfully updating can be in Send-Update state. The partner server not sending updates is then always in Probe or unknown state. When the DHCP server comes up if there is no client activity, both DNS servers are often in the unknown state. This changes when the DHCP server tries to do DNS updates.
Partner Role	For DHCP failover only, the failover role of the partner—Main or Backup.
Partner State	For DHCP failover only, the partner's state—None, Startup, Normal, Communications-interrupted, Partner-down, Potential-conflict, Recover, Paused, Shutdown, or Recover-done.
Update Response Complete	For DHCP failover only, the percentage of completed update responses, valid only if there are outstanding update responses.

Table 6-2 Attributes for DHCP Related Failover Servers

Failover Partner Attribute	Description
General attributes	
current-time	Current time on the server returning this object.
comm-state	None, OK, or Interrupted.
maximum-client-lead-time	Current maximum client lead time (MCLT) on this system.
sequence-number	Sequence number unique across failover objects, if different from the sequence in the lease, the lease is considered "not up to date" independent of the sf-up-to-date lease flag.
Local server information	
our-ipaddr	IP address of the interface to this server.
role	Failover role of the server returning this object—None, Main, or Backup.
state	State of the local server—None, Startup, Normal, Communications-interrupted, Partner-down, Potential-conflict, Recover, Paused, Shutdown, or Recover-done.
start-time-of-state	Time at which the current failover state began.
start-of-comm-interrupted	Time at which this partner most recently went into communications-interrupted state. This is valid across reloads, while the start-time-of-state never has a time earlier than the most recent server reload.
est-end-recover-time	Valid if *update-request-in-progress* is not set to None. If it appears, the time at which the server enters the recover- done state if the update request outstanding is complete. If it does not appear, then the server enters recover-done whenever update-request is completed.
use-other-available	If false or unset, then this server cannot use other-available leases. If true, then the server can use other-available leases. Valid at all times, but should only be true if in partner-down state.
use-other-available-time	If, in partner-down state, the *use-other-available* is false or unset, the time when *use-other-available* will go to true.
safe-period-remaining	Duration in seconds remaining in safe-period. If not set to 0, then this server is currently running down a safe period with respect to its partner.
Partner server information	
ipaddr	IP address of the partner server.
partner-role	Failover role of the partner of the server returning this object—None, Main, or Backup.
partner-state	Last known state which the partner end of the failover relationship is in—None, Startup, Normal, Communications-interrupted, Partner-down, Potential-conflict, Recover, Paused, Shutdown, or Recover-done.
start-time-of-partner-state	Time at which the partner current failover state began.
est-partner-end-recover-time	If the *partner-state* is Recover, an estimated prediction of when the partner will time out its MCLT and finish being in recover state.
last-comm-ok-time	Time at which this server last found communications to be OK.

Table 6-2 Attributes for DHCP Related Failover Servers (continued)

Failover Partner Attribute	Description
Update requests sent to partner	
update-request-outstanding	If None or unset, then the server does not have an update request queued for its partner. If not set to None, then it does have an update request queued for its failover partner. Valid values are None, Update, and Update-all.
update-request-start-time	Time at which any *update-request-outstanding* request was started.
update-request-done-time	Time at which the last of any update request completed.
Update requests processed for partner	
update-response-in-progress	If this server is processing an update response, gives information about the type and origin of the response.
update-response-percent-complete	If *update-response-outstanding* appears, the percent complete of the current update response.
update-response-start-time	Time that the update response mentioned in *update-response-in-progress* was started.
update-response-done-time	Time that the most recent update response sent an update done to the partner server.

Other controls are available on these pages:

- To refresh the data on the View Failover Related Server page, click **Refresh Data**.
- On the View Failover Related Server page, if the partner is in the Communications-interrupted failover state, you can click **Set Partner Down** in association with an input field for the partner-down date setting. This setting is initialized to the value of the *start-of-communications-interrupted* attribute. (In Normal web UI mode, you cannot set this date to be an earlier value than the initialized date. In Expert web UI mode, you can set this value to any date.) After clicking **Set Partner Down**, you return to the List Related Servers for DHCP Server page to view the result of the partner-down action. Never set both partners to Partner Down mode.
- To return from the List Related Servers for DHCP Server page or View Failover Related Server page, click **OK**.

CLI Commands

To list the related servers in a DHCP failover pair, use **dhcp getRelatedServers**.

Connecting to Local Clusters

In the web UI, if you have an equivalent administrator account at the local cluster, you can single sign-on to the local cluster Manage Servers page by clicking the Go Local icon () next to the cluster name on the View Tree of Server Clusters page or List Server Clusters page. To return to the regional cluster web UI, click the Go Regional icon () at the top right corner of the local cluster page. If you do not have an equivalent account at the local cluster, the Go Local icon opens the local cluster login page.

Synchronizing with Local Clusters

Synchronization is configuring regional and local clusters so that they can work together in a unified fashion. When you synchronize:

1. The list of local servers are copied to the regional cluster.
2. A shared secret is established between the regional and local clusters for single sign-on.

Synchronization occurs once when you create a local cluster at the regional cluster. However, changes might occur at the local cluster periodically, requiring you to resynchronize with it. For example, you might change the username and password used to make local connections. Resynchronization does not happen automatically—you must click the Resynchronize icon ( ) next to the cluster name on the List Server Clusters page. The result is a positive confirmation for success or an error message for a failure.

When you upgrade the local cluster, you should also resynchronize the cluster. For synchronization to be effective, the user account specified for the local cluster must be a superuser. If you get a synchronization error message, check the local cluster to ensure that it is running properly.

Note When you resynchronize clusters at the regional cluster, an automatic reinitialization of replica data occurs. The result is that for larger server configurations, resynchronization might take several minutes. The benefit, however, is that you do not need a separate action to update the replica data.

Replicating Local Cluster Data

Replication is copying the configuration data from a local server to the regional cluster replica database. Replication needs to occur before you can pull DHCP object data into the regional server database. During replication:

1. The current data from the local database is copied to the regional cluster. This usually occurs once.
2. Any changes made in the master database since the last replication are copied over.

Replication happens at a given time interval. You can also force an immediate replication by clicking the Replicate icon ( ) next to the cluster name on the View Tree of Server Clusters page or List Server Clusters page.

You can set the automatic replication interval on the Add Server Cluster page, or adjust it on the Edit Server Cluster page, using the *poll-replica-interval* attribute. This interval is preset at four hours. You can also set the fixed time of day to poll replica data by using the *poll-replica-offset* attribute; its default value is zero hours (no offset).

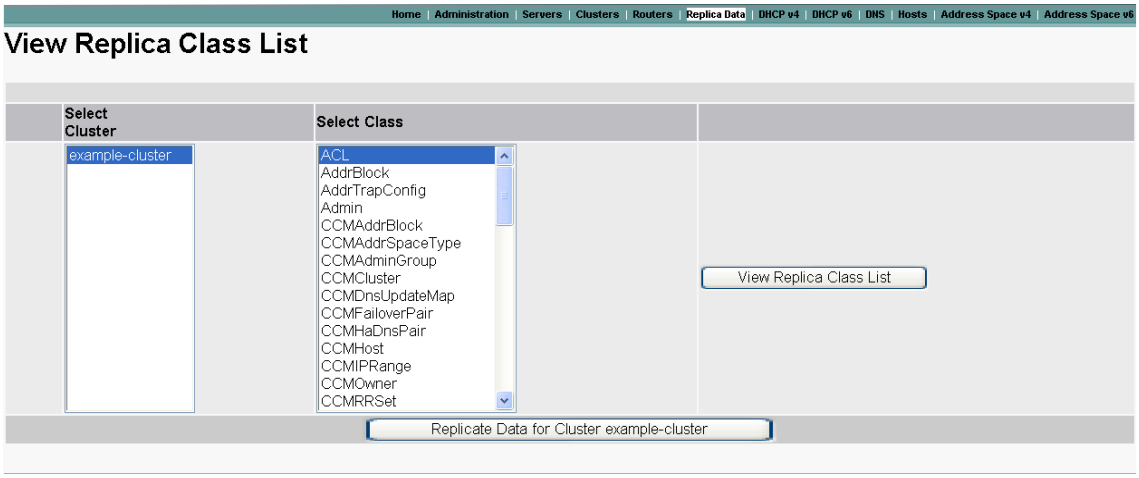
Caution If the replica database is corrupted in any way, the regional CCM server will not start. If you encounter this problem, stop the regional service, remove (or move) the replica database files located in the *install-path*/regional/data/replica directory (and the log files in the /logs subdirectory), then restart the regional server. Doing so recreates the replica database without any data loss.

Viewing Replica Data

In the web UI, you can view the replica data cached in the replica database at the regional cluster by clicking **Replica Data**. This opens the View Replica Class List page (see Figure 6-1 on page 6-8).

Regional Web UI

Figure 6-1 View Replica Class List Page (Regional)



On this page, select the:

1. Cluster in the Select Cluster list.
2. Object class in the Select Class list.
3. Replicate the data for the cluster and class chosen. Click the Replicate icon ( ).
4. View the replica data. Click **View Replica Class List**, which opens a List Replica Data for Cluster page for the cluster and specific class of object you choose. On this page, you can:
   - Click the name of an object to open a View page at the regional cluster. Return to the List Replica page by clicking **Return to** *object* **List**.

     Note The List Replica Address Blocks and List Replica Subnets pages do not provide this function. To view the address blocks or subnets for the local cluster, use the Go Local icon ( ).

   - Click the Go Local icon ( ) to go to the List page for the object at the local cluster. Return to the List Replica *object* page by clicking the Go Regional icon ( ).

Click **Return** on the List Replica Data for Cluster page to return to the View Replica Class List page.

Deactivating, Reactivating, and Recovering Data for Clusters

Deactivating a cluster might be necessary if you suspect that a hard disk error occurred where configuration data could have been lost. You can deactivate the cluster, remedy the problem, recover cluster data from the replica database, then reactivate the cluster. This saves you from having to delete and then recreate the cluster with all of its data lost in the process.

Deactivating, reactivating, and recovering the data for a cluster is available only in the web UI, and you must be an administrator assigned the central-config-admin role.

Data that is not recovered (and that you need to manually restore) includes:

- Contents of the **cnr.conf** file (see the "Modifying the cnr.conf File" section on page 7-22)
- Web UI configuration files
- Unprotected DNS resource records
- Product licenses
- Administrator accounts
- Lease history
- Extension scripts

Note Restoring the data to a different IP address requires some manual reconfiguration of such things as DHCP failover server pair and High-Availability (HA) DNS server pair addresses.

Regional Web UI

Deactivate a cluster by clicking the Activated icon ( ) in the Activation column for the cluster. This immediately changes the icon to the Deactivated ( ) icon to show the status of the cluster. Deactivating a cluster disables deleting, synchronizing, replicating data, and polling subnet utilization and lease history. These operations are not available while the cluster is deactivated.

Deactivating the cluster also displays the Recover icon ( ) in the Recover Data column of the cluster. Click this icon to recover the replica data. This opens a separate "in process" status window that prevents any operations on the web UI pages while the recovery is in process. As soon as the recovery is successful, the disabled functions are again enabled and available.

To reactivate the cluster, click the Deactivated icon ( ) to change back to the Activated icon ( ) and show the status as active.

Polling Subnet Utilization and Lease History Data

Subnet utilization and lease history data are automatically collected at any regional cluster where these features are enabled for the DHCP server or failover pair. The default polling interval to update the regional databases is 4 hours. You can poll the servers immediately by clicking the Poll icon ( ) for the cluster in the Poll Subnet Utilization column or Poll Lease History column on the View Tree of Server Clusters page or List Server Clusters page. For this manual polling, if the server is in a failover relationship, data is only retrieved for the subnets where the server is the main.

If you have address space privileges (you are assigned the regional-addr-admin role with at least the subnet-utilization and lease-history subroles), you can query the subnet utilization or lease history data by clicking **Address Space** (see the "Generating Subnet Utilization History Reports" section on page 9-14, or the "Running IP Lease Histories" section on page 22-21).

See Also

Polling Process, page 6-10
Adjusting the Polling Intervals, page 6-10

Polling Process

When the regional cluster polls the local cluster for subnet utilization or lease history, it first requests all available data up to the current time. This time is recorded in the history databases, and subsequent polls request only new data from this time forward. All times are stored relative to each local cluster time, adjusted for that cluster time zone.

If the times on each server are not synchronized, you might observe odd query results. For example, if the regional cluster time lags behind that of a local cluster, the collected history might be in the future relative to the time range queries at the regional cluster. If so, the result of the query would be an empty list. Data merged from the several clusters could also appear out of sequence, because of the different time skews between local clusters. This type of inconsistency would make it difficult to interpret trends. To avoid these issues, using a network time service for all clusters is strongly recommended.

Adjusting the Polling Intervals

You can adjust the automatic polling interval for subnet utilization and lease history, along with other attributes. These attributes are set in three places at the regional cluster, with the following priority:

1. **Failover pair** (see the "Managing DHCP Failover Pairs" section on page 6-17)—These values override the cluster settings (only for subnets in the failover pair), and set additional attributes to control how polling to the backup server occurs if the main server is not available:
   - If the main failover server is unavailable, the subnets on the backup server are polled.
   - If there are no failover pair settings for these attributes, the main server values are used.

   In the CLI, set the attributes listed in Table 6-3 using the **failover-pair** command.
2. **Cluster**—These values override the server-wide settings, unless they are unset, in which case the server values are used. The cluster values are set when adding or editing the cluster. In the CLI, set the attributes listed in Table 6-3, using the **cluster** command.
3. **Regional CCM server** (the preset polling interval is 4 hours)—This is set on the Edit CCM Server page, accessible by clicking **Servers**, then the Local CCM Server link. In the CLI, set the attributes listed in Table 6-3 using the **ccm** command.

Note If subnet utilization or lease history collection is not explicitly turned on at the local cluster DHCP server (see the "Enabling Subnet Utilization Collection" section on page 6-11 and the "Enabling Lease History Collection" section on page 6-12), no data is collected, even though polling is on by default. Subnet utilization collection at the DHCP server is distinct from polling at the regional cluster, and polling does not automatically trigger collection. Subnet utilization collection must occur before new polling picks up any new data. Because this collection is preset to every 15 minutes, the polling interval should be set higher than this interval (the automatic polling interval is preset to every 4 hours). This also means that subsequent explicit polling performed before the next *collect-addr-util-interval* will not return any new subnet utilization data.

Table 6-3 Subnet Utilization and Lease History Polling Regional Attributes

Attribute Type	Subnet Utilization	Lease History
Polling interval—How often to poll data	*poll-subnet-util-interval* 0 (no polling) to 1 year, preset to 4 hours for the CCM server	*poll-lease-hist-interval* 0 (no polling) to 1 year, preset to 4 hours for the CCM server

Table 6-3 Subnet Utilization and Lease History Polling Regional Attributes (continued)

Attribute Type	Subnet Utilization	Lease History
Retry interval—How often to retry after an unsuccessful polling	*poll-subnet-util-retry* 0 to 4 retries	*poll-lease-hist-retry* 0 to 4 retries
Offset—Hour of the day to guarantee polling	*poll-subnet-util-offset* 0 to 24h (0h= midnight)	*poll-lease-hist-offset* 0 to 24h (0h=midnight)
Polling priority for the regional failover pair—Pull data from the main or backup server first (failover pair setting only)	*poll-subnet-util-server-first* choose **mainserver** (preset value) or **backupserver**	*poll-lease-history-server-first* choose **mainserver** (preset value) or **backupserver**

The polling offset attribute ensures that polling occurs at a specific hour of the day, set as 24-hour time, in relation to the polling interval. For example, if you set the interval to 4h and the offset to 6h (6 A.M.), the polling occurs at 2 A.M., 6 A.M., 10 A.M., 2 P.M., 6 P.M., and 10 P.M. each day.

Enabling Subnet Utilization Collection

Step 1 Configure the local cluster DHCP server with scopes and address ranges so that clients have requested leases.

Step 2 Explicitly enable subnet utilization collection. The DHCP server attributes to set are:

- ***collect-addr-util-duration***—Maximum period the DHCP server maintains data. You must change this from the preset value of 0 (no collection) to some reasonable value (see the context sensitive help for this attribute for the impact on memory).

  If you are configuring simple DHCP failover, disable individual polling of the main and backup DHCP servers. Instead, enable the failover pair polling by setting the failover pair attribute *poll-subnet-util-interval*, so as to collect one set of data from both servers.

- ***collect-addr-util-interval***—Frequency the server collects snapshots of the data (preset to 15 minutes). How you juggle this value with that of the *collect-addr-util-duration* attribute determines how much memory you use (see the context sensitive help for this attribute).

  In the CLI, set the attributes using the **dhcp set** command.

Step 3 If in staged scope edit mode, reload the local cluster DHCP server.

Step 4 At the regional cluster, create the cluster that includes this DHCP server.

Step 5 Go to the Subnet Utilization Settings section of the Add Server Cluster or Edit Server Cluster page.

Step 6 Set the attributes in Table 6-3 on page 6-10.

Step 7 Click **Modify Cluster**.

Step 8 In the regional web UI, click the Poll Subnet Utilization icon ( ) for the cluster to obtain the initial set of subnet utilization data. This data is refreshed automatically at each polling interval. Note that if you subsequently click the Poll Subnet Utilization icon, new subnet utilization data does not appear until after the next collection interval (*collect-addr-util-interval*) on the DHCP server (preset to 15 minutes).

Enabling Lease History Collection

Step 1 Configure the local cluster DHCP server with scopes and address ranges so that clients have requested leases.

Step 2 Explicitly enable lease history data collection. The DHCP server attributes to set are:

- ***ip-history***—Set this to enabled.
- ***ip-history-detail***—Set this to enabled if you want to collect detailed history data.
- ***ip-history-max-age***—Limit on the age of the history records (preset to 4 weeks).

  In the CLI, set the attributes using the **dhcp set** command.

Step 3 If in staged scope edit mode, reload the local cluster DHCP server.

Step 4 At the regional cluster, create the cluster that includes this DHCP server.

Step 5 In the regional web UI, go to the Lease History Settings section of the Add Server Cluster or Edit Server Cluster page.

Step 6 Set the attributes in Table 6-3 on page 6-10.

Step 7 Click **Modify Cluster**.

Step 8 On the List Server Clusters page, click the Replica icon ( ) next to the cluster name.

Step 9 Click the Poll Lease History icon ( ) for the cluster involved to obtain the initial set of lease history data. This data is refreshed automatically at each polling interval.

Managing DHCP Scope Templates

Scope templates apply certain common attributes to multiple scopes. These common attributes include a scope name based on an expression, policies, address ranges, and an embedded policy options based on an expression. The scope templates you add or pull from the local clusters are visible on the List DHCP Scope Templates page.

For details on creating and editing scope templates, and applying them to scopes, see the "Creating and Applying Scope Templates" section on page 20-3. The regional cluster web UI has the added feature of pushing scope templates to local clusters and pulling them from local clusters.

See Also

Pushing Scope Templates to Local Clusters

Pulling Scope Templates from Replica Data, page 6-13

Pushing Scope Templates to Local Clusters

You can push the scope templates you create from the regional cluster to any of the local clusters. In the web UI, if you want to push a specific template to a cluster, click **Push Scope Template** on the List DHCP Scope Templates page. If you want to push all of them, click **Push All Scope Templates**. Both open the Push Scope Template Data to Local Clusters page.

Regional Web UI

The Push Scope Template Data to Local Clusters page identifies the data to push, how to synchronize it with the local cluster, and the cluster or clusters to which to push it. The data synchronization modes are:

- **Ensure** (preset value)—Ensures that the local cluster has new data without affecting any existing data.
- **Replace**—Replaces data without affecting other objects unique to the local cluster.
- **Exact**—Available for "push all" operations only. Use this with caution, because it overwrites the data and deletes any other objects unique to the local cluster.

Choose the destination cluster or clusters in the Available field and move it or them to the Selected field.

Tip The synchronization mode and cluster choice settings are persistent for the duration of the current login session, so that they are in effect each time you access this page, unless you change them.

After making these choices, click **Push Data to Clusters**. This opens the View Push Scope Template Data Report page.

Pulling Scope Templates from Replica Data

You may choose to pull scope templates from the replica data of the local clusters instead of explicitly creating them. (You may first want to update the policy replica data by clicking the Replicate icon [ ] next to the cluster name.) To pull the scope templates in the regional web UI, click **Pull Replica Scope Templates**.

Regional Web UI

The Select Replica DHCP Scope Template Data to Pull page shows a tree view of the regional server replica data for the local clusters' scope templates. The tree has two levels, one for the local clusters and one for the scope templates in each cluster. You can pull individual scope templates from the clusters, or you can pull all of their scope templates. To pull individual scope templates, expand the tree for the cluster, then click **Pull Scope Template** next to its name. To pull all the scope templates from a cluster, click **Pull All Scope Templates from Cluster**.

To pull the scope templates, you must also choose a synchronization mode:

- **Ensure**—Ensures that the regional cluster has new data without affecting any existing data.
- **Replace** (preset value)—Replaces data without affecting other objects unique to the regional cluster.
- **Exact**—Available for "pull all" operations only. Use this with caution, because it overwrites the data and deletes any other objects unique to the regional cluster.

Managing DHCP Policies

Every DHCP server must have one or more policies defined for it. Policies define lease duration, gateway routers, and other configuration parameters, in what are called DHCP options. Policies are especially useful if you have multiple scopes, because you need only define a policy once and apply it to the multiple scopes.

For details on creating and editing DHCP policies, and applying them to scopes, see the "Configuring DHCP Policies" section on page 21-1. The regional cluster web UI has the added feature of pushing policies to, and pulling them from, the local clusters.

See Also

Pushing Policies to Local Clusters
Pulling Policies from Replica Data, page 6-14

Pushing Policies to Local Clusters

You can also push the policies you create from the regional cluster to any of the local clusters. In the regional web UI, if you want to push a specific policy to a cluster, click **Push Policy** on the List DHCP Policies page. If you want to push all of them, click **Push All Policies**.

Regional Web UI

The Push DHCP Policy Data to Local Clusters page identifies the data to push, how to synchronize it with the local cluster, and the cluster or clusters to which to push it. The data synchronization modes are:

- **Ensure** (preset value)—Ensures that the local cluster has new data without affecting any existing data.
- **Replace**—Replaces data without affecting other objects unique to the local cluster.
- **Exact**—Available for push-all operations only. Use this with caution, because it overwrites the data and deletes any other objects unique to the local cluster.

Choose the destination cluster or clusters in the Available field and move it or them to the Selected field. Then click **Push Data to Clusters** to open the View Push Policy Data Report page.

Tip The synchronization mode and cluster choice settings are persistent for the duration of the current login session, so that they are in effect each time you access this page, unless you change them.

Pulling Policies from Replica Data

You may choose to pull policies from the replica data of the local clusters instead of explicitly creating them. (In the regional web UI, you may first want to update the policy replica data by clicking the Replicate icon [ ] next to the cluster name.) To pull the policies, click **Pull Replica Policies**.

Regional Web UI

The Select Replica DHCP Policy Data to Pull page shows a tree view of the regional server replica data for the local clusters' policies. The tree has two levels, one for the local clusters and one for the policies in each cluster. You can pull individual policies from the clusters, or you can pull all of their policies. To pull individual policies, expand the tree for the cluster, then click **Pull Policy** next to its name. To pull all the policies from a cluster, click **Pull All Policies from Cluster**.

To pull all the policies, you must also choose a synchronization mode:

- **Ensure**—Ensures that the regional cluster has new data without affecting any existing data.
- **Replace** (preset value)—Replaces data without affecting other objects unique to the regional cluster.
- **Exact**—Available for "pull all" operations only. Use this with caution, because it overwrites the data and deletes any other objects unique to the regional cluster.

Managing DHCP Client-Classes

Client-classes provide differentiated services to users that are connected to a common network. You can group your user community based on administrative criteria, and then ensure that each user receives the appropriate class of service. Although you can use the Network Registrar client-class facility to control any configuration parameter, the most common uses are for:

- **Address leases**—How long a set of clients should keep its addresses.
- **IP address ranges**—From which lease pool to assign clients addresses.
- **DNS server addresses**—Where clients should direct their DNS queries.
- **DNS hostnames**—What name to assign clients.
- **Denial of service**—Whether unauthorized clients should be offered leases.

For details on creating and editing client-classes, see Chapter 24, "Configuring Client-Classes and Clients." The regional cluster web UI has the added feature of pushing client-classes to, and pulling them from, the local clusters.

See Also

Pushing Client-Classes to Local Clusters
Pulling Client-Classes from Replica Data, page 6-16

Pushing Client-Classes to Local Clusters

You can also push the client-classes you create from the regional cluster to any of the local clusters. If you want to push a specific client-class to a cluster in the web UI, click **Push Client-Class** on the List DHCP Client-Classes page. If you want to push all of them, click **Push All Client-Classes**.

Regional Web UI

The Push Client-Class Data to Local Clusters page identifies the data to push, how to synchronize it with the local cluster, and the cluster or clusters to which to push it. The data synchronization modes are:

- **Ensure** (preset value)—Ensures that the local cluster has new data without affecting any existing data.
- **Replace**—Replaces data without affecting other objects unique to the local cluster.
- **Exact**—Available for "push all" operations only. Use this with caution, because it overwrites the data and deletes any other objects unique to the local cluster.

Choose the destination cluster or clusters in the Available field and move it or them to the Selected field. Then click **Push Data to Clusters** to open the View Push Client-Class Data Report page.

Tip The synchronization mode and cluster choice settings are persistent for the duration of the current login session, so that they are in effect each time you access this page, unless you change them.

Pulling Client-Classes from Replica Data

You may choose to pull client-classes from the replica data of the local clusters instead of explicitly creating them. (In the web UI, you might first want to update the client-class replica data by clicking the Replicate icon [ ] next to the cluster name.) To pull the client-classes, click **Pull Replica Client-Classes**.

Regional Web UI

The Select Replica DHCP Client-Class Data to Pull page shows a tree view of the regional server replica data for the local clusters' client-classes. The tree has two levels, one for the local clusters and one for the client-classes in each cluster. You can pull individual client-classes from the clusters, or you can pull all of their client-classes. To pull individual client-classes, expand the tree for the cluster, then click **Pull Client-Class** next to its name. To pull all the client-classes from a cluster, click **Pull All Client-Classes from Cluster**.

To pull the client-classes, you must also choose a synchronization mode:

- **Ensure**—Ensures that the regional cluster has new data without affecting any existing data.
- **Replace** (preset value)—Replaces data without affecting other objects unique to the regional cluster.
- **Exact**—Available for "pull all" operations only. Use this with caution, because it overwrites the data and deletes any other objects unique to the regional cluster.

Managing Virtual Private Networks

A virtual private network (VPN) is a specialized address space identified by a key. A VPN allows address overlap in a network, because the addresses are distinguished by separate keys. Most IP addresses exist in the global address space outside of a VPN. You can create regional VPNs only if you are an administrator assigned the dhcp-management subrole of the central-cfg-admin role.

For details on creating and editing VPNs, and applying them to various network objects, see the "Configuring Virtual Private Networks Using DHCP" section on page 23-15. The regional web UI has the added feature of pushing VPNs to local clusters and pulling them from local clusters.

See Also

Pushing VPNs to Local Clusters
Pulling VPNs from Replica Data, page 6-17

Pushing VPNs to Local Clusters

You can push the VPNs you create from the regional cluster to any of the local clusters. In the web UI, if you want to push a specific VPN to a cluster, click **Push VPN** on the List/Add VPNs page. If you want to push all of them, click **Push All VPNs**.

Regional Web UI

The Push VPN Data to Local Clusters page identifies the data to push, how to synchronize it with the local cluster, and the cluster or clusters to which to push it. The data synchronization modes are:

- **Ensure** (preset value)—Ensures that the local cluster has new data without affecting any existing data.
- **Replace**—Replaces data without affecting other objects unique to the local cluster.
- **Exact**—Available for "push all" operations only. Use this with caution, because it overwrites the data and deletes any other objects unique to the local cluster.

Choose the destination cluster or clusters in the Available field and move it or them to the Selected field. Then click **Push Data to Clusters** to open the View Push VPN Data Report page.

Tip The synchronization mode and cluster choice settings are persistent for the duration of the current login session, so that they are in effect each time you access this page, unless you change them.

Pulling VPNs from Replica Data

Instead of explicitly creating VPNs, you can pull them from the local clusters. (In the regional web UI, you may first want to update the VPN replica data by clicking the Replica icon [ ] next to the cluster name.) To pull the replica data, click **Pull Replica VPNs** to open the Select Replica VPN Data to Pull page.

This page shows a tree view of the regional server replica data for the local clusters' VPNs. The tree has two levels, one for the local clusters and one for the VPNs in each cluster. You can pull individual VPNs or you can pull all of them. To pull individual VPNs, expand the tree for the cluster, then click **Pull VPN** next to its name. To pull all the VPNs, click **Pull All VPNs from Cluster**.

To pull the VPNs, you must choose a synchronization mode:

- **Ensure**—Ensures that the regional cluster has new data without affecting any existing data.
- **Replace** (preset value)—Replaces data without affecting other objects unique to the regional cluster.
- **Exact**—Available for "pull all" operations only. Use this with caution, because it overwrites the data and deletes any other objects unique to the regional cluster.

Managing DHCP Failover Pairs

With DHCP failover, a backup DHCP server can take over for a main server if the latter comes off the network for any reason. You can use failover to configure two servers to operate as a redundant pair. If one server is down, the other server seamlessly takes over so that new DHCP clients can get, and existing clients can renew, their addresses. Clients requesting new leases need not know or care about which server responds to their lease request. These clients can obtain leases even if the main server is down.

In the regional web UI, you can view any created failover pairs on the List Failover Pairs page. To access this page, click **DHCP Configuration**, then **Failover**. This functionality is available only to administrators who are assigned the dhcp-management subrole of the central-cfg-admin role.

For details on creating and editing failover pairs, see the "Creating and Synchronizing Failover Server Pairs" section on page 27-4. The regional cluster web UI has the added feature of pulling addresses from local clusters to create the failover pairs.

To pull the address space for a failover pair, you must have regional-addr-admin privileges.

Regional Web UI

Step 1 On the List Failover Pairs page or View Unified Address Space page, click **Pull Replica Address Space**.

Step 2 Choose the data synchronization mode (**Update**, **Complete**, or **Exact**) on the Select Pull Replica Address Space page. The results of choosing these modes are described in the table on the page.

Step 3 Click **Report** at the bottom of the page.

Step 4 Click **Run** on the Report Pull Replica Address Space page.

Step 5 Click **OK** on the Run Pull Replica Address Space page.

Managing Lease Reservations

You can push lease reservations you create from the regional cluster to any of the local clusters. In the regional cluster web UI, click **Push All Reservations** on the List Reservations page (click **DHCP**, then **Reservations**). Note that you cannot push individual reservations. If the cluster pushed to is part of a DHCP failover configuration, pushing a reservation also pushes it to the partner server.

Regional Web UI

The ensuing page identifies the data to push, how to synchronize it with the local cluster, and the cluster or clusters to which to push it. The data synchronization modes are:

- **Ensure**—Ensures that the local cluster has new data without affecting any existing data.
- **Replace** (preset value)—Replaces data without affecting other objects unique to the local cluster.
- **Exact**—Available for "push all" operations only. Use this with caution, because it overwrites the data and deletes any other objects unique to the local cluster.

Choose the destination cluster or clusters in the Available field and move it or them to the Selected field.

Tip The synchronization mode and cluster choice settings are persistent for the duration of the current login session, so that they are in effect each time you access this page, unless you change them.

After making these choices, click **Push Data to Clusters**. This opens the View Push Reservations Data Report page. Click **OK** on this page.

You can also pull the replica address space on the List Reservations page, and opt whether to omit reservations when doing so. You should use this option only to reduce processing time when you are sure that there are no pending changes to reservations to merge. To omit reservations for the pull, click a check mark in the *Omit Reservations?* check box, then click **Pull Data**.

Tip When recovering from an error condition, in Expert mode on the List Reservations page, you can synchronize the CCM server from the DHCP server and the other way around.



CHAPTER 7

Maintaining Servers and Databases

This chapter explains how to administer and control your local and regional server operations.

Managing Servers

If you are assigned the server-management subrole of the ccm-admin role, you can manage the Network Registrar servers as follows:

- **Start**—Load the database and start the server.
- **Stop**—Stop the server.
- **Reload**—Stop and restart the server. (Note that you do not need to reload the server with each dynamic update. See Chapter 28, "Configuring DNS Update" for details.)
- **Check statistics**—See the "Displaying Statistics" section on page 7-11.
- **View logs**—See the "Searching the Logs" section on page 7-6.
- **Manage interfaces**—See the specific protocol pages for how to manage server interfaces.

Starting and stopping a server is self-explanatory. When you reload the server, Network Registrar performs three steps—stops the server, loads configuration data, and restarts the server. Only after you reload the server does it use your changes to the configuration.

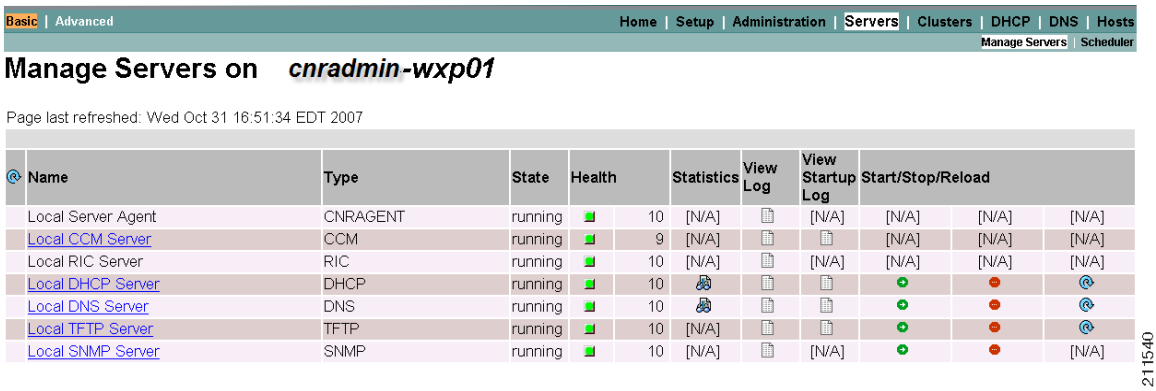
Note The DNS, DHCP, and SNMP servers are enabled by default to start on reboot. The TFTP server is not enabled by default to start on reboot. You can change this using [**server**] *type* **enable** or **disable start-on-reboot** in the CLI.

Local Basic or Advanced and Regional Web UI

You can manage the protocol servers in the following ways depending on if you are a:

- **Local or regional cluster administrator**—Click **Servers** to open the Manage Servers page (see Figure 7-1 on page 7-2 for a local cluster example).

Figure 7-1 Manage Servers Page (Local Basic)



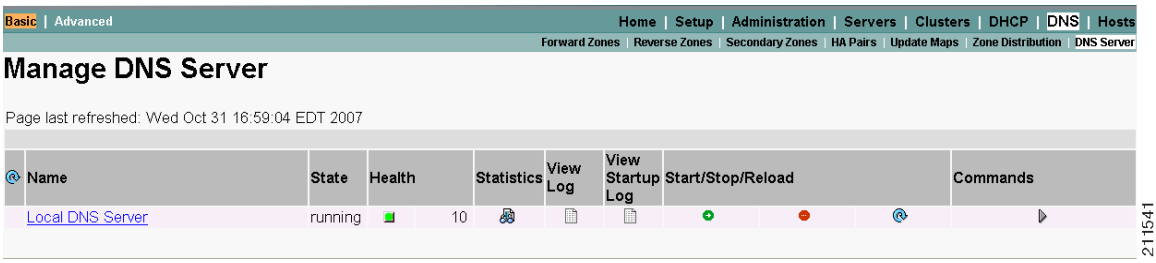
The local and regional cluster web UI access to server administration is identical, even though the available functions are different. As a regional administrator, you can check the state and health of the regional CCM server, server agent, and Router Interface Configuration (RIC) server. However, you cannot stop, start, reload, or view statistics, logs, or interfaces for them.

At the local cluster, you can manage the DHCP, DNS, TFTP, and SNMP servers:

- Click the Statistics icon ( ) to view statistics for the server. (See the "Displaying Statistics" section on page 7-11.)
- Click the Log icon ( ) in the View Log column to view the log messages for the server. (See the "Logging Server Events" section on page 7-6.)
- Click the Start icon ( ) to start the server.
- Click the Stop icon ( ) to stop the server.
- Click the Reload icon ( ) to reload the server.

- **Local cluster DNS administrator**—Click **DNS**, then **DNS Server** to open the Manage DNS Server page (see Figure 7-2).

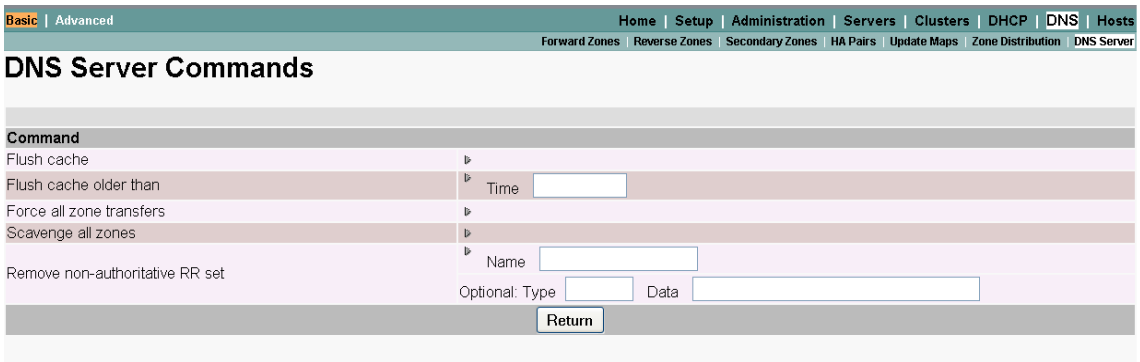
Figure 7-2 Manage DNS Server Page (Local Basic)



Along with the Statistics ( ), Log ( ), Start ( ), Stop ( ), and Reload ( ) functions, you can also perform other functions when you click the Run icon ( ) in the Commands column to open the DNS Server Commands page (see Figure 7-3 on page 7-3).

In Expert mode, you can also synchronize the CCM server from the DNS server. Click **Sync CCM Server from DNS Server** to open the Sync CCM Server from DNS Server page. Note the change sets to synchronize on this page, then click **Run**, followed by **Return**.

Figure 7-3 DNS Server Commands Page (Local Basic)

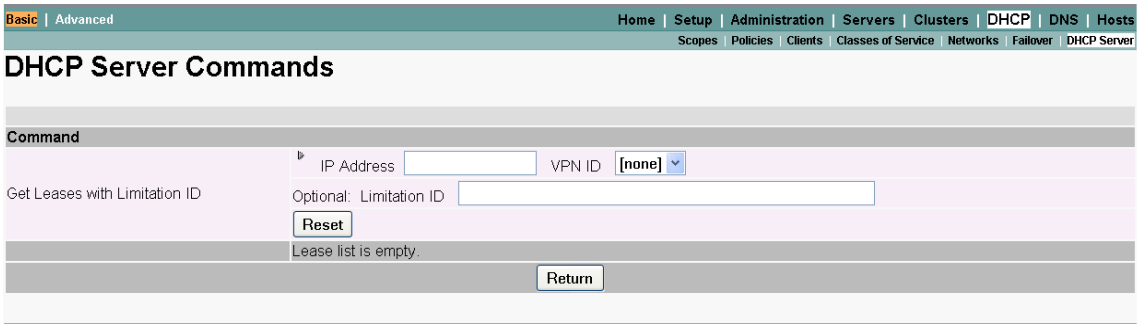


The server command functions are:

- **Flushing cache** (see the "Flushing DNS Cache" section on page 17-15)—Click the Run icon ( ). If you want to flush cache older than a certain time, enter the time in the Time field next to Flush cache older than, then click the Run icon. This is the equivalent of **dns flushCache** in the CLI.
- **Forcing all zone transfers** (see the "Enabling Zone Transfers" section on page 15-15)—Click the Run icon ( ). This is the equivalent of **dns forceXfer secondary** in the CLI.
- **Scavenging all zone transfers** (see the "Scavenging Dynamic Records" section on page 28-18)—Click the Run icon ( ). This is the equivalent of **dns scavenge** in the CLI.
- **Removing nonauthoritative resource record sets** (see the "Removing Cached Records" section on page 16-5)—Enter the name of the RR set in the Name field, then click the Run icon ( ). If you optionally want to remove nonauthoritative RR sets based on specific types or data, enter these values in the Type or Data field, respectively. This is the equivalent of **dns removeCachedRR** *name type data* in the CLI.

- **Local cluster DHCP administrator**—Click **DHCP**, then **DHCP Server** to open the Manage DHCP Server page. Along with the Statistics ( ), Log ( ), Start ( ), Stop ( ), and Reload ( ) functions, you can also perform other functions when you click the Run icon ( ) in the Commands column to open the DHCP Server Commands page (see Figure 7-4).

Figure 7-4 DHCP Server Commands Page (Local Basic)



This page provides the Get Leases with Limitation ID feature, to find clients that are associated through a common limitation identifier (see the "Administering Option 82 Limitation" section on page 24-18). Enter at least the IP address of the currently active lease in the IP Address field, then

click the Run icon ( ). You can also enter the limitation ID itself in the form *nn*:*nn*:*nn* or as a string ("*nnnn*"), in which case the IP address becomes the network in which to search. This function is the equivalent of **dhcp limitationList** *ipaddress limitation-id* **show** in the CLI.

CLI Commands

In the CLI, the regional cluster allows CCM server management only:

- To start the server, use **server** *type* **start** (or simply *type* **start**; for example, **dhcp start**).
- To stop the server, use **server** *type* **stop** (or simply *type* **stop**; for example, **dhcp stop**). If stopping the server, it is advisable to save it first using the **save** command.
- To reload the server, use **server** *type* **reload** (or simply *type* **reload**; for example, **dhcp reload**). Network Registrar stops the server you chose, loads the configuration data, and then restarts the server.
- To set or show attributes for the server, use [**server**] *type* **set** *attribute*=*value* or [**server**] *type* **show**. For example:

```
nrcmd> ccm set ipaddr=192.168.50.10
```

See Also

Scheduling Recurring Tasks
Logging Server Events, page 7-6
Log Files, page 7-8
Change Logs and Tasks, page 7-9
Monitoring and Reporting Server Status, page 7-10
Running Data Consistency Rules, page 7-19
Troubleshooting Servers, page 7-21

Scheduling Recurring Tasks

In Basic and Advanced user mode in the local cluster web UI, you can schedule a number of recurring tasks. These tasks are:

- Reloading the DHCP server.
- Reloading the DNS server.
- Synchronizing DHCP failover server pairs:
  - If in staged scope edit mode, reload the main DHCP server.
  - Synchronize the failover configuration to the backup DHCP server.
  - If in staged scope edit mode, reload the backup DHCP server.
- Synchronizing High-Availability (HA) DNS server pairs:
  - If in staged scope edit mode, reload the main DNS server.
  - Synchronize the HA DNS configuration to the backup DNS server.
  - If in staged scope edit mode, reload the backup DNS server.

- Synchronizing zone distribution maps:
  - If in staged scope edit mode, reload the main DNS server.
  - If in staged scope edit mode, reload the backup HA DNS server.
  - Synchronize the zone distribution maps.
  - If in staged scope edit mode, reload the secondary DNS server or servers.

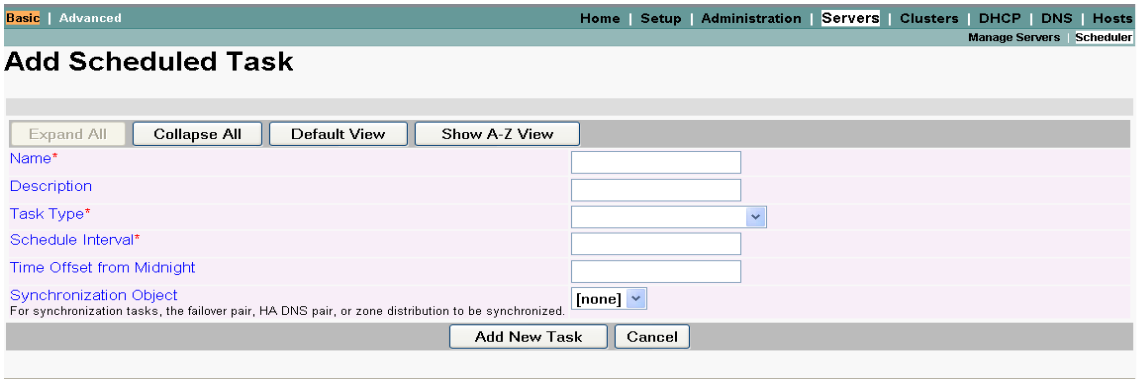
Local Basic or Advanced Web UI

To set up one or more of these recurring server tasks:

Step 1 Click **Servers**, then **Scheduler** to open the List Scheduled Tasks page.

Step 2 Click **Add Task** to open the Add Scheduled Task page (see Figure 7-5).

Figure 7-5 Add Scheduled Task Page (Local Basic)



Step 3 Enter values in the appropriate fields:

a. Name of the scheduled task. This can be any identifying text string.

b. Description of the task.

c. Pull down from the available list of task types, which are:

- **dhcp-reload**—Reloads the DHCP server
- **dns-reload**—Reloads the DNS server
- **sync-dhcp-pair**—Synchronizes the DHCP failover server pair
- **sync-dns-pair**—Synchronizes the HA DNS failover server pair
- **sync-zd-map**—Synchronizes zone distribution maps
- **sync-dns-update-map**—Synchronizes DNS update maps

d. In Advanced mode, select to enable or disable the task (the preset value is enabled in both modes).

e. Indicate the time interval for the scheduled task, such as 60m or 4w2d.

f. Indicate any scheduled offset, or when you want the task scheduled at a fixed time of day, as an hour from 12:00 midnight. The scheduled interval must be less than 24h, and the offset must be less than the interval. For example, if you set the *scheduled-interval* to 4h and the *scheduled-offset* to 2, the task would occur at 2:00 A.M., 6:00 A.M., 10:00 A.M., 2:00 P.M., 6:00 P.M., and 10:00 P.M.

g. If you are synchronizing a DHCP failover pair, HA DNS pair, or zone distribution map, you can also choose the relevant object identifier associated with the task.

Step 4 Click **Add New Task**.

Step 5 If you click the name of the task on the List Scheduled Tasks page, on the Edit Scheduled Task page you can view (in the Task Status section) the last status or the list of last errors (if any) that occurred during the task execution. Click **Run Now** to run the task immediately.

Logging Server Events

When you start Network Registrar, it automatically starts logging Network Registrar system activity. Network Registrar maintains all the logs by default on:

- **Windows**—*install-path*\logs
- **Solaris and Linux**—*install-path*/logs (to view these logs, use the **tail -f** command)

Tip To avoid filling up the Windows Event Viewer and preventing Network Registrar from running, in the Event Log Settings, check the **Overwrite Events as Needed** box. If the events do fill up, save them to a file, then clear them from the Event Log.

Local Basic or Advanced and Regional Web UI

Server logging is available in the web UI when you open the Manage Servers page for a server (see the "Managing Servers" section on page 7-1), then click the Log icon ( ) in the View Log column for the server. This opens the Log for Server page. The log is in chronological order with the page with the latest entries shown first. If you need to see earlier entries, click the left arrow at the top or bottom of the page.

See Also

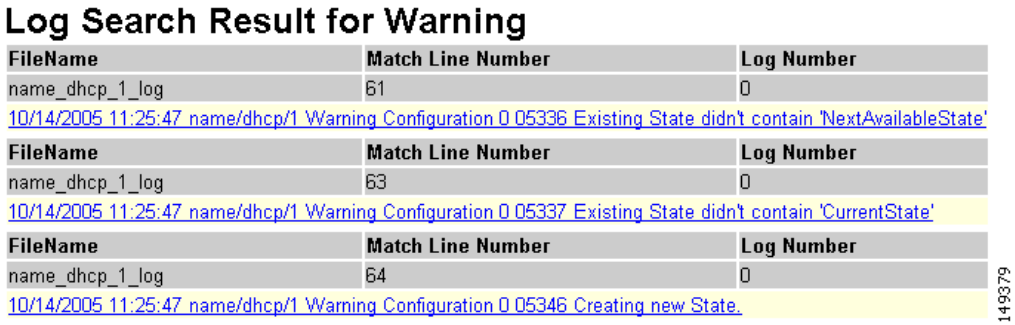
Searching the Logs, page 7-6
Logging Format and Settings, page 7-7

Searching the Logs

The web UI provides a convenient way to search for entries in the activity and startup log files. You can locate specific message text, log message IDs, and message timestamps using a regular expression string entry. When you click the Log icon ( ) in the View Log or View Startup column on the Manage Servers page (or one of the specific server pages), this opens a Log for Server page. In the text field next to the Search icon ( ) at the top or bottom of the page, enter the search string in the regular expression syntax. (For example, entering **name?** searches for occurrences of the string "name" in the log file.)

Clicking the Search icon ( ) opens a Log Search Result page in a separate browser window (see Figure 7-6 on page 7-7 for an example). The page shows the log file, line number of the match, and the log number.

Figure 7-6 Log Search Result Page (Local)



Click the name of the log message, which opens the Log for Server page with the full message text. Change between Table and Text view by clicking the Log icon ( ). Click **Close** on the Log Search Result page to close the browser window.

Logging Format and Settings

The server log entries include the following categories:

- **Activity**—Logs the activity of your servers.
- **Info**—Logs standard operations of the servers, such as starting up and shutting down.
- **Warning**—Logs warnings, such as invalid packets, user miscommunication, or an error in a script while processing a request.
- **Error**—Logs events that prevent the server from operating properly, such as out of memory, unable to acquire resources, or errors in configuration.

Note Warnings and errors go to the Event Viewer on Windows (see the Tip on page 7-6). For a description of the log messages for each server module, see the *install-path*/docs/msgid/MessageIdIndex.html file.

Local Basic or Advanced and Regional Web UI

You can affect which events to log. For example, to set the logging for the local cluster DNS and DHCP server:

- **DNS**—Click **DNS**, then **DNS Server** to open the Manage DNS Server page. Click the name of the server to open the Edit DNS Server page. Expand the Logging attributes section to view the log settings. Make changes to these settings as desired, click **Modify Server**, then reload the server. (See Table 17-2 on page 17-19 for the log settings to maximize DNS server performance.)
- **DHCP**—Click **DHCP**, then **DHCP Server** to open the Manage DHCP Server page. Click the name of the server to open the Edit DHCP Server page. Expand the Logging section to view the log settings. Make changes to these settings as desired, click **Modify Server**, then reload the server. (See Table 23-2 on page 23-12 for the log settings to maximize DHCP server performance.)

CLI Commands

Use **dns set log-settings**, **dhcp set log-settings**, and **tftp set log-settings** for the respective servers.

Log Files

Table 7-1 describes the Network Registrar log files in the *install-path*/logs directory.

Table 7-1 Log Files in .../logs Directory

Component	File in /logs Directory	Local/Regional	Logs
Installation	install_cnr_log	Both	Installation process
Upgrade	mcdupgrade_log	Both	Upgrade process
Server agent	agent_server_1_log	Both	Server agent starts and stops
Port check	checkports_log	Both	Network ports
DNS server	name_dns_1_log	Local	DNS activity
DHCP server	name_dhcp_1_log	Local	DHCP activity
TFTP server	file_tftp_1_log file_tftp_1_trace	Local	TFTP activity
SNMP server	cnrsnmp_log	Local	SNMP activity
RIC server	ric_server_log	Both	RIC server activity
CCM database	config_ccm_1_log	Both	CCM configuration, starts, stops
Web UI	cnrwebui_log	Both	Web UI state
Tomcat/web UI (in cnrwebui subdirectory)	catalina_log.*date*.txt jsui_log.*date*.txt localhost_access_log.*date*.txt	Both	CCM database for Tomcat server and web UI (because new files are created daily, periodically archive old log files)

Each component can generate a number of log files, each with a preconfigured maximum size of 1 MB. The first log file name has the _log suffix. When this file reaches its maximum size, it gets the .01 version extension appended to its name and a new log file is created without the version extension. Each version extension is incremented by one for each new file created. When the files reach their configured maximum number, the oldest file is deleted and the next oldest assumes its name. The usual maximum number is four for the DNS, DHCP, and TFTP servers.

Note Some user commands can create *User authentication* entries in the Server Agent log because of separate connections to the cluster. Do not interpret these as a system security violation by another user.

CLI Commands

You can check the configured maximums for the DNS, DHCP, and TFTP servers using [**server**] *type* **serverLogs show** in the CLI, which shows the maximum number (*nlogs*) and size (*logsize*) of these protocol server log files. You can adjust these parameters using [**server**] *type* **serverLogs set nlogs=***value* and [**server**] *type* **serverLogs set logsize=***value*. You cannot adjust these maximums for any of the other log files.

Change Logs and Tasks

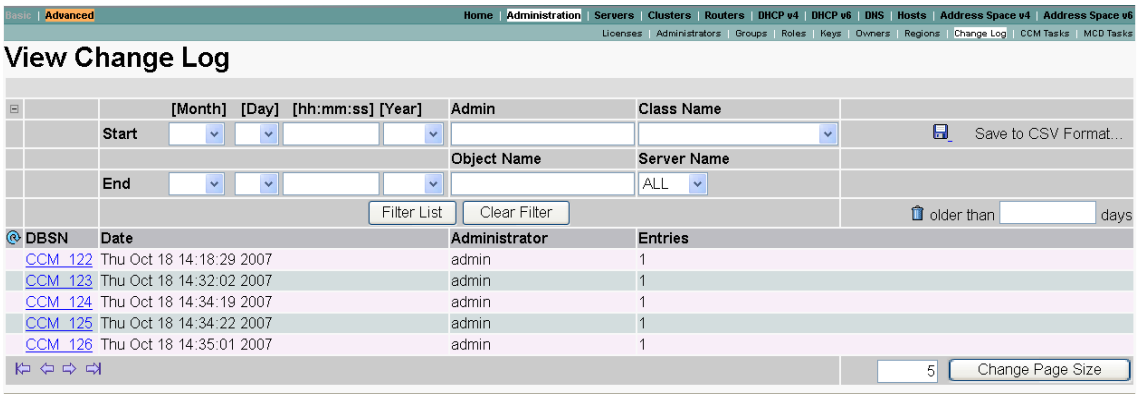
In the web UI, you can view the change logs and tasks associated with configurations you make.

Local Advanced and Regional Web UI

Click **Administration**, then **Change Log**, **CCM Tasks**, or **MCD Tasks**. To view the change log and tasks, you must be assigned the database subrole of the ccm-admin or regional-admin role:

- The View Change Log page (see Figure 7-7) shows all the change logs, sorted by DBSN name. To get to the bottom of the list, click the right arrow at the bottom left of the page. Click the DBSN number of the change log entry to open a View Change Set page for it.

Figure 7-7 View Change Log Page (Local Advanced)



- The List CCM Tasks page shows all the CCM tasks.
- The List MCD Tasks page shows all the MCD tasks.

On the View Change Log page, you can filter the list, manually trim it, and save it to a file. You can filter the list by:

- Start and end dates
- Administrator who initiated the changes
- Configuration object class
- Specific object
- Object identifier (ID), in the format OID-00:00:00:00:00:00:00:00
- Server

Click **Filter List** or **Clear Filter** (to clear the filter that persists through the session). You can initiate a trim of the change log by setting how many days old you want the record to get before trimming it, by setting a number of days value in the “older than” field and clicking the Delete icon ( ).

To save the change log entries to a comma-separated values (CSV) file, click the Save icon ( ).

If a task is associated with a change log, it appears on the View Change Set page. You can click the task name to open the View CCM Task or View MCD Task page for it.

Monitoring and Reporting Server Status

Monitoring the status of a server involves checking its:

- State
- Health
- Statistics
- Log messages
- Address usage
- Related servers (DNS and DHCP)
- Leases (DHCP)

See Also

Server StatesDisplaying Health, page 7-10
Displaying Statistics, page 7-11
Displaying IP Address Usage, page 7-15
Displaying Related Servers, page 7-16
Displaying Leases, page 7-18

Server States

All Network Registrar protocol servers (DNS, DHCP, SNMP, and TFTP) pass through a state machine consisting of the following states:

- **Loaded**—First step after the server agent starts the server (transitional).
- **Initialized**—Server was stopped or fails to configure.
- **Unconfigured**—Server is not operational because of a configuration failure (transitional).
- **Stopped**—Server was administratively stopped and is not running (transitional).
- **Running**—Server is running successfully.

The two essential states are initialized and running, because the server transitions through the states so quickly that the other states are essentially invisible. Normally, when the server agent starts the server, it tells the server to be up. The server process starts, sets its state to loaded, then moves up to running. If you stop the server, it walks down the states to initialized, and if you restart, it moves up to running again. If it fails to configure for some reason, it drops back to initialized, as if you had stopped it.

There is also an exiting state that the server is in very briefly when the process is exiting. The user interface can also consider the server to be disabled, but this rarely occurs and only when there is no server process at all (the server agent was told not to start one).

Displaying Health

You can display aspects of the health of a server, or how well it is running. The following items can decrement the server health, so you should monitor their status periodically. For the:

- Server agent (local and regional clusters)
- CCM server (local and regional clusters)

- DNS server (local cluster):
  - Configuration errors
  - Memory
  - Disk space usage
  - Inability to contact its root servers
- DHCP server (local cluster):
  - Configuration errors
  - Memory
  - Disk space usage
  - Packet caching low
  - Options not fitting in the stated packet limit
  - No more leases available
- TFTP server (local cluster):
  - Memory
  - Socket read or write error
  - Exceeding the overload threshold and dropping request packets
- RIC server (regional cluster)

Tip Health values range from 10 (the highest level of health) to 0 (the server is not running). Use the existence of any descending health values as a reminder to check the log files for the server.

On Solaris or Linux, you can run the **cnr_status** command, in the *install-path*/usrbin/ directory, to see if your local cluster server is running. See the *Installation Guide for Cisco Network Registrar*.

Local Basic or Advanced and Regional Web UI

Click **Administration**, then **Servers**. Check the Manage Servers page for the state and health of each server (see Figure 7-1 on page 7-2 for a DHCP server example).

CLI Commands

Use [**server**] *type* **getHealth**. The number 10 indicates the highest level of health, 0 that the server is not running.

Displaying Statistics

To display server statistics, the server must be running.

Local Basic or Advanced and Regional Web UI

Go to the Manage Servers page, then click the Statistics icon (), if available, in the Statistics column. On the Server Statistics page, click the name of the attribute to get popup help.

The DHCP and DNS statistics are each divided into two groups of statistics. The first group is for total statistics and the second group is for sample statistics. The total statistics are accumulated over time. The sample statistics occur during a configurable sample interval. The names of the two categories vary per server and per user interface, and are identified in Table 7-2.

Table 7-2 Server Statistics Categories

Server	User Interface	Total Statistics (Command)	Sample Statistics (Command)
DHCP	Web UI	Total Statistics	Activity Summary
	CLI	Total Counters since the start of the last DHCP server process (**dhcp getStats**)	Sampled counters since the last sample interval (**dhcp getStats sample**)
DNS	Web UI	Total Statistics	Sample Statistics
	CLI	Total Counters since the start of the last server process (**dns get Stats**)	Sampled counters since the last sample interval (**dns getStats sample**)

To set up the sample counters, you must activate either the *collect-sample-counters* attribute for the server or a *log-settings* attribute value called activity-summary. You can also set a *log-settings* value for the sample interval for each server, which is preset to 5 minutes. The *collect-sample-counters* attribute is preset to true for the DNS server, but is preset to false for the DHCP server. For example, to enable the sample counters and set the interval for DHCP, set the following attributes for the DHCP server:

- Enable *collect-sample-counters* (**dhcp enable collect-sample-counters**)
- Set *log-settings* for activity-summary (**dhcp set log-settings=activity-summary**)
- Set *activity-summary-interval* to 5m (**dhcp set activity-summary-interval=5m**)

CLI Commands

In the CLI, if you use [**server**] *type* **getStats**, the statistics are encoded in curly braces followed by sets of digits, as described in Table 7-3 on page 7-13 for DNS, Table 7-4 on page 7-14 for DHCP, and Table 7-5 on page 7-14 for TFTP. The **server** *type* **getStats all** command is more verbose and identifies each statistic on a line by itself. Using the additional **sample** keyword shows the sample statistics only.

Reset the counters and total statistic by using **dhcp resetStats** or **dns resetStats**.

See Also

DNS Statistics, page 7-12
DHCP Statistics, page 7-13
TFTP Statistics, page 7-14

DNS Statistics

The DNS server statistics in the web UI appear on the DNS Server Statistics page, and you can click each statistic name to open a description for it.

The CLI **dns getStats** command has the following options:

```
dns getStats [performance | query | errors | security | maxcounters | ha | ipv6 | all]
[total | sample]
```

The **dns getStats all** command is the most commonly used. The **dns getStats** command without this option returns the statistics in a single line of positional values in the following format (Table 7-3 shows how to read these values):

```
nrcmd> dns getStats
100 Ok
{1} 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```

Table 7-3 DNS Statistics

Digit	Statistic	Description
{1}	id	Implementation ID (release and build information).
2	config-recurs	Recursion services—(1) available, (2) restricted, (3) unavailable.
3	config-up-time	Time (in seconds) elapsed since the last server startup.
4	config-reset-time	Time (in seconds) elapsed since the last server reset (restart).
5	config-reset	Status or action to reinitializes any name server state—If using the (2) reset action, reinitializes any persistent name server state; the following are read-only statuses: (1) other—server in some unknown state, (3) initializing, or (4) running.
6	counter-auth-ans	Number of queries answered authoritatively.
7	counter-auth-no-names	Number of queries returning `authoritative no such name` responses.
8	counter-auth-no-data-resps	Number of queries returning `authoritative no such data` (empty answer) responses.
9	counter-non-auth-datas	Number of queries answered nonauthoritatively (cached).
10	counter-non-auth-no-datas	Number of queries answered nonauthoritatively with no data.
11	counter-referrals	Number of queries forwarded to other servers.
12	counter-errors	Number of responses answered with errors (RCODE values other than 0 or 3).
13	counter-rel-names	Number of requests received for names of only one label (relative names).
14	counter-req-refusals	Number of refused queries.
15	counter-req-unparses	Number of unparsable requests.
16	counter-other-errors	Number of aborted requests due to other errors.
17	total-zones	Total number of configured zones.

DHCP Statistics

The DHCP server statistics in the web UI appear on the DHCP Server Statistics page, and you can click each statistic name to open a description for it.

The CLI **dhcp getStats** command has the following options:

```
dhcp getStats [[all | server [,] failover [,] dhcpv6] [total | sample]
```

The **dhcp getStats all** command is the most commonly used. The **dhcp getStats** command without this option returns the statistics in a single line of positional values in the following format (Table 7-4 shows how to read these values):

```
nrcmd> dhcp getStats
100 Ok
{1} 2 3 4 5 6 7 8
```

Table 7-4 DHCP Statistics

Digit	Statistic	Description
{1}	start-time-str	Date and time of last server reload, as a text string.
2	total-discovers	Number of DISCOVER packets received.
3	total-requests	Number of REQUEST packets received.
4	total-releases	Number of RELEASED packets received.
5	total-offers	Number of OFFER packets sent.
6	total-acks	Number of acknowledgement (ACK) packets sent.
7	total-naks	Number of negative acknowledgement (NAK) packets sent.
8	total-declines	Number of DECLINE packets received.

TFTP Statistics

The TFTP server statistics in the web UI appear on the TFTP Server Statistics page, and you can click each statistic name to open a description for it. Table 7-5 shows the TFTP statistics encoded as output to the generic **tftp getStats** command, in the format:

```
nrcmd> tftp getStats
100 Ok
{1} 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
```

Table 7-5 TFTP Statistics

Digit	Attribute	Description
{1}	id	Implementation ID (release and build information).
2	server-state	State of the server (up or down).
3	server-time-since-start	Running time since last start.
4	server-time-since-reset	Running time since last reset.
5	total-packets-in-pool	Number of packets in the pool.
6	total-packets-in-use	Number of packets the server is using.
7	total-packets-received	Number of packets received since the last start or reload.
8	total-packets-sent	Number of packets sent since the last start or reload.
9	total-packets-drained	Number of packets read and discarded since the last start or reload.
10	total-packets-dropped	Number of packets dropped since the last start or reload.
11	total-packets-malformed	Number of packets received that were malformed since the last start or reload.

Table 7-5 TFTP Statistics (continued)

Digit	Attribute	Description
12	total-read-requests	Number of packets read since the last start or reload.
13	total-read-requests-completed	Number of read packets completed since the last start or reload.
14	total-read-requests-refused	Number of read packets refused since the last start or reload.
15	total-read-requests-ignored	Number of read packets ignored since the last start or reload.
16	total-read-requests-timed-out	Number of read packets that timed out since the last start or reload.
17	total-write-requests	Number of read packets that were write requests since the last start or reload.
18	total-write-requests-completed	Number of write requests completed since the last start or reload.
19	total-write-requests-refused	Number of write requests refused since the last start or reload.
20	total-write-requests-ignored	Number of write requests ignored since the last start or reload.
21	total-write-requests-timed-out	Number of write requests that timed out since the last start or reload.
22	total-docsis-requests	Number of DOCSIS requests received since the last start or reload.
23	total-docsis-requests-completed	Number of DOCSIS requests completed since the last start or reload.
24	total-docsis-requests-refused	Number of DOCSIS requests refused since the last start or reload.
25	total-docsis-requests-ignored	Number of DOCSIS requests ignored since the last start or reload.
26	total-docsis-requests-timed-out	Number of DOCSIS requests that timed out since the last start or reload.
27	read-requests-per-second	Number of read requests per second.
28	write-requests-per-second	Number of write requests per second.
29	docsis-requests-per-second	Number of DOCSIS requests per second.

Displaying IP Address Usage

Displaying IP address usage gives a good overview of how clients are currently assigned addresses.

Local Advanced and Regional Web UI

You can look at the local or regional cluster address space, or generate a subnet utilization or lease history report at the regional cluster, to determine IP address usage. These functions are available in both web UIs by clicking **Address Space**, if you have address space privileges at the local or regional cluster.

You can determine the current address space utilization by clicking the View icon ( ) in the Current Usage column for the unified address space, address block, and subnet (see the "Viewing Address Utilization for Address Blocks, Subnets, and Scopes" section on page 9-12). You can also get the most current IP address utilization by querying the lease history (see the "Querying Leases" section on

page 22-29). In the latter case, the regional CCM server references the appropriate DHCP server directly. To ensure this subnet-to-server mapping, you must update the regional address space view so that it is consistent with the relevant local cluster. Do this by pulling the replica address space, or reclaiming the subnet to push to the DHCP server (see the "Reclaiming Subnets" section on page 9-10). Also ensure that the particular DHCP server is running.

CLI Commands

You can generate an IP address usage report using the **report** command. The command has the following syntax:

report [**column-separator=***string* | **dhcpv4** | **dhcp-only** | **dhcpv6** | **file=***outputfile* | **vpn=***name*

The column-separator specifies the character string that separates the report columns (the preset value is the space character). If you want to include more than one space, precede them with the backslash (\) escape character (enclosed in quotation marks). You can specify DHCPv4 or DHCPv6 addresses (**dhcp-only** is the same as **dhcpv4**). Not specifying the VPN returns the addresses in the current VPN only.

Displaying Related Servers

Network Registrar displays the relationship among servers in a DNS zone distribution or a DHCP failover configuration. In the web UI, you can view a related servers page when you click the Related Servers icon ( ) on various pages. You can use the display of related servers to diagnose and monitor misconfigured or unreachable servers.

See Also

Monitoring Remote Servers Using Persistent Events
DNS Zone Distribution Servers, page 7-17
DHCP Failover Servers, page 7-18

Monitoring Remote Servers Using Persistent Events

To service clients that require updates to DNS and LDAP related servers, the DHCP server uses a persistent event algorithm to ensure updates to related servers when a related server is temporarily unavailable. In addition, the algorithm prevents a misconfigured or offline DNS server from using up all the available update resources.

At startup, the DHCP server calculates the number of related servers in the configuration that require persistent events. A preconfigured Maximum Pending Events attribute (an Expert mode attribute that specifies the number of in-memory events that is preset to 40,000) is divided by the number of servers to obtain a limit on the number of events permitted for each remote server. This calculation covers related DNS and LDAP servers (DHCP failover does not use persistent storage for events). The DHCP server uses this calculation to issue log messages and take the action described in Table 7-6 on page 7-17. The table shows a hypothetical case of a DHCP server with four related DNS servers each having a limit of 10K events.

Table 7-6 Persistent Event Algorithm

Event Reached	DHCP Server Action
50% of the calculated per-server limit (Maximum Pending Events value divided by the number of total related servers); for example, 5K events on a related server out of a total of 40K maximum pending events	Issues an INFO log message every 2 minutes, as long as the limits are exceeded: `The queue of events for the name remote server at address has x events, and has reached the info limit of y/2 events out of an upper limit of y events per remote server. The remote server may be misconfigured, inoperative, or unreachable.`
100% of the calculated per-server limit and less than 50% of the Maximum Pending Events value; for example, 10K events on a related server, with fewer than 10K total maximum pending events	Issues a WARNING log message every 2 minutes, as long as the limits are exceeded: `The queue of events for the name remote server at address has x events, has exceeded the limit of y events per remote server, but is below the limit of z total events in memory. The remote server may be misconfigured, inoperative, or unreachable.`
100% of the calculated per-server limit and 50% or more of the Maximum Pending Events value; for example, 10K events on a related server, with 20K total maximum pending events	Issues an ERROR log message every 2 minutes, as long as the limits are exceeded: `The queue of events for the name remote server at address has x events, and has grown so large that the server cannot continue to queue new events to the remote server. The limit of y events per remote server and z/2 total events in memory has been reached. This and future updates to this server will be dropped. The current eventID n is being dropped.` The server drops the current triggering event and all subsequent events with that server.
100% of the Maximum Pending Events value; for example, 40K events across all related servers	Issues an ERROR log message: `The queue of pending events has grown so large that the server cannot continue to queue new events. The queue's size is z, and the limit is z.` The server drops all subsequent events with all related servers.

SNMP traps and DHCP server log messages also provide notification that a related server is unreachable.

DNS Zone Distribution Servers

A DNS zone distribution simplifies creating multiple zones that share the same secondary server attributes. You can view and set the primary and secondary DNS servers in a zone distribution.

Local Basic or Advanced Web UI

Click **DNS**, then **Zone Distribution**. This opens the List Zone Distributions page. The local cluster allows only one zone distribution, the default. Click this zone distribution name to open the Edit Zone Distribution page, which shows the authoritative and secondary servers in the zone distribution.

Regional Web UI

Click **DNS**, then **Zone Distributions**. This opens the List/Add Zone Distributions page (see Figure 15-7 on page 15-21). The regional cluster allows creating more than one zone distribution. Click the zone distribution name to open the Edit Zone Distribution page, which shows the primary, authoritative, and secondary servers in the zone distribution.

CLI Commands

Create a zone distribution using **zone-dist** *name* **create** *primary-cluster*, then view it using **zone-dist list**. For example:

```
nrcmd> zone-dist distr-1 create Boston-cluster
nrcmd> zone-dist list
```

DHCP Failover Servers

Related servers in a DHCP failover pair relationship can show the following information:

- **Type**—Main or backup DHCP server.
- **Server name**—DNS name of the server.
- **IP address**—Server IP address in dotted octet format.
- **Requests**—Number of outstanding requests, or two dashes if not applicable.
- **Communication status**—OK or INTERRUPTED.
- **Cluster state**—Failover state of this DHCP server.
- **Partner state**—Failover state of its partner server.

For details on DHCP failover implementation, see Chapter 27, "Configuring DHCP Failover."

Local Basic or Advanced Web UI

Click **DHCP**, then **Failover**. The List DHCP Failover Pairs page shows the main and backup servers in the failover relationship.

CLI Commands

Use **dhcp getRelatedServers** to display the connection status between the main and partner DHCP servers. If there are no related servers, the output is simply `100 Ok`.

Displaying Leases

After you create a scope, you can monitor lease activity and view lease attributes.

Local Basic or Advanced Web UI

Click **DHCP**, then **Scopes** or **Prefixes** (in Advanced mode). On the List/Add DHCP Scopes or List DHCPv6 Prefixes page, click the View icon () in the Leases column to open the List DHCP Leases for Scope or List DHCP Leases for Prefix page.

Regional Web UI

Click **Address Space**, then **Lease History**. Set the query parameters, then click **Query Lease History**. (See the "Querying Leases" section on page 22-29.)

Running Data Consistency Rules

Using consistency rules, you can check such data inconsistencies as overlapping address ranges and subnets. You can set data consistency rules at the regional and local clusters. See Table 7-7 and Table 7-8 for the consistency rules you can set in each case.

Table 7-7 Data Consistency Rule Settings at the Regional Cluster

Regional Consistency Rule	Description
Attributes Consistency Rule	Ensures that policies and client-classes with the same names at replication have the same values.
Broadcast Address Rule	Ensures that the dynamic address range for a scope should not include a broadcast address.
Cable Helper Consistency Rule	Ensures that cable helpers and DHCP failover are consistent.
Client-Class Selection Match Tag Rule	Ensures that all client-classes have a scope-selection tag defined by some scope.
Ensure Scope for Subnet Rule	Ensures that a scope is present for every subnet.
Ensure Subnets for Scopes Rule	Ensures that a subnet is present for every scope.
IP Range Consistency Rule	Identifies any overlapping static or dynamic address ranges.
Owner Match Rule	Ensures that each subnet on the router has the same owner as the subinterface.
Router Subnets in Database Rule	Ensures that each subnet on the router has a corresponding subnet in the regional database.
Selection Tags Consistency Rule	Ensures that scope-selection tags on scopes match with one of those defined in the address types.
Subnet Consistency Rule	Identifies any overlapping subnets.
Utilization Collection Rule	Ensures that the DHCP server collection interval is greater than the regional server collection interval.

Table 7-8 Data Consistency Rule Settings at the Local Cluster

Local Consistency Rule	Description
Broadcast Address Rule	Ensures that the dynamic address range for a scope should not include a broadcast address.
CNAME RR and Host Alias Consistency Rule	Ensures that CNAME records and host alias names are consistent.

Table 7-8 Data Consistency Rule Settings at the Local Cluster (continued)

Local Consistency Rule	Description
Client-Class Selection Match Tag Rule	Ensures that all client-classes have a scope-selection tag defined by some scope.
Failover Pair Consistency Rule	Ensures that the CCM server and DHCP server have the same versions of the DHCP failover pair objects.
IP Range Consistency Rule	Identifies any overlapping static or dynamic address ranges.
Lame Delegation Rule	Ensures that no lame delegation exists for managed subzones.
Missing Hosts Consistency Rule	Identifies any missing hosts for PTR records.
Missing PTR Records Consistency Rule	Identifies any missing PTR records for hosts.
Owner Match Rule	Ensures that each subnet on the router has the same owner as the subinterface.
Router Subnets in Database Rule	Ensures that each subnet on the router has a corresponding subnet in the regional database.
Subnet Consistency Rule	Identifies any overlapping subnets.
Zone Reference Rule	Ensures the resolvability of DNS update policies, access control lists (ACLs), and encryption key objects referenced from zones.

Local Basic or Advanced and Regional Web UI

Step 1 Click **Home**, then **Consistency Rules** to open the List Consistency Rules page (see Figure 7-8 on page 7-21 for a local cluster version of this page).

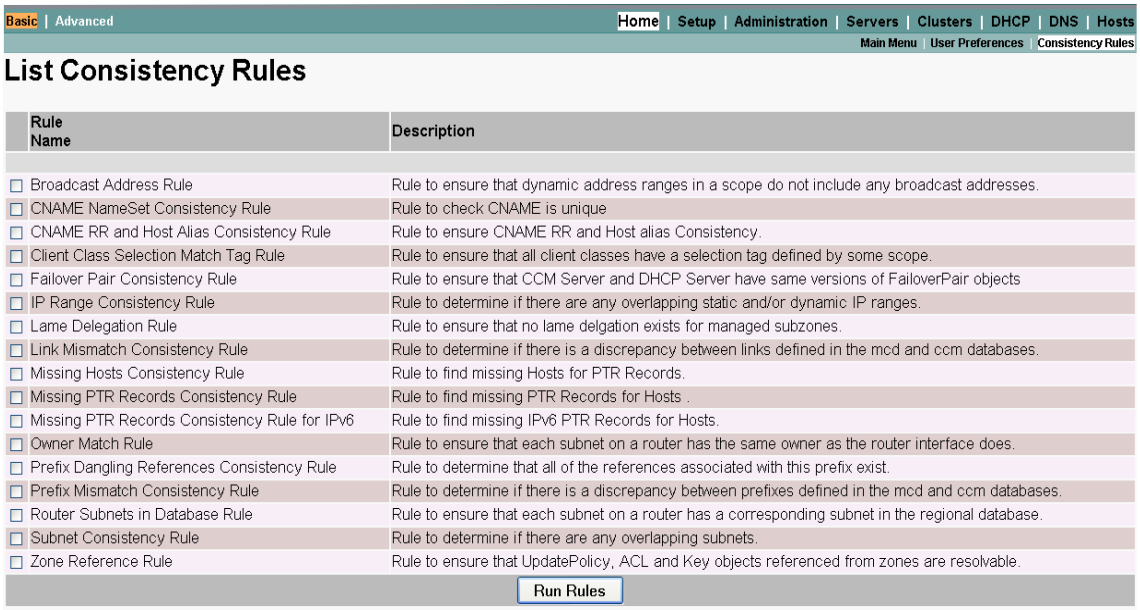
Step 2 Click check marks for each of the listed consistency rules that you want to apply (see Table 7-7 on page 7-19 and Table 7-8 on page 7-19).

Step 3 Click **Run Rules**. This opens the Consistency Rules Result page. If the CCM server finds violations, the page lists them for each rule.

Step 4 To view the network objects that caused the violations, click the **+** sign next to Detailed Objects under the violation. The result shows the object class name, object ID, sequence number, and the values impacted by the inconsistency.

Step 5 Click **Return** to return to the List Consistency Rules page.

Figure 7-8 List Consistency Rules Page (Local Basic)



Troubleshooting Servers

The following sections describe troubleshooting the configuration and the DNS, DHCP, and TFTP servers.

See Also

Immediate Troubleshooting Actions, page 7-21
Modifying the cnr.conf File, page 7-22
Troubleshooting Server Failures, page 7-22
Troubleshooting and Optimizing the TFTP Server, page 7-23
Solaris and Linux Troubleshooting Tools, page 7-25
Using the TAC Tool, page 7-25

Immediate Troubleshooting Actions

When facing a problem, it is crucial not to cause further harm while isolating and fixing the initial problem. Here are things to do (or avoid doing) in particular:

- Have 512 MB or more of memory and 2.5 GB or more of a data partition.
- Do not reboot a cable modem termination system (CMTS).
- Enable DHCP failover.
- Do not reload, restart, or disrupt Network Registrar with failover resynchronization in progress.

Modifying the cnr.conf File

Network Registrar uses the **cnr.conf** file for basic configuration parameters. This file is normally located in the *install-path*/conf directory. Network Registrar creates the file during installation and processes it line by line.

You can edit this file if configuration parameters change. Note that during normal operation, you would not want to change the values, especially sensitive ones such as the MCD key. However, certain conditions might require you to modify certain values, such as when you move the data files for disk space reasons.

The format of the **cnr.conf** file consists of parameter name-value pairs, one per line; for example, for a Windows local cluster installation:

```
cnr.rootdir=C:\\Program Files\\Network Registrar\\Local
cnr.ccm-port=1234
cnr.datadir=C:\\Program Files\\Network Registrar\\Local\\data
cnr.java-home=C:\\Program Files\\Java\\jre1.5.0_12
cnr.logdir=C:\\Program Files\\Network Registrar\\Local\\logs
cnr.https-port=8443
cnr.tempdir=C:\\Program Files\\Network Registrar\\Local\\temp
cnr.http-port=8080
cnr.ccm-mode=local
cnr.mcdpath=ccmsrv
cnr.backup-time=23:45
cnr.mcd-key=01A58EE87DD4EE78368805FD366F8505446FAD449D6C7A
```

Directory paths must be in the native syntax for the operating system. The format allows the use of colons (:) in directory paths, but not as name-value pair separators; it does not allow line continuation or embedded unicode characters. Other modifications to the file might include the location of the log directory (see the "Log Files" section on page 7-8) or the time **mcdshadow** backups should occur (see the "Setting Shadow Backup Times" section on page 8-3).

Troubleshooting Server Failures

The server agent processes (nwreglocal and nwregregion) normally detect server failures and restart the server. You can usually recover from the failure and the server is not likely to fail again immediately after restarting. On rare occasions, the source of the server failure prevents the server from successfully restarting, and the server fails again as soon as it restarts. In such instances, perform the following steps:

Step 1 If the server takes a significantly long time to restart, stop and restart the server agent. On:

- Windows:

  net stop nwreglocal or **nwregregion**
  net start nwreglocal or **nwregregion**

- Solaris:

  /etc/init.d/nwreglocal stop or **nwregregion stop**
  /etc/init.d/nwreglocal stop or **nwregregion start**

- Linux:

  /etc/rc.d/init.d/nwreglocal stop or **nwregregion stop**
  /etc/rc.d/init.d/nwreglocal stop or **nwregregion start**

Step 2 Keep a copy of all the log files. Log files are located in the *install-path*/logs directory on Solaris and Linux, and the *install-path*\logs folder on Windows. The log files often contain useful information that can help isolate the cause of a server failure.

Step 3 Use the TAC tool, as described in the "Using the TAC Tool" section on page 7-25, or save the core or user.dmp file, if one exists, depending on the operating system:

- **Windows**—The user.dmp file is located in the system directory, which varies depending on the Windows system. Search for this file and save a renamed copy.
- **Solaris and Linux**—The core file is located in the *install-path*. Save a renamed copy of this file that Network Registrar does not overwrite.

Step 4 On Windows, use the native event logging application to save the System and Application event logs to files. You can do this from the Event Viewer. These event logs often contain data that helps debug Network Registrar server problems. For a description of the log messages for each server module, see the *install-path*/docs/msgid/MessageIdIndex.html file.

Troubleshooting and Optimizing the TFTP Server

You can set certain attributes to troubleshoot and optimize TFTP server performance.

See Also

Tracing TFTP Server Activity, page 7-23
Optimizing TFTP Message Logging, page 7-24
Enabling TFTP File Caching, page 7-24

Tracing TFTP Server Activity

To trace TFTP server activity, set the *packet-trace-level* attribute to a value of 1 through 4, depending on the level of verbosity you want the TFTP server to use to write messages to the trace file. The trace files are located in the /logs subdirectory of the installation directory. Windows tracing goes to the file_tftp_1_log file; Solaris and Linux tracing goes to the /var/nwreg2/{local | regional}/logs/file_tftp_1_log and file_tftp_1_trace files.

Here are the trace levels, with each higher level being cumulative:

- **0**—Disables all server tracing (the default).
- **1**—Displays all the log messages in the trace file.
- **2**—Displays the client IP address and port number for all packets.
- **3**—Displays the packet header information.
- **4**—Displays the first 32 bytes of the packet.

Note Setting and getting the trace level only works if the TFTP server is started. Turn on packet tracing only for debugging purposes, and then not for any extended time, for performance reasons.

Optimizing TFTP Message Logging

You can improve TFTP server performance by restricting logging and tracing. By default, the server logs error, warning, and informational messages to file_tftp_1_log files. You can set the log levels using a few TFTP server parameters:

- **Log level** (use the *log-level* attribute)—Master controller of server logging, which is preset to, and is best left at, level 3 (logs all error, warning, and informational messages). As with packet tracing, the higher logging levels are cumulative. If set to 0, no server logging occurs.
- **Log settings** (use the *log-settings* attribute)—This is the second level of logging control and takes only two values, *default* or *no-success-messages*. The *default* log setting does not alter the default value of log level 3 (error, warning, and informational messages). However, you may want to disable writing success informational messages, and thereby improve server performance, by changing the log settings to *no-success-messages*.
- **Log file count and size** (use the *log-file-count* attribute)—Sets how many log files to maintain and how large to allow them to get in the /logs directory. The default value is to maintain a maximum of four files of one megabyte each.

Note Reload the TFTP server after changing these values.

Enabling TFTP File Caching

You can improve TFTP server performance significantly by enabling file caching on the server. You must do this explicitly, because it is preset to disabled. You must also create and point to a file cache directory, and you can set the maximum size of this directory. Here are the steps:

Step 1 Determine where you want the TFTP cache files to go. This becomes a subdirectory of the TFTP home directory, which is preset to *install-path*/data/tftp (on Solaris and Linux, it is /var/nwreg2/{local | regional}/data/tftp). If you want a different location, set the *home-directory* attribute.

Step 2 Change to the TFTP home directory and create the cache directory, such as CacheDir, in the home directory, using the **mkdir Cachedir** command. Note that Network Registrar ignores any files in any subdirectories of this cache directory.

Step 3 Use the *file-cache-directory* attribute to set up the TFTP server to point to the cache directory. You cannot use relative paths in the directory name, such as .../cachedir. If the directory does not exist, file caching cannot occur.

Step 4 Use the *file-cache-max-memory-size* attribute to set the maximum memory size, in bytes, of the cache. The preset value is 32 KB. Network Registrar loads all files into cache that cumulatively fit this memory size. If you set the value to 0, Network Registrar does not cache any data, even if you enable file caching.

Step 5 Copy all of the files you want cached into the cache directory, and not into any subdirectory. Because all files in this directory are loaded into cache, do not include large files.

Step 6 Enable the *file-cache* attribute to enable file caching, then reload the server. Network Registrar logs the name of each cached file, and skips any it cannot load. It reads in all files as binary data and translates them as the TFTP client requests. For example, if a client requests a file as NetASCII, the client receives the cached data in that form.

Step 7 Writing to cache is not allowed. If you need to update a cache file, overwrite it in the cache directory, then reload the server.

Solaris and Linux Troubleshooting Tools

You can also use the following commands on Solaris and Linux systems to troubleshoot Network Registrar. To:

- See all Network Registrar processes:

  ps -leaf | grep nwr

- Monitor system usage and performance:

  top
  vmstat

- View login or bootup errors:
  - On Solaris—**grep /var/adm/messages***
  - On Linux—**grep /var/log/messages***
- View the configured interfaces and other network data:

  ifconfig -a

Using the TAC Tool

There may be times when any amount of troubleshooting steps will not resolve your problem and you have to resort to contacting the Cisco Technical Assistance Center (TAC) for help. Network Registrar provides a tool so that you can easily assemble the server or system error information, and package this data for TAC support engineers. This eliminates having to manually assemble this information with TAC assistance. The resulting package from this tool provides the engineers enough data so that they can more quickly and easily diagnose the problem and provide a solution.

The **cnr_tactool** utility is available in the bin directory of the Windows, and usrbin directory of the UNIX or Linux, installation directories. Execute the **cnr_tactool** utility:

> **cnr_tactool -N** *username* **-P** *password* [**-d** *output-directory*]

The output directory is optional and normally is the temp directory of the installation directories (in the /var path on Solaris and Linux). If you do not supply the username and password on the command line, you are prompted for them:

```
> cnr_tactool
username:
password:
[processing messages....]
```

The tool generates a packaged tar file whose name includes the date and version. The tar file contains all the diagnostic files.



CHAPTER 8

Backup and Recovery

This chapter explains how to maintain the Cisco Network Registrar databases.

Backing Up Databases

Because the Network Registrar databases do a variety of memory caching and can be active at any time, you cannot rely on third-party system backups to protect the database. They can cause backup data inconsistency and an unusable replacement database.

For this purpose, Network Registrar provides a shadow backup utility, **mcdshadow**. Once a day, at a configurable time, Network Registrar takes a snapshot of the critical files. This snapshot is guaranteed to be a consistent view of the databases.

See Also

Syntax and Location, page 8-2
Backup Strategy, page 8-2
Database Recovery Strategy, page 8-3
Backing Up and Recovering MCD Data, page 8-4
Backing Up CNRDB Data, page 8-7
Recovering CNRDB Data from Damaged Databases, page 8-7
Using Session Assert Commands for Data Management, page 8-10
Virus Scanning While Running Network Registrar, page 8-10
Using the cnr_exim Data Import and Export Tool, page 8-11
Using the mcdadmin Tool, page 8-12
Using the cnrdb_recover Utility, page 8-14
Using the cnrdb_verify Utility, page 8-15
Using the cnrdb_checkpoint Utility, page 8-15
Using the keybuild Tool, page 8-15
Using the dbcheck Tool, page 8-16
Restoring DHCP Data from a Failover Server, page 8-16
Troubleshooting Databases, page 8-10

Syntax and Location

Be sure to understand that the notation ".../data/db" in the following sections refers to directories in the Network Registrar product data location path, depending on the operating system:

- **Windows**—".../data" means the data directory, which by default is **\Program Files\Network Registrar\Local\data** or **\Program Files\Network Registrar\Regional\data**.
- **Solaris and Linux**—".../data" means the data directory, which by default is **/var/nwreg2/local/data** or **/var/nwreg2/regional/data**.

Network Registrar database utility programs mentioned in the following sections are located in the ".../bin" directory, which you run as its full path name:

- **Windows**—".../bin/*program*" means the program file in the bin directory, which by default is **\Program Files\Network Registrar\Local\bin***program* or **\Program Files\Network Registrar\Regional\bin***program*.
- **Solaris and Linux**—".../bin/*program*" means the program file in the bin directory, which by default is **/opt/nwreg2/local/usrbin/***program* or **/opt/nwreg2/regional/usrbin/***program*.

Note Use only the approved utilities for each type of database. In Windows, if you want to run the utility from outside the installed path, you must set the CNR_HOME environment variable.

Backup Strategy

The backup strategy involves backing up the following databases using the **mcdshadow** utility:

- **MCD database**—...data/db, ...data/mcd
- **CNRDB databases**—...data/dhcp, ...data/dns, ...data/cnrsnmp, ...data/leasehist, ...data/subnetutil, and ...data/replica
- **CCM database**—...data/ccm/ndb

Note If you change the location of the data directory, you must edit the **cnr.conf** file, which is located in .../conf (see the "Modifying the cnr.conf File" section on page 7-22). Change the **cnr.datadir** variable to the full path to the data directory. For example, the following is the default value on Windows:

```
cnr.datadir=C:\\Program Files\\Network Registrar\\Local\\data
```

The most basic component of a backup strategy is the daily shadow backup. When problems occur with the operational database, you might need to try recovering based on the shadow backup of the previous day. Therefore, you must recognize and correct any problems that prevent a successful backup.

The most common problem is disk space exhaustion. To get a rough estimate of disk space requirements, take the size of the .../data directory and multiply by 10. System load, such as usage patterns, application mix, and the load on Network Registrar itself, may dictate that a much larger reserve of space be available.

You should regularly archive existing shadow backups (such as to tape, other disks, or other systems) to preserve them for possible future recovery purposes.

Caution There are different database technologies Network Registrar uses with distinct sets of utility programs, as described in the following sections. Using a utility on the wrong type of database can cause database corruption. Use only the utilities indicated. Also, never use the database utilities on the operational database, only on a copy.

See Also

Setting Shadow Backup Times
Performing Manual Backups
Using Third Party Backup Programs with mcdshadow

Setting Shadow Backup Times

You can set the time at which an automatic shadow backup should occur by editing the **cnr.conf** file (in .../conf). Change the **cnr.backup-time** variable to the hour and minute of the automatic shadow backup, in 24-hour *HH*:*MM* format, then restart the server agent. For example, the following is the preset value:

```
cnr.backup-time=23:45
```

Performing Manual Backups

You can also initiate a manual shadow backup with the **mcdshadow** utility, which requires root privileges. Enter the **mcdshadow** command at the prompt to perform the shadow backup.

Note To restore a DHCP data from a failover partner that is more up to date than using a shadow backup, see the "Restoring DHCP Data from a Failover Server" section on page 8-16.

Using Third Party Backup Programs with mcdshadow

You should avoid scheduling third party backup programs while **mcdshadow** is operating, with a margin of an hour in either direction. As described in the "Setting Shadow Backup Times" section on page 8-3, the default shadow backup time is daily at 23:45.

Caution Configure third party backup programs to skip the Network Registrar operational database directories and files, and to back up only their shadow copies. The operational files are listed in the "Backup Strategy" section on page 8-2.

Network Registrar also maintains lock files in the .../temp directory. They are recreated during reboot, but are vital while a system is running. Any maintenance process (such as virus scanning and archiving) should exclude this temporary directory, together with the operational database directories and files.

Database Recovery Strategy

Network Registrar uses two types of databases, which means that there are two sets of tools to use for recovering data. The first type of database is MCD and the second type is CNRDB. Because the two databases are closely related, restore them together, unless directed otherwise.

Table 8-1 describes the databases and their types.

Table 8-1 Network Registrar Databases for Recovery

Subdirectory	Cluster	Type	Description
db	local, regional	MCD	MCD Configuration database. Stores configuration data. Works in tandem with the MCD transaction database.
mcd	local, regional	CNRDB	MCD Configuration database.
ccm	local, regional	CNRDB	Central Configuration Management (CCM) database. Stores local centrally managed cluster data.
dns	local	CNRDB	DNS database. Stores DNS resource record state and zone configuration data for the DNS server.
dhcp	local	CNRDB	DHCP database. Stores lease state data for the DHCP server.
cnrsnmp	local	CNRDB	SNMP database. Stores data for the SNMP server.
dhcpeventstore	local		Queue that Network Registrar maintains to interact with external servers, such as for LDAP and DNS Update interactions. Recovery is not necessary.
tftp	local		Default data directory for the TFTP server. Recovery is not necessary.
replica	regional	CNRDB	Stores replica data for the local clusters.
leasehist	regional	CNRDB	DHCP lease history database.
subnetutil	regional	CNRDB	Subnet utilization database.

The general approach to recovering a Network Registrar installation is:

1. Stop the Network Registrar server agent.
2. Restore or repair the data.
3. Restart the server agent.
4. Monitor the server for errors.

After you are certain that you executed a successful database recovery, always manually execute the **mcdshadow** utility to make a backup of the current configuration and state.

Backing Up and Recovering MCD Data

The operational database is in .../data/db and the shadow (backup) copy created when running **mcdchadow** is created in the …/data/db.bak directory. The actual filenames are mcddb.d01, mcddb.d02, mcddb.d03, and mcddb.dbd.

Caution For the MCD database, use only the **dbcheck** and **keybuild** utilities. Do not use the cnrdb_archive, cnrdb_recover, or cnrdb_verify utility that applies to the CNRDB database. See the "Troubleshooting Databases" section on page 8-10 for details. The **dbcheck** utility encounters errors if you try to check the databases while the servers are running. You must stop the servers before you run the utility.

See Also

Checking MCD Database Integrity
Recovering MCD Data from Damaged Databases
Recovering MCD Data from Backups
MCD Recovery Errors, page 8-6
MCD Data Files, page 8-6

Checking MCD Database Integrity

You can use the **dbcheck** utility to check the integrity of the MCD database. Stop all Network Registrar servers, then go to the .../data/db directory for the MCD database. As a safety check, enter the command **../../bin/dbcheck -a mcddb** (this requires system administrator or root privileges).

Recovering MCD Data from Damaged Databases

Depending on the event that caused the database corruption, you can restore the database to a healthy state using the current data by rebuilding its key file:

Step 1 Stop the Network Registrar Server Agent.

Step 2 Change to the .../data/db directory.

Step 3 Rebuild the key files by entering the command **../../bin/keybuild mcddb**.

Step 4 As a safety check, enter the command **dbcheck mcddb**. This requires system administrator or root privileges.

Remember to stop the server agent first. If there are any indications that the database recovery was unsuccessful, restore the database from a backup, as described in the "Recovering MCD Data from Backups" section on page 8-5. Also see the "MCD Recovery Errors" section on page 8-6.

Recovering MCD Data from Backups

Use the shadow backup to recover MCD data if rebuilding the key file was unsuccessful. This can happen if a system failure corrupts the regular working database, or the disk on which it resides is corrupted.

Step 1 Stop the Network Registrar Server Agent. Be certain that enough disk space is available for a copy of the database files, plus a 15% safety margin.

Step 2 Move the contents of the .../data//db directory to a different location, then delete any files in the .../data/db/log directory. You can move the files back after restoring the database, or you can back them up for future use.

Step 3 Ensure that the following four files are in the .../data/db.bak directory:

- mcddb.d01
- mcddb.d02
- mcddb.d03
- mcddb.dbd

Step 4 Copy these files to .../data/db. Do not use the **move** command, because you may need these files again.

Step 5 To create the key files, enter the command **../../bin/keybuild mcddb**.

Step 6 As a safety check, enter the command **dbcheck mcddb**. This requires system administrator or root privileges.

Step 7 Restore the accompanying CCM database files (see the "Recovering CNRDB Data from Backups" section on page 8-9).

Caution The CCM CNRDB database maintains centrally managed configuration data that is synchronized with the server configuration data. After restoring from backup, be sure also to restore the CCM CNRDB database from the same backup to maintain a consistent use of your configuration data.

MCD Recovery Errors

You should not have errors. If you do, ensure that:

- The Network Registrar Server Agent is stopped.
- Your current working directory is .../data/db directory.
- The mcddb.dbd and mcdschema.txt files are present in the .../data/db directory. If the files are lost or corrupted, backup files are stored in the .../data/db.bak directory.

MCD Data Files

You need the files listed in Table 8-2 for a fully functioning MCD database. As described earlier, only a subset of these files are present in a shadow backup. You need at a minimum the mcddb.dbd, mcddb.d0*x*, and mcdschema.txt files to rebuild the database from a backup.

Table 8-2 MCD Data Files in .../data/db Directory

Data File	Description
mcddb.dbd	Template file that describes the low level data schema for the MCD runtime library. Network Registrar cannot run without this file.
mcddb.k01-k03	Key files that contain the redundant data—Network Registrar does not back up these files because they can be completely rebuilt with the **keybuild** command.
mcddb.d01-d03	MCD data repository files.
mcdConfig.txt	Text file from which Network Registrar configures the initial install time database.
mcdschema.txt	Text file containing a version number of the schema in the mcddb.dbd file. Network Registrar does not try to open the database unless the number in this file matches a version constant in the libraries. Network Registrar cannot run without this file.
vista.taf, tcf, tjf	Working files MCD runtime uses to ensure transactional integrity. When database files are restored from a backup, the server uses or discards these files as needed.

Backing Up CNRDB Data

In the case of the CNRDB databases, the **mcdshadow** utility copies the database and all log files to a secondary directory in the directory tree of the installed Network Registrar product. For:

- **DHCP**—The operational database is in the .../data/dhcp/ndb directory, with the log files in the …/data/dhcp/ndb/logs directory. The shadow copies are in the …/data/dhcp.bak/ndb directory.
- **DNS**—The operational database is in the .../data/dns/ndb directory. The important operational components are the High-Availability (HA) DNS and changeset database and zone checkpoint files. The changeset database is in the .../data/dns/ndb/dns.ndb file, with log files in the …/data/dns/ndb/logs directory. The zone checkpoint files are in the .../data/dns/zchk directory. The shadow copies are in the …/data/dns.bak directory.
- **SNMP**—The operational database and log files are in the .../data/cnrsnmp/ndb directory. The shadow copies are in the …/data/cnrsnmp.bak/ndb directory.
- **CCM**—The operational database and log files are in the .../data/ccm/ndb directory. The shadow copies are in the …/data/ccm.bak directory.
- **MCD**—The operational database and log files are in the .../data/mcd/ndb directory. The shadow copies are in the …/data/mcd.bak directory.
- **Lease history**—The operational database and log files are in the .../data/leasehist directory. The shadow copies are in the …/data/leasehist.bak directory.
- **Subnet utilization**—The operational database and log files are in the .../data/subnetutil directory. The shadow copies are in the …/data/subnetutil.bak directory.
- **Replica**—The operational database and log files are in the .../data/replica directory.

The actual file naming convention is:

- **Database**—dhcp.ndb and dns.ndb.
- **Log files**—log.0000000001 through log.9999999999. The number of files varies with the rate of change to the server. There are typically only a small number. The specific filename extensions at a site vary over time as the database is used. These log files are not humanly readable.

Recovering CNRDB Data from Damaged Databases

This section describes a procedure that recovers any or all CNRDB type databases. Depending on the event that caused the database corruption, you can restore the database to a healthy state by using the current data. This is the best option. Always attempt recovery on a copy of the database file and associated log files, never on the operational files. This is a simple file copy operation, distinct from a shadow backup. Also, never attempt a recovery while Network Registrar is running.

In most cases, you can use the log files that accompany the databases (such as the DHCP log files in .../data/dhcp/ndb/logs) to repair a failed server database. You can do so because the log files journal all database activity. You should never move, rename, or delete these log files, even after successfully completing a recovery. In fact, the recovery process uses copies rather than the originals of these files.



Caution It is possible to damage the CNRDB database files without the damage being immediately obvious. Such damage could occur because of (a) inappropriately deleting log files; (b) mixing pre- and post-recovery database and log files; (c) attempting recovery of database files currently in use by an application; or (d)

using tools intended for other databases, such as MCD database tools. For the CNRDB database, be sure to use only the **cnrdb_archive**, **cnrdb_recover**, and **cnrdb_verify** utilities. Do not use the **keybuild** and **dbcheck** utilities that apply to the MCD database.

Use the **cnrdb_recover** utility (see the "Using the cnrdb_recover Utility" section on page 8-14), included in the Network Registrar product distribution, for database recovery. Use this tool with care. You should never use it directly on an operational database, or on files another application is concurrently accessing. On a successful database recovery, do not intermingle the recovered files (database file and log files) with files from another source, such as the operational database or shadow backups. Recovered database files acquire state information that make them incompatible with older database files.

Step 1 Stop the Network Registrar server agent. This stops all the protocol servers. Ensure that enough disk space is available for a copy of the database files, plus a 15% safety margin.

On Solaris, you can use the **df –k** utility to check your disk space, then **stop** to stop the server agent:

```
> df -k
> /etc/init.d/nwreglocal stop
```

Step 2 Create a backup directory, named backup, outside the Network Registrar installation tree. On Windows, this could be C:\temp\backup. On Solaris and Linux, this could be /tmp/backup. (As a precaution, a copy of the directory tree of the current database that you are going to repair will get copied here.)

For example, use **mkdir** on Solaris to create a backup directory:

```
> mkdir /tmp/backup
```

Step 3 Copy the database subdirectories you want to restore under .../data to the backup directory. For example, to recover the DHCP database, recursively copy the .../data/dhcp directory and its subdirectories to /tmp/backup:

```
> cp -rp /var/nwreg2/local/data/dhcp /tmp/backup
```

When the copy is completed, double check that the database file and all log files were copied correctly. Do not allow these files to be modified in any way. Do not run any utilities or servers on these files.

Step 4 Use the log journals to repair the database:

a. Copy the log files from their subdirectory into the same directory as the database files. For example:

```
> cd /var/nwreg2/local/data/dhcp/ndb
> cp ./logs/*
```

b. From the database file directory, run the **cnrdb_recover** program, using the **–c** and **–v** options. It is helpful to use **–v** in that it displays output in the absence of errors (see the "Using the cnrdb_recover Utility" section on page 8-14). For example:

```
> cd /var/nwreg2/local/data/dhcp/ndb
> /opt/nwreg2/local/bin/cnrdb_recover -v -c
db_recover: Finding last valid log LSN: file: 1 offset 95181
db_recover: Recovery starting from [1][28]
db_recover: Recovery complete at Mon June 19 18:44:15 2006
db_recover: Maximum transaction ID 800000009 Recovery checkpoint [1][95229]
db_recover: Recovery complete at Mon Jun 19 18:44:15 2006
db_recover: Maximum transaction ID 80000000 Recovery checkpoint [1][95529]
```

c. Run the **cnrdb_verify** utility for each of the servers. There is no output if the verification is successful (see the "Using the cnrdb_verify Utility" section on page 8-15). For example:

```
> cd /var/nwret2/local/data/dhcp/ndb
> /opt/nwreg2/local/bin/cnrdb_verify dhcp.ndb
```

d. Delete the log files. For example:

```
> cd /var/nwret2/local/data/dhcp/ndb
> rm log.*[0-9]
```

e. Optionally, for additional confidence, run the **cnrdb_archive** utility:

- **cnrdb_archive –l**—Lists all log files
- **cnrdb_archive –s**—Lists the database file

f. If there are any indications that an error occurred, proceed to restore the database from a backup, as described in the "Recovering CNRDB Data from Backups" section.

Step 5 For DHCP only, delete the files in the .../data/dhcpeventstore directory.

Step 6 Restart Network Registrar.

Recovering CNRDB Data from Backups

If there are any indications, such as server log messages or missing data, that database recovery was unsuccessful, you may need to base a recovery attempt on the current shadow backup (in the Network Registrar installation tree). Here is how:

1. Stop the Network Registrar server agent.
2. Move the operational database files to a separate temporary location.
3. Copy each .../data/*name*.bak directory to .../data/*name*; for example, copy .../data/mcd.bak to .../data/mcd.
4. Rebuild the key files by entering the command **../../bin/keybuild mcddb**.

Note If you set the **cnr.dbrecover** variable to false in the **cnr.conf** file to disable recovery during the **mcdshadow** nightly backup, you must also do a recovery as part of these steps.

5. Once the files are renamed, restart the server agent.

The CNRDB database maintains centrally managed configuration data that is synchronized with the server configuration databases. If you restore the CNRDB files from backup, also restore the MCD database from the same backup.

Note If the recovery fails, perhaps because the current shadow backup is simply a copy of corrupted files, use the most recent previous shadow backup. (This illustrates the need to regularly archive shadow backups.) You cannot add operational log files to older shadow backup files. All data added to the database since the shadow backup was made will be lost.

After a successful database recovery, using the **mcdshadow** utility, initiate an immediate shadow backup and archive the files (see the "Performing Manual Backups" section on page 8-3).

Using Session Assert Commands for Data Management

You can use the CLI **session assert** commands to simplify interactions with external data management processes. It also helps in writing multicommand batch scripts that stop processing if an asserted precondition fails. You generally use these commands with the default session format of *script*. If the assertion passes, you get a `100 Ok` message. If it fails, you get a `107 Assertion Failed xxx.dbsn (minor-serial-number) = value` message and the CLI exits.

The **session assert locked** command exits the CLI if it cannot lock the session. This sample command file performs batch operations requiring a lock. Note that the session default format is normally set to user format; you set it to script format here:

```
session set default-format=script
session assert locked
commands-that-require-a-lock
```

The **session assert dhcp.dbsn** command exits the CLI session if the minor serial number of the DHCP server does not match (**==**) or does not exceed (**!=**) the value given. The minor serial number is incremented with each configuration change. Get its value using **dhcp get dbsn**. This sample script modifies a DHCP server based on configuration version 1234:

```
session set default-format=script
dhcp get dbsn
session assert dhcp.dbsn == 1234
scope scope1 create 192.168.1.0 255.255.255.0
scope scope1 addRange 192.168.1.10 192.168.1.200
```

This sample script lists DHCP configuration changes made since version 1234:

```
session set default-format=script
session assert dhcp.dbsn != 1234
scope list
policy list
client-class list
```

Virus Scanning While Running Network Registrar

If you have virus scanning enabled on your system, it is best to configure it to exclude certain Network Registrar directories from being scanned. Including these directories might impede Network Registrar operation. Exclude from scanning the .../data, .../logs, and .../temp directories and their subdirectories.

Troubleshooting Databases

The following sections describe troubleshooting the Network Registrar databases.

See Also

Using the cnr_exim Data Import and Export Tool, page 8-11
Using the mcdadmin Tool, page 8-12
Using the cnrdb_recover Utility, page 8-14
Using the cnrdb_verify Utility, page 8-15
Using the cnrdb_checkpoint Utility, page 8-15

Using the keybuild Tool, page 8-15
Using the dbcheck Tool, page 8-16
Restoring DHCP Data from a Failover Server, page 8-16

Using the cnr_exim Data Import and Export Tool

Because Network Registrar extends the data repositories to serve the web UI, the current Network Registrar data import and export tool, **mcdadmin**, is no longer adequate (see the "Using the mcdadmin Tool" section on page 8-12). The **cnr_exim** data import and export tool now serves to import data to and export data from Network Registrar servers. The **cnr_exim** tool overcomes the **mcdadmin** limitations of not being able to export unprotected resource record information.

Note, however, that **cnr_exim** simply overwrites existing data and does not try to resolve conflicts.

Before using the **cnr_exim** tool, exit from the CLI, then find the tool on:

- **Windows**—...\bin\cnr_exim.exe
- **Solaris and Linux**—.../usrbin/cnr_exim

You must reload the server for the imported data to become active.

Note that text exports are for reading purposes only. You cannot reimport them.

The text export prompts for the username and password (the cluster defaults to the local cluster). The syntax is:

```
> cnr_exim -e exportfile [-N username -P password -C cluster]
```

To export (importable) raw data, use the **–x** option:

```
> cnr_exim -e exportfile -x
```

To export DNS server and zone components as binary data in raw format, use the **–x** and **–c** options:

```
> cnr_exim -e exportfile -x -c "dnsserver,zone"
```

The data import syntax is (the import file must be in raw format):

```
> cnr_exim -i importfile [-N username -P password -C cluster]
```

You can also overwrite existing data with the **–o** option:

```
> cnr_exim -i importfile -o
```

Table 8-3 describes all the qualifying options for the **cnr_exim** tool.

Table 8-3 cnr_exim Options

Option	Description
–a *value*	Allows exporting and importing of protected or unprotected RRs. Valid *values* are: **protectedRR** **unprotectedRR** On export or import, all RRs are exported by default, so you must use a value to export or import just the protected or unprotected RRs.

Table 8-3 cnr_exim Options (continued)

Option	Description
–c "*components***"**	Imports or exports Network Registrar components, as a quoted, comma-delimited string. Use **–c help** to view the supported components. User are not exported by default; you must explicitly export them using this option, and they are always grouped with their defined groups and roles. Secrets are never exported. **Note** After you import administrator names, you must set new passwords for them. If you export groups and roles separately from usernames (which are not exported by default), their relationship to usernames is lost.
–C *cluster*	Imports from or exports to the specified cluster. Preset to **localhost**.
–e *exportfile*	Exports the configuration to the specified file.
–h	Displays help text for the supported options.
–i *importfile*	Imports the configuration to the specified file. The import file must be in raw format.
–N *username*	Imports or exports using the specified username.
–o	When used with the **–i** (import) option, overwrites existing data.
–p *port*	Port used to connect to the SCP server.
–P *password*	Imports or exports using the specified password.
–x	When used with the **–e** (export) option, exports binary data in (importable) raw format.

Using the mcdadmin Tool

The **mcdadmin** tool is a utility with which you can import and export the MCD database and diagnose Network Registrar conditions under Cisco guidance. This tool is relevant for versions of Network Registrar before Release 6.0 only. To run the tool requires root privileges.

Caution Use this tool only under the guidance of the Cisco Technical Assistance Center. Casual use can inflict catastrophic damage to the Network Registrar configuration database. This tool is only relevant for versions of Network Registrar before Release 6.0. For Network Registrar Release 6.0 data imports and exports, see the "Using the cnr_exim Data Import and Export Tool" section on page 8-11.

Before using the **mcdadmin** tool, exit from the CLI, then find the tool on:

- **Windows**—...\bin\mcdamin.exe
- **Solaris and Linux**—.../usrbin/mcdadmin

Run the tool from the command shell. This example displays the application version number:

```
> mcdadmin -v
```

This command exports the scopePC scope configuration data to the cnrconfig.txt file:

```
> mcdadmin -e cnrconfig.txt -p /servers/name/dhcp/1/scopes/scopePC/ -x -z m=3
```

This command overwrites the current configuration database with the installation defaults:

```
> mcdadmin -i cnrconfig.txt -z m=3
```

Table 8-4 describes the qualifying options. The absolute path to an object in the MCD tree, *dbpath*, always begins and ends with a forward slash (**/**).

Table 8-4 mcdadmin Options

Option	Description
–a *area*	Specifies the area of the database to use: **config** (the preset value), **state**, **altconfig**, or **altstate**.
–c	Creates the MCD database, and deletes your existing one. **Note** Use this option only under Cisco Technical Assistance Center guidance.
–d *dbname*	Specifies the database name or the **default** database.
–e *exportfile*	Exports the configuration to the specified file, or a dash (**–**) for standard output. `> mcdadmin -N admin -P changeme -e mcdconfig.txt` `> mcdadmin -N admin -P changeme -e -`
–G *gen*	When used with the **–e** (export) option, specifies the generation number for the resource record table export.
–H	Dumps the full table generational history.
–i *importfile*	Imports the configuration from the specified file, or a dash (**–**) for standard input. `> mcdadmin -N admin -P changeme -o -i mcdconfig.txt -z m=3` **Note** If used with **–o**, **–i** overwrites the current database. If not used with **–o**, creates new the entries introduced by the file.
–k	When used just by itself, kills the lock manager.
–l	When used with the **–i** (import) option, hold an exclusive lock while importing.
–o	Overwrites the contents of the current database with entries of the same name.
–O	Merges the data into the existing database, while ignoring unique keys and the resource record tables.
–p *dbpath*	When used with the **–e** (export) option, specifies the subset of the configuration database to export, as an absolute path. `> mcdadmin -N admin -P changeme -e scopepc.txt` `-p /servers/name/dhcp/1/scopes/scopePC/ -x -z m=3`
–r *dbpath*	Removes the specified path in the MCD database, not including subentries. `> mcdadmin -r /servers/name/dhcp/1/scopes/scopePC/`
–R *dbpath*	Recursively removes all entries below and including the specified path. **Note** Using this option can destroy large portions of the MCD database that may require partial or total reconstruction
dbpath	The use of *dbpath* in –p, –r, and –R is either an absolute path name starting with /, when it refers to that part of the MCD tree, or a *server zone name* tuple, starting with a colon (**:**). The fields after the colon are whitespace-separated and the *name*, or *zone* and *name*, can be left unspecified. For example, **–p ":myserv example.com. mypath"**.
–t *dir*	Specifies the directory from which to get the database templates.
–T	Recopies the database templates when creating the database.

Table 8-4 mcdadmin Options (continued)

Option	Description
–v or **–V**	Prints version information.
–x	When used with the **–e** (export) option, exports binary data in raw format. `> mcdadmin -N admin -P changeme -e mcdconfig.txt -x`
–z *class=n*	Sets the debugging class or classes to level *n*. `> mcdadmin -N admin -P changeme -c -o -i mcdconfig.txt -z m=3`

Using the cnrdb_recover Utility

The **cnrdb_recover** utility is useful in restoring the Network Registrar databases to a consistent state after a system failure. You would typically use the **–c** and **–v** options with this command (Table 8-5 describes all of the qualifying options). The utility is located in the installation bin directory.

Table 8-5 cnrdb_recover Options

Option	Description
–c	Performs a catastrophic recovery instead of a normal recovery. It not only examines all the log files present, but also recreates the .ndb (or .db) file in the current or specified directory if the file is missing, or updates it if is present.
–e	Retains the environment after running recovery, rarely used unless there is a db_config file in the home directory.
–h *dir*	Specifies a home directory for the database environment. By default, the current working directory is used.
–t	Recovers to the time specified rather than to the most current possible date. The time format is *[[CC]YY]MMDDhhmm[.ss]* (the brackets indicating optional entries, with the omitted year defaulting to the current year).
–v	Runs in verbose mode.
–V	Writes the library version number to the standard output, and exits.

In the case of a catastrophic failure, restore a snapshot of all database files, along with all log files written since the snapshot. If not catastrophic, all you need are the system files at the time of failure. If any log files are missing, **cnrdb_recover –c** identifies the missing ones and fails, in which case you need to restore them and perform the recovery again.

Use of the catastrophic recovery option is highly recommended. In this way, the recovery utility plays back all the available database log files in sequential order. If, for some reason, there are missing log files, the recovery utility will report errors. For example, the following gap in the log files listed:

```
log.0000000001
log.0000000053
```

will result in the following error that might warrant a TAC case opened:

```
db_recover: Finding last valid log LSN:file:1 offset 2411756
db_recover: log_get: log.0000000002: No such file or directory
db_recover: DBENV->open: No such for or directory
```

Using the cnrdb_verify Utility

The **cnrdb_verify** utility is useful for verifying the structure of the Network Registrar databases. The command requires a file parameter. Use this utility only if you are certain that there are no programs running that are modifying the file. Table 8-6 describes all its qualifying options. The utility is located in the installation bin directory. The syntax is described in the usage information when you run the command:

```
C:\Program Files\Network Registrar\Local\bin>cnrdb_verify
usage: db_verify [-NoqV] [-h dir] [-P password] file
```

Table 8-6 cnrdb_verify Options

Option	Description
–h *dir*	Specifies a home directory for the database environment. By default, the current working directory is used.
–N	Prevents acquiring shared region locks while running, intended for debugging errors only, and should not be used under any other circumstances.
–o	Ignores database sort or hash ordering and allows **cnrdb_verify** to be used on nondefault comparison or hashing configurations.
–P *password*	User password, if the file is protected.
–q	Suppresses printing any error descriptions other than exit success or failure.
–V	Writes the library version number to the standard output, and exits.

Using the cnrdb_checkpoint Utility

The **cnrdb_checkpoint** utility is useful in setting a checkpoint for the database files so as to keep them current. The utility is located in the installation bin directory. The syntax is described in the usage information when you run the command:

```
C:\Program Files\Network Registrar\Local\bin>cnrdb_checkpoint ?
usage: db_checkpoint [-1Vv] [-h home] [-k kbytes] [-L file] [-P password] [-p min]
```

Using the keybuild Tool

The **keybuild** tool is a utility you can use to rebuild the database key files, which contain redundant data with the MCD files:

- **Windows**—Click **Start > Settings > Control Panel > Administrative Tools > Services**, highlight Network Registrar Local Server Agent or Network Registrar Regional Server Agent, then click **Stop**. You need system administrator privileges. Go to the *install-location*\data\db folder and run the keybuild.exe file:

  ```
  keybuild mcddb
  ```

- **Solaris**—Stop the Server Agent. Go to the .../data/db folder. You need root privileges:

  ```
  /etc/init.d/nwreglocal stop
  /etc/init.d/nwregregion stop
  keybuild mcddb
  ```

Using the dbcheck Tool

The **dbcheck** tool is a utility you can use to check the MCD integrity. You must have system administrator or root privileges to run the **dbcheck** tool:

- **Windows**—Click **Start > Settings > Control Panel > Administrative Tools > Services**, highlight Network Registrar Local Server Agent or Network Registrar Regional Server Agent, click **Start**, then go to the ...\data\db folder and run the dbcheck.exe file:

  ```
  dbcheck mcddb
  ```

- **Solaris and Linux**—Stop the server agent, then go to the .../data/db folder and run the dbcheck file:

  ```
  /etc/init.d/nwreglocal stop
  /etc/init.d/nwregregion stop
  dbcheck -a mcddb
  ```

Caution Do not run the **dbcheck** utility while the servers are running.

Restoring DHCP Data from a Failover Server

You can restore DHCP data from a failover server that is more current than the result of a shadow backup. Be sure that the failover partner configurations are synchronized, then, on the failover partner:

On Windows

1. Set the default path; for example:

   ```
   SET PATH=%PATH%;.;C:\PROGRA~1\NETWOR~1\LOCAL\BIN
   ```

2. Stop the server agent:

   ```
   net stop "Network Registrar Local Server Agent"
   ```

3. Delete the eventstore, ndb, and logs directories:

   ```
   del C:\Program Files\Network Registrar\Local\data\dhcpeventstore\*.*
   del C:\Program Files\Network Registrar\Local\data\dhcp\ndb\dhcp.ndb
   del C:\Program Files\Network Registrar\Local\data\dhcp\ndb\logs\*.*
   ```

4. Remove the MCD DHCP failover state:

   ```
   mcdadmin -N username -P password -a state -R /servers/name/dhcp/1/state
   ```

5. Restart the server agent:

   ```
   net start "Network Registrar Local Server Agent"
   ```

On Solaris and Linux

1. Stop the server agent:

   ```
   /etc/init.d/[nwreglocal|aicservagt] stop
   ```

2. Determine the processes running:

   ```
   ps -leaf | grep nwr
   ```

3. Kill the remaining processes:

   kill -9 *pid*

4. Delete the eventstore, ndb, and logs directories:

   ```
   rm /var/nwreg2/data/dhcpeventstore/*.*
   rm -r /var/nwreg2/data/dhcp/ndb/*
   ```

5. Remove the MCD DHCP failover state:

   mcdadmin -N *username* **-P** *password* **-a state -R /servers/name/dhcp/1/state**

6. Restart the server agent:

   /etc/init.d/[nwreglocal|aicservagt] start



PART 3

Address Management



CHAPTER 9

Managing Address Space

Address blocks provide an organizational structure for addresses used across the network. Address blocks can consist of static addresses or dynamic addresses allocated to DHCP servers for lease assignment. An address block can have any number of child address blocks and can culminate in one or more child subnets. The address block administrator is responsible for these objects. This administrator can create parent and child address blocks or subnets, which are always the leaf nodes of the address space. Static subnets can be further subdivided into one or more IP address ranges. However, dynamically added subnets create their own subnets that the administrator cannot modify or delete.

Note For IPv6 address management, see also the "Viewing IPv6 Address Space" section on page 26-9.

Address Block Administrator Role

The address block administrator role manages address space at a higher level than that of specific subnet or static address allocations. This is actually a middle manager role, because there is likely to be a higher authority handing out address blocks to the system.

See Also

Required Permissions
Role Functions, page 9-2

Required Permissions

To exercise the functions available to the address administrator, you must have at the:

- **Regional cluster**—The regional-addr-admin role assigned. This role should probably be unencumbered by further subnet-utilization, lease-history, ric-management, and dhcp-management subrole restrictions.
- **Local cluster**—The addrblock-admin role assigned.

Role Functions

These functions are available to the address block administrator at the:

- **Regional cluster**:
  - Address aggregation. For example, if the 10.0.0.0/16 address block exists at the regional cluster and a local cluster administrator creates the 10.1.1.0/24 address block, the local address block (through replication) is rolled up under its parent at the regional cluster. This allows a unified view of the address space at the regional cluster without affecting the local cluster configuration.
  - Address delegation. Administrators can delegate address space to the local cluster, thereby giving up authority of the delegated object.
  - Subnet utilization reports. The regional cluster supports subnet utilization reporting across regions, protocol servers, and sets of network hardware. The central configuration administrator can poll the local clusters for subnet utilization by virtual private network (VPN), if defined, time range, and criteria that contain the following choices: owner, region, address type, address block, subnet, or all. For details on querying subnet utilization, see the "Generating Subnet Utilization History Reports" section on page 9-14.
  - Lease history reports. This provides a single vantage point on the lease history of multiple DHCP servers. The administrator can query the history data at the local cluster to constrain the scope of the history report. Lease histories can be queried by VPN (if defined), time range and criteria that contain the following choices: IP address, MAC address, IP address range, or all. This is an important feature to meet government and other agency mandates concerning address traceability. For details on querying lease history, see the "Querying Leases" section on page 22-29.
  - Polling configurations. The administrator can control the intervals and periods of local cluster polling for replication, IP histories, and subnet utilization. You can also set the lease history and subnet utilization trimming ages and compacting intervals at the CCM server level. (See Chapter 6, "Managing the Central Configuration.")
  - Check the DHCP and address data consistency.
- **Local cluster**:
  - Manage address blocks, subnets, and address types.
  - Check the DHCP and address data consistency.

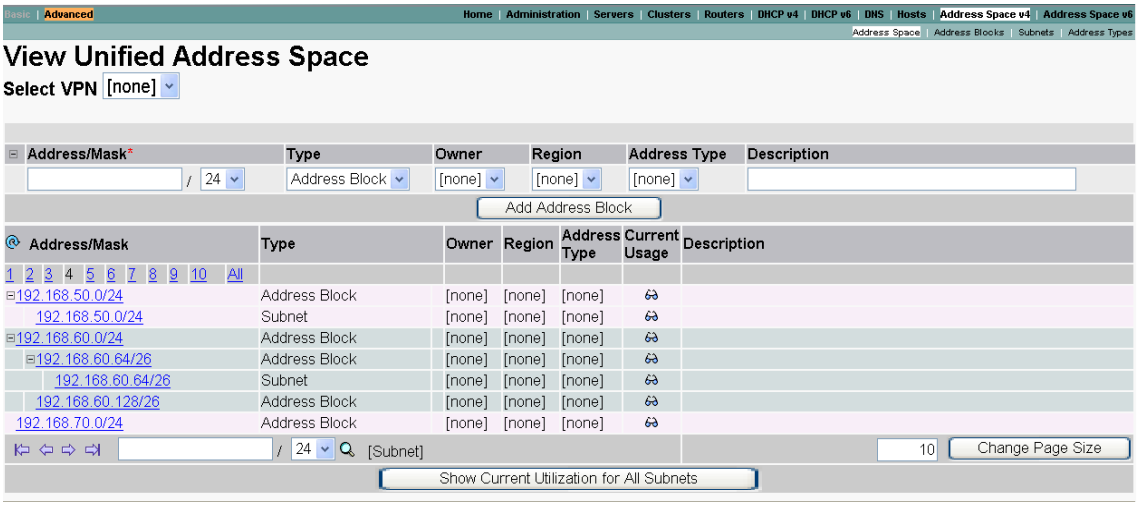
Viewing Address Space

The address space is a hierarchical tree of address blocks and subnets in IPv4 and prefixes in IPv6, sorted in IP address order. You can choose the level of depth at which to display the tree. You can also expand and contract nodes, which recursively expands or contracts all child nodes. If you pick a new level, this overrides the previous expansion or contraction.

Local Advanced and Regional Web UI

Click **Address Space**, then **Address Space** to open the View Unified Address Space page (see Figure 9-1 for the IPv4 version of the page). Note that you can choose a VPN (if configured).

Figure 9-1 View IPv4 Address Space Page (Local Advanced)



211546

Pulling Replica Address Space from Local Clusters

You may choose to pull address space from the replica data of the local clusters instead of explicitly creating it.

Note Pulling replica address space from a local cluster where IPv4 subnets were removed does not clear the server name on the subnet. Although the subnet is no longer used, it is still considered allocated to the server. Hence, the delete operation does not appear for the subnet, so that you cannot delete the subnet from the regional cluster. To push or reallocate the subnet to a different cluster, or remove it from the regional cluster, you must first reclaim the subnet (see the "Reclaiming Subnets" section on page 9-10). This clears the reference to the local server.

Regional Web UI

Step 1 On the View Unified Address Space (or View Unified v6 Address Space) page, click **Pull Data**. (To omit reservations in the pull, click a check mark in the Omit Reservations? check box.)

Step 2 Choose the data synchronization mode on the Select Pull Replica Address Space (or Select Pull Replica IPv6 Address Space) page.

Step 3 Click **Report** at the bottom of the page.

Step 4 Click **OK** on the Report Pull Replica Address Space (or Report Pull Replica IPv6 Address Space) page.

Address Blocks and Subnets

An address block is an aggregate of IP addresses based on a power-of-two address space that can be delegated to an authority. For example, the 192.168.0.0/16 address block (part of the RFC 1918 private address space) includes 2^{16} (or 65536) addresses. Address blocks can be further divided into child address blocks and subnets. For example, you might want to delegate the 192.168.0.0/16 address block further into four child address blocks—192.168.0.0/18, 192.168.64.0/18, 192.168.128.0/18, and 192.168.192.0/18.

Note The DHCP server also uses address blocks to manage subnet allocation for on-demand address pools (see the "Configuring Virtual Private Networks and Subnet Allocation" section on page 23-14). Address blocks used for dynamic address pools must be created using the **dhcp-address-block** command in the CLI. The unified address view in the web UI also displays these dynamic address blocks, but does not provide an edit link to them, because they have been delegated in their entirety to the DHCP server. They should not be further subdivided for subnet allocation. The DHCP server automatically handles these address blocks as it receives subnet requests. These address pools are indicated by a **D** (for "Delegated").

A subnet is the leaf node of the address space and cannot be further subdivided. If you create the 192.168.50.0/24 subnet, you can subsequently create an address block by that same name, and the subnet will become a child of the address block. However, you cannot further subdivide or delegate the 192.168.50.0/24 subnet.

Subnets can have one or more defined address ranges. Address blocks cannot have address ranges. When you create an address range for a subnet by using the web UI, it becomes a static range, meaning that it cannot be allocated dynamically using DHCP. However, the web UI shows any dynamic ranges defined by DHCP scopes for the subnet. Displaying the ranges as such indicates where overlaps may occur between assigning static addresses for the address space and dynamic addresses for scopes.

The address space view shows the hierarchy of address block and subnets and their parent-child relationships. The hierarchy does not go down to the level of address ranges for each subnet. These are displayed when you access the subnet.

See Also

Viewing Address Blocks, Subnets, and Address Types, page 9-4
Knowing When to Add Address Blocks, page 9-5
Adding Address Blocks, page 9-5
Delegating Address Blocks, page 9-7
Pushing Subnets to Local DHCP Servers and Routers, page 9-8
Creating Reverse Zones from Subnets, page 9-9
Reclaiming Subnets, page 9-10
Adding Children to Address Blocks, page 9-10
Adding Address Ranges to Subnets, page 9-11
Viewing Address Utilization for Address Blocks, Subnets, and Scopes, page 9-12

Viewing Address Blocks, Subnets, and Address Types

You can view the address blocks and subnets created for a network.

Local Advanced and Regional Web UI

Click **Address Space**, then **Address Space**. This opens the View Unified Address Space page (see Figure 9-1 on page 9-3).

To choose a level of depth for the address space, click one of the numbers across the top, or click **All** to get all levels. The address space appears below the row of numbers. The Address Type column identifies the type of object displayed, an address block or a subnet. The Owner column identifies the owner of the address space, and the Region column identifies the assigned region for the address space.

Address spaces that were assigned dynamically are indicated by a **D** (for "Delegated") in the Address Type column. You cannot delete this delegated address space.

To refresh the view, click the Refresh icon ( ).

You can add, modify, and delete address types. Click **Address Space**, then **Address Types** to open the List Address Types page. Click **Add Address Types** to open the Add Address Type page, and modify settings on the Edit Address Type page. You can also pull replica address types and push address types to the local clusters on the List Address Types page.

Knowing When to Add Address Blocks

This use case describes the set of user actions associated with adding a new address block to the network in a shared management network. These preconditions are assumed:

1. From summary IP address utilization reports (see the "Enabling Subnet Utilization Collection" section on page 6-11), an address block administrator notes that the top level address block of the company is nearing the 90% utilization mark.
2. The address block administrator submits a request for more address space from ARIN (or some other numbering authority) and the request is granted.

Once the address space is made available, the regional address administrator:

1. Adds the new blocks to the central address block map, and based on a review of the utilization reports, creates and delegates address blocks to be used by the local clusters. The action of delegating the address blocks causes them to be pushed to the local clusters.
2. Allocates the new address space to network elements as needed, using router and failover synchronization features to simplify the configuration tasks:
   - Allocates subnets to a failover pair (gets a scope template for the subnet, either from the subnet or the failover pair).
   - Allocates subnets to a router interface configuration (RIC) server interface and failover pair.
   - Finds a free subnet (finds the address block of the right type).
   - Allocates the free subnet to an address destination (DHCP server or other destination).

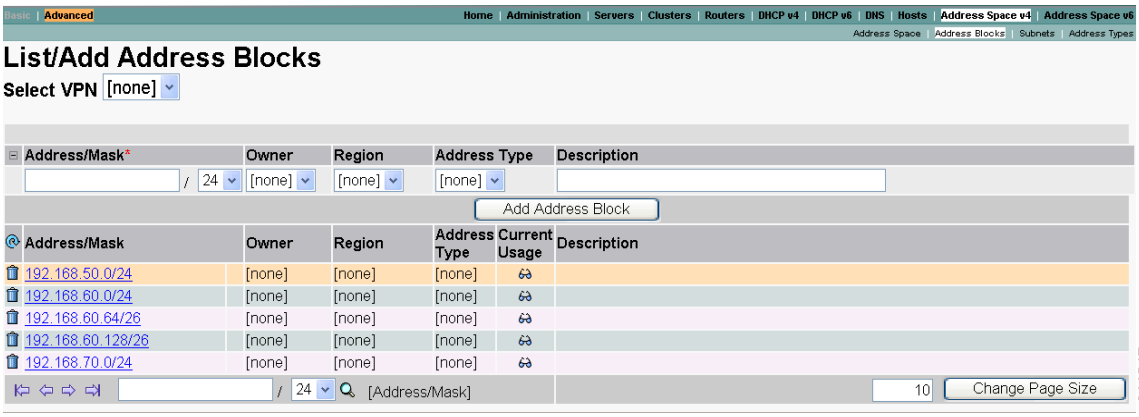
Adding Address Blocks

Once you configure your network, you can add DHCPv4 address blocks.

Local Advanced and Regional Web UI

To view CCM address blocks, click the **Address Blocks** submenu to open the List/Add Address Blocks page (see Figure 9-2 on page 9-6 for the local cluster version).

Figure 9-2 List/Add Address Blocks Page (Local Advanced)



To add an address block, enter its network address in the Address/Mask field, then choose the address mask from the drop-down list. For example, enter 192.168.50.0 in the Address Mask field, then choose 24 in the drop-down list to create the 192.168.50.0/24 address block, which is all the addresses in the range 192.168.50.0 through 192.168.50.256.

For a review of the number of available addresses for each subnet mask, see Table 9-1. These available hosts exclude the two network and broadcast addresses in each range.

Table 9-1 Subnet Masking

Network Mask	Octet Designation	Available Hosts in Each Address Range
/8	255.0.0.0	16777214
/9	255.128.0.0	8338606
/10	255.192.0.0	4194302
/11	255.224.0.0	2097150
/12	255.240.0.0	1048574
/13	255.248.0.0	524286
/14	255.252.0.0	262142
/15	255.254.0.0	131070
/16	255.255.0.0	65534
/17	255.255.128.0	32766
/18	255.255.192.0	16382
/19	255.255.224.0	8190
/20	255.255.240.0	4084
/21	255.255.248.0	2046
/22	255.255.252.0	1022
/23	255.255.254.0	5010
/24	255.255.255.0	254
/25	255.255.255.128	126
/26	255.255.255.192	62

Table 9-1 Subnet Masking (continued)

Network Mask	Octet Designation	Available Hosts in Each Address Range
/27	255.255.255.224	30
/28	255.255.255.240	14
/29	255.255.255.248	6
/30	255.255.255.252	2

Delegating Address Blocks

Address block delegation is the coordinated actions of marking the delegated address block at the regional cluster as being delegated to a local cluster and creating the delegated address block in the local cluster. To delegate an address block to a local cluster, the address block cannot have child address blocks or subnets. The delegated address block created at the local server must have the same address size as the one at the regional cluster.

You can delegate only one address block to one local cluster at a time; you cannot delegate it to multiple local clusters. You can also delegate an address block to an owner.

To delegate an address block, you must:

1. Have the central configuration administrator create a local cluster to which to delegate the address block (see the "Configuring Server Clusters" section on page 6-2).
2. Have the central configuration administrator synchronize the regional cluster with the local cluster (see the "Synchronizing with Local Clusters" section on page 6-7). The local cluster will have address source references to the regional cluster through the synchronization process.
3. Delegate the address block to the cluster or an owner.

Regional Web UI

For example:

Step 1 Have the central configuration administrator create a local cluster, ServProv-One:

a. Log in to the regional cluster as the central configuration administrator.

b. Click **Clusters**, then **Cluster List**.

c. Click **Add Cluster** on the List Server Clusters page to open the Add Server Cluster page.

d. Enter the cluster name **ServProv-One** and the connection data, then click **Add Cluster**.

e. On the List Server Clusters page, click the Resynchronize icon () next to ServProv-One.

Step 2 As regional address administrator, create an address block:

a. Log in to the regional cluster as the regional address administrator.

b. Click **Address Space**, then **Address Blocks** to open the List/Add Address Blocks page.

c. Enter **192.168.50.0** in the Address/Mask field, then choose **24** in the mask drop-down list.

d. Click **Add Address Block**.

Step 3 Delegate the address block to a cluster or owner:

a. Click the name of the address block to open the Edit Address Block page.

b. In the Address Block Delegation section of the page, choose either a local cluster or an owner to which to delegate the address block.

c. Click **Delegate Block**. The Edit Address Block page now indicates that the address block is delegated.

d. If there are further modifications to the address block, click **Modify Address Block**, otherwise click **Cancel**.

e. The List/Add Address Blocks page now identifies the address block as being delegated (**D**). To undelegate it, edit the address block again, then click **Reclaim Address Block**.

Pushing Subnets to Local DHCP Servers and Routers

You can push subnets to local DHCP servers and routers.

Local Advanced and Regional Web UI

Step 1 Have the central configuration administrator create a local cluster and resynchronize it with the local cluster.

Step 2 Create a subnet at the regional cluster:

a. Click **Address Space**, then **Subnets**. This opens the List/Add Subnets page (see Figure 15-4 on page 15-13).

b. Enter at least the network address and choose the mask of the subnet, then click **Add Subnet**.

Step 3 Have the central configuration administrator create a scope template so that it can create a scope to contain a subnet:

a. Log in to the regional cluster as the central configuration administrator.

b. Click **DHCP**, then **Scope Templates** to open the List DHCP Scope Templates page.

c. Click **Add Scope Template** to open the Add DHCP Scope Template page.

d. Among other entries on this page, enter the **create-range** expression in the Range Expression field to create a scope with that subnet. (If you choose a policy for the scope template, be sure that the policy exists at the local cluster, or you must push the policy to the local cluster. See the "Pushing Policies to Local Clusters" section on page 6-14.) Click **Add Scope Template**.

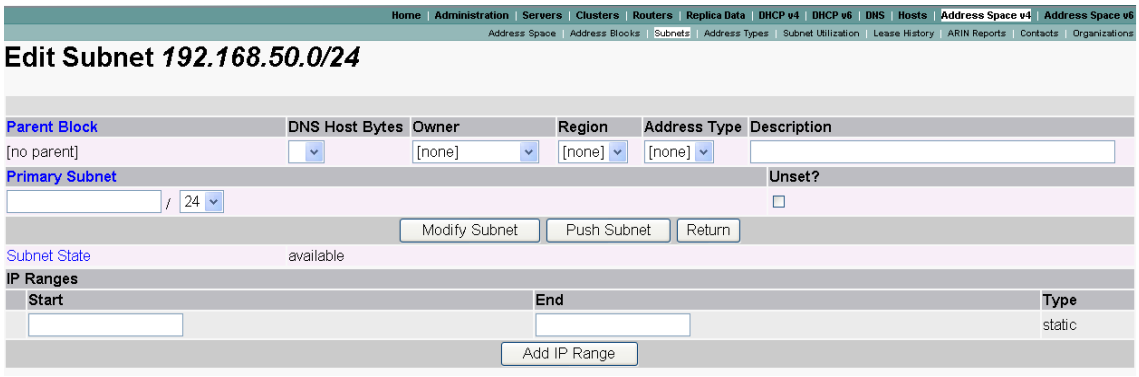
Step 4 As regional address administrator, add the subnet to the local cluster DHCP server:

a. Log in to the regional cluster as the regional address administrator.

b. Click **Address Space**, then **Subnets** to open the List/Add Subnets page.

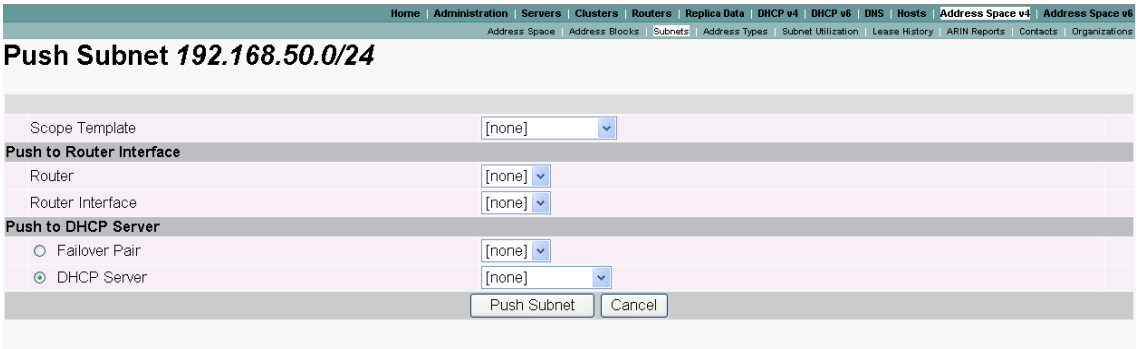
c. Click the name of the subnet to open the Edit Subnet page (see Figure 9-3 on page 9-9).

Figure 9-3 Edit Subnet Page (Regional)



d. Click **Push Subnet**. This opens the Push Subnet page (see Figure 9-4).

Figure 9-4 Push Subnet Page (Regional)



e. Choose the scope template from the drop-down list.

f. Choose the router and the router interface from the drop-down lists.

g. Choose the DHCP Server radio button, then choose the cluster from the drop-down list.

h. Click **Push Subnet**.

Creating Reverse Zones from Subnets

You can create reverse zones from subnets directly on the List/Add Subnets page instead of having to do so manually (see the "Adding Reverse Zones from Subnets" section on page 15-13). Click the Create icon ( ) in the Reverse Zone column of the List/Add Subnets page to open the Create Reverse Zone(s) for Subnet page. On that page, choose a configured zone template from the drop-down list, then click **Report** to return to the List/Add Subnets page.

See Also

Reclaiming Subnets
Adding Address Ranges to Subnets, page 9-11
Viewing Address Utilization for Address Blocks, Subnets, and Scopes, page 9-12
Pushing Subnets to Local DHCP Servers and Routers, page 9-8

Reclaiming Subnets

Once you delegate a subnet to the DHCP or RIC server, you can reclaim it if necessary.

Local Advanced and Regional Web UI

Step 1 Click **Address Space**, then **Subnets** to open the List/Add Subnets page (see Figure 15-4 on page 15-13).

Step 2 Click the name of the subnet to open the Edit Subnet page (see Figure 9-3 on page 9-9).

Step 3 Click **Reclaim Subnet** to open the Reclaim Subnet page.

Step 4 If you want to force deleting the subnet, click a check mark in the Force Delete box.

Step 5 Click **Reclaim Subnet**.

Note When you push or reclaim subnets for a managed or virtual router, this sets the primary and secondary relationships that are set for the router for all the related subnets and scopes as well. For details on routers, see the "Pushing and Reclaiming Subnets for Routers" section on page 11-4.

Adding Children to Address Blocks

You might want to subdivide undelegated address blocks into child address blocks or subnets.

Local Advanced and Regional Web UI

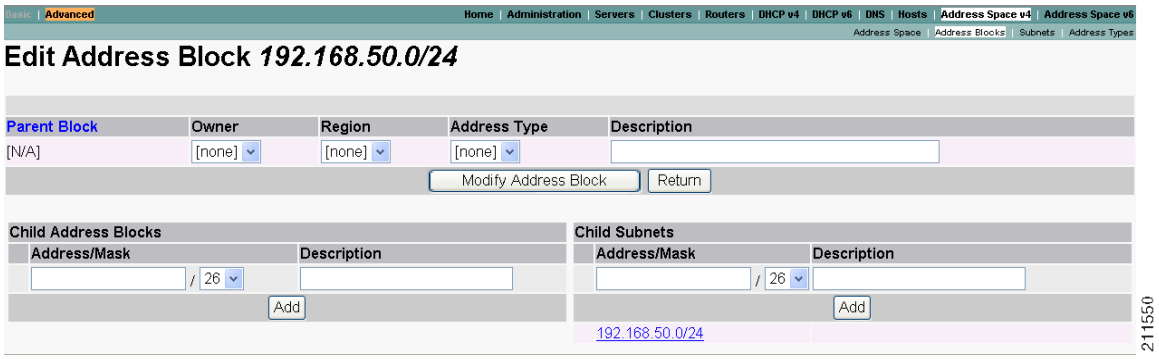
Step 1 Click **Address Space**, then **Address Blocks** to open the List/Add Address Blocks page (see Figure 9-2 on page 9-6).

Step 2 Click the name of an address block that is not marked as delegated (**D**). This opens the Edit Address Block page (see Figure 9-5 on page 9-11).

Step 3 To add a child address block, add an address that is part of the address block network address in the Address/Mask field of the Child Address Blocks section of the Edit Address Block page. Choose a higher mask value than the parent address block, then click **Add**.

An error message appears if you try to set the same network address for a child address block as for a child subnet.

Figure 9-5 Edit Address Block Page (Local Advanced)



Omitting a value when you click **Add** automatically adds the subdivisions of the parent address space with the appropriate mask value. For example, if the parent space is 192.168.50.0/24, you omit any child subnet value, and click **Add**, the web UI adds the children in this order:

192.168.50.0/26
192.168.50.64/26
192.168.50.128/26
192.168.50.192/26

Step 4 To add a child subnet, add an address in the Address/Mask field of the Child Subnets section of the page that is part of the address block network address, but choose a higher mask value than the parent address block. Then click **Add**.

An error message appears if you try to set the same network address for a child address block as for a child subnet.

If you omit a value when you click **Add**, this automatically adds the subdivisions of the parent address space with the appropriate mask value. For example, if the parent space is 192.168.50.0/24, you omit any child subnet value, and click **Add**, the web UI adds the children in this order:

192.168.50.0/26
192.168.50.64/26
192.168.50.128/26
192.168.50.192/26

Adding Address Ranges to Subnets

You can edit the subnet data and add any number of address ranges to a subnet. These ranges must be in the designated network of the subnet.

Local Advanced and Regional Web UI

Step 1 Click **Address Space**, then **Subnets** to open the List/Add Subnets page (see Figure 15-4 on page 15-13).

Step 2 Click the name of the subnet to which you want to add address ranges. This opens the Edit Subnet page (see Figure 9-3 on page 9-9).

Step 3 Enter the starting address of the range in the Start field in the IP Ranges area of the page, then add the ending address in the End field. If you add just the host numbers in these fields, the relative address in the range determined by the address mask is used.

Step 4 Click **Add IP Range**.

Viewing Address Utilization for Address Blocks, Subnets, and Scopes

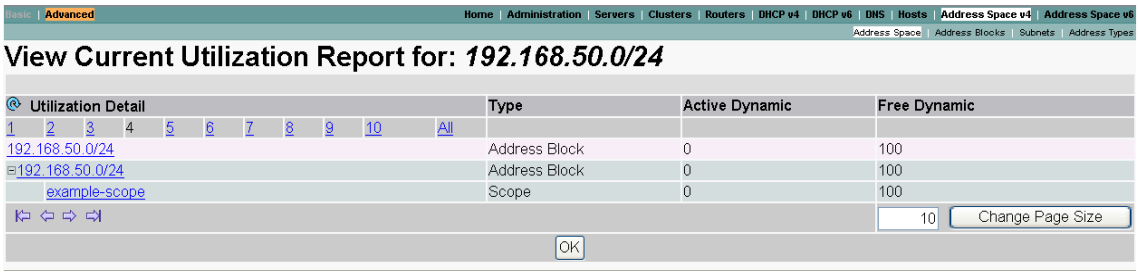
You can view the current address utilization for address blocks, subnets, and scopes.

Tip For address utilization for IPv6 prefixes, see the "Viewing Address Utilization for Prefixes" section on page 26-22.

Local Advanced and Regional Web UI

The function is available on the View Unified Address Space page (see Figure 9-1 on page 9-3), List/Add Address Blocks page (see Figure 9-2 on page 9-6), and List/Add Subnets page (see Figure 15-4 on page 15-13). When you click the View icon ( ) in the Current Usage column, or the **Show Current Utilization for All Prefixes** button, the View Current Utilization Report page appears (see Figure 9-6).

Figure 9-6 View Current Utilization Report Page (Local Advanced)



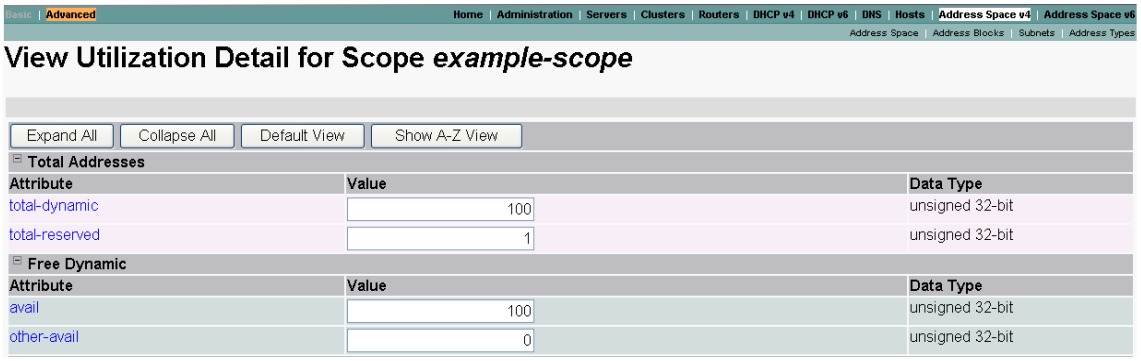
Note To ensure the proper subnet-to-server mapping on this page, you must update the regional address space view so that it is consistent with the relevant local cluster. Do this by pulling the replica address space, or reclaiming the subnet to push to the DHCP server (see the "Reclaiming Subnets" section on page 9-10). Also ensure that the particular DHCP server is running.

The other columns on the View Current Utilization Report page identify:

- **Type**—Whether the address space is an address block, subnet, or scope.
- **Active Dynamic**—Addresses that are part of a dynamic range managed by DHCP and that are currently leased, but not reserved.
- **Free Dynamic**—Addresses that are not currently leased.
- **Active Reserved**—Addresses that are part of a dynamic range and are reserved.
- **View Utilization History**—Appears at the regional cluster only. Clicking the Report icon ( ) opens the List Subnet Utilization Records page, where you can refine the subnet utilization history query.

The Utilization Detail column items are expandable on the View Current Utilization Report page so that you can view the scope data for an address block or subnet. If you click the address block, subnet, or scope name in this column, this opens the View Utilization Detail page (see Figure 9-7 for a partial view of a scope page).

Figure 9-7 View Utilization Detail Page (Local Advanced)



The View Utilization Detail page is a read-only page that shows detailed address utilization attributes for the address block, subnet, or scope. The address utilization attributes are described in Table 9-2.

Table 9-2 Address Utilization Attributes

Utilization Attribute	Description
Total Addresses	
total-dynamic	Total number of leases, excluding reserved ones.
total-reserved	Total number of reserved leases.
Free Dynamic	
avail	Number of dynamic leases that are currently available for issue to clients.
other-avail	Number of dynamic leases that the DHCP failover partner currently has available for issue to clients.
Active Dynamic	
offered	Number of dynamic leases that are currently offered to clients, but not yet acknowledged as being leased.
leased	Number of dynamic leases that are currently acknowledged as leased to clients.
expired	Number of dynamic leases that are past the lease expiration period, but will not be available for other clients (except after the policy grace-period expires).
pend-avail	Number of dynamic leases that are waiting acknowledgement from the failover partner that it did not reissue the lease.
Reserved	
reserved-active	Number of reserved leases that clients are actively using.
reserved-inactive	Number of reserved leases that clients are not actively using.

Table 9-2 Address Utilization Attributes (continued)

Utilization Attribute	Description
Unavailable	
unavail	Number of unreserved dynamic leases that a client declines or the server marks with an address conflict (usually indicating configurations that need correcting).
reserved-unavail	Number of reserved leases that a client declines or the server marks with an address conflict (usually indicating configurations that need correcting).
Deactivated	
leased-deactivated	Number of dynamic leases that clients are actively leasing (that are not offered, expired, or released), but that an administrator deactivated.
reserved-leased-deactivated	Number of reserved leases that clients are actively leasing (that are not offered, expired, or released), but that an administrator deactivated.

Generating Subnet Utilization History Reports

You can extract subnet utilization history data so that you can determine how many addresses in the subnet were allocated and what the free address space is. You can use additional administrative functions to trim and compact the subnet utilization database of records, to manage the size of the database.

See Also

Enabling Subnet Utilization History Collection at the Local Cluster
Querying Subnet Utilization History Data, page 9-15
Trimming and Compacting Subnet Utilization History Data, page 9-16
Viewing Subnet Utilization History Data, page 9-17

Enabling Subnet Utilization History Collection at the Local Cluster

You must explicitly enable subnet utilization collection for the local cluster DHCP server.

Local Basic or Advanced Web UI

Step 1 Click **DHCP**, then **DHCP Server**.

Step 2 On the Manage DHCP Server page, click the **Local DHCP** Server link.

Step 3 On the Edit DHCP Server page, look for the Subnet Utilization Settings attributes, which determine how frequently snapshots of the data occur and over which period of time the data should be maintained:

- ***collect-addr-util-duration***—Maximum period, in hours, the DHCP server maintains address utilization data. The preset value is 0. To disable DHCP server from collecting any address utilization data, unset this parameter or set it to 0.
- ***collect-addr-util-interval***—Frequency, in minutes or hours, that the DHCP server should maintain address utilization data snapshots, assuming that the *collect-addr-util-duration* attribute is not unset or set to 0. The preset value is 15 minutes.

Note that both of these parameters can impact DHCP server memory. Each snapshot of data collected for every interval is 68 bytes. For example, if there are 10 scopes, the collection duration is set to 24 hours, and the collection interval is set to one hour, memory used by the DHCP server to maintain address utilization data is 24 times 68 bytes for each scope, or 16 K.

Step 4 Click **Modify Server** at the bottom of the page.

Step 5 Reload the DHCP server.

Querying Subnet Utilization History Data

You collect subnet utilization by first having subnets and setting up the scopes, address ranges, and collection criteria at the local cluster. You then set up the local cluster containing the DHCP server as part of the regional cluster, and enable polling the subnet utilization data from the regional cluster.

Regional Web UI

Step 1 Click **Clusters**, then **Cluster List** to open the List Remote Clusters page.

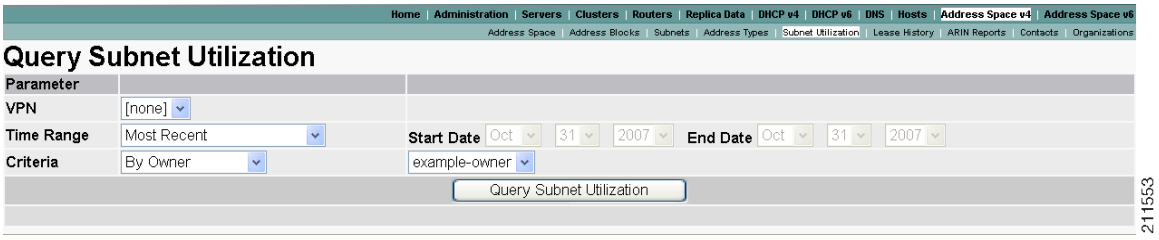
Step 2 Click the name of the local cluster to open the Edit Remote Cluster page.

Step 3 Look for the Subnet Utilization Settings attributes:

- ***poll-subnet-util-interval***—Polling interval; be sure that this is set to a reasonable time interval greater than 0.
- ***poll-subnet-util-retry***—Retry count in case of a polling failure; preset to one retry.
- ***poll-subnet-util-offset***—Fixed time when polling occurs. For example, setting the offset to 13h (1 P.M.) with the polling interval set to 2h means that polling occurs every two hours, but it must occur at 1 P.M. each day.

Step 4 You must also set the selection criteria for querying the subnet utilization data—Click **Address Space**, then **Subnet Utilization** to open the Query Subnet Utilization page (see Figure 9-8).

Figure 9-8 Query Subnet Utilization Page (Regional)



Step 5 You can query subnet utilization history based on the following criteria:

a. **Time range**—Choose from one of the following time ranges for the lease history data:
   - last 10 days
   - last 30 days
   - last 60 days
   - last 90 days
   - from/to (limited to 90 days)

     If you choose this value, also choose the Start Date and End Date month, day, and year from the drop-down lists. The result depends on the value of the *poll-subnet-util-interval* attribute.

b. **Criteria**—Choose the criteria on which you want to base the query:
   - **By Owner**—Choose the owner from the adjacent drop-down list.
   - **By Region**—Choose the region from the adjacent drop-down list.
   - **By Address Type**—Choose the address type from the adjacent drop-down list.
   - **By Address Block**—Choose the address block from the adjacent drop-down list.
   - **By Subnet**—Choose the subnet from the adjacent drop-down list.
   - **All**—Choose by all owners, regions, address types, address blocks, and subnets.

Step 6 Click **Query Subnet Utilization** to open the List Subnet Utilization Records page (see the "Viewing Subnet Utilization History Data" section on page 9-17).

Trimming and Compacting Subnet Utilization History Data

If you enable subnet utilization, its database is trimmed automatically based on the expiration time of each record. You can also compact the data so that you can view subsets of the records older than a certain age. The CCM server performs background trimming at the regional cluster, which trims off the subnet utilization data older than a certain age at regular intervals. The trimming interval is preset to 24 hours, and the age (how far back to go in time before trimming) to 24 weeks.

Regional Web UI

You must be a central configuration administrator assigned the database subrole to adjust the values of and perform subnet utilization database trimming and compacting.

Step 1 Click **Servers** to open the Manage Servers page.

Step 2 Click the **Local CCM Server** link to open the Edit CCM Server page.

Step 3 Under the Subnet Utilization Settings, set the following attributes:

- ***trim-subnet-util-interval***—How often to trim the old subnet utilization data automatically, the default being not to trim the data. You must set this to a value to trigger any background trimming. The bounded values are 0 to one year, and you can use units in seconds (s), minutes (m), hours (h), days (d), weeks (w), months (m), and years (y).
- ***trim-subnet-util-age***—How far back in time to trim the old subnet utilization data automatically, the preset value being 24 weeks. (However, the *trim-subnet-util-interval* value must be set to other than 0 for trimming to be in effect at all.) The bounded values are 24 hours to one year, and you can use units in seconds (s), minutes (m), hours (h), days (d), weeks (w), months (m), and years (y).

Step 4 You can also force immediate trimming and compacting. Find the Trim/Compact Inputs section:

a. **Trim/Compact age**—How far in time to go back to trim the data. There are no bounds to this value. However, if you set a very small value (such as 1m), it trims or compacts very recent data, which can be undesirable. In fact, if you set it to zero, you lose all of the collected data. Setting the value too high (such as 10y) may end up not trimming or compacting any data.

b. **Compact interval**— Time interval at which to compact the subnet utilization records older than the Trim/Compact age. This interval can be some multiple of the polling interval. For example, if the compact interval is set to twice the polling interval, it eliminates every other record.

Step 5 If you are trimming immediately, click **Trim All Subnet Utilization** among the controls at the bottom of the page. If you are compacting the data, click **Compact All Subnet Utilization**.

Viewing Subnet Utilization History Data

The DHCP server gathers subnet utilization data into three broad categories:

- Active Reserved
- Active Unreserved
- Free Reserved

Each of these categories has a current value for a given collection interval, and low and high values over the life of the DHCP server.

To illustrate the three subnet utilization categories, consider this DHCP scope configuration:

```
Scope 10.10.10.0/24
Range 10.10.10.1   10.10.10.10
Range 10.10.10.20 10.10.10.30
Reservation  10.10.10.1  MAC-1
Reservation  10.10.10.2  MAC-2
Reservation  10.10.10.41 MAC-3
Reservation  10.10.10.42 MAC-4
```

Of the 254 potential leases, only 31 are configured, and two reservations are outside the address range.

Immediately after configuring the scope, adding the ranges and reservations, and reloading the DHCP server, these counters appear for subnet utilization:

```
Active Reserved 0
Active Unreserved 0
Free Unreserved 20
```

As soon as clients MAC-1 and MAC-2 get their reserved leases, subnet utilization then shows as:

```
Active Reserved 2
Active Unreserved 0
Free Unreserved 20
```

When the client MAC-5 gets lease 10.10.10.3, subnet utilization then shows as:

```
Active Reserved 2
Active Unreserved 1
Free Unreserved 19
```

Regional Web UI

Click **Address Space**, then **Subnet Utilization** to open the List Subnet Utilization Records page.

Tip At the top left corner of the List Subnet Utilization Records page is either the Log icon () for the Netscape browsers that you can click to view a text version of the report, or the Save icon () for the Internet Explorer browser so that you can save the report to a file (preset to .txt).

Click one of the records to open the View Subnet Utilization Record page for that record.



CHAPTER 10

Managing Hosts

This chapter explains how to configure hosts in DNS zones. Before you proceed with the concepts in this chapter, read Chapter 15, "Managing Zones," which explains how to set up the basic properties of a primary and secondary DNS server and its zones.

See Also

Managing Hosts in Zones
Adding Additional RRs for the Host, page 10-2
Editing Hosts, page 10-3
Removing Hosts, page 10-3

Managing Hosts in Zones

You can manage the resource records (RRs) for a host by configuring the host rather than the individual RRs. When you define a host, the DNS server automatically creates an Address (A) RR in IPv4, or an AAAA RR in IPv6, for each address you specify. If you specify one or more aliases for the host, the server also creates a Canonical Name (CNAME) RR for each alias. You can also have the server create a Pointer (PTR) RR for the host in the reverse zone for the host, if the reverse zone exists.

Local Basic or Advanced Web UI

Step 1 Click **Hosts**.

Step 2 If you have only one zone configured, the List/Add Hosts for Zone page appears and the **Zones** tab is inactive. If you have multiple zones configured, also click **Zones**, then, on the List Zones page, click the name of the zone to open the List/Add Hosts for Zone page (see Figure 5-8 on page 5-21).

Tip You can sort by hostname, IP address, IPv6 address (if appropriate), or alias by clicking the corresponding column heading on the List/Add Host for Zone page. However, for zones with a large number of hosts (more than 50,000), restrict the sort to the hostname. Sorting based on IP address or alias can take significantly longer, and could fail if you exceed the memory capacity of the CCM server.

Step 3 Enter the name of the host and its IPv4 or IPv6 address or comma-separated addresses.

Step 4 If the host has alias names, enter a comma-separated list.

Step 5 If you want to create a corresponding Pointer (PTR) RR for the host and you know that the reverse zone for the host exists, click a check mark in the Create PTR Records? box.

Step 6 Click **Add Host**.

Step 7 To confirm, click **DNS**, then **Forward Zones** to open the List/Add Zones page (see Figure 5-7 on page 5-20).

Step 8 Click the View icon () in the RRs column in the row of the zone name. This opens the List/Add DNS Server RRs for Zone page.

CLI Commands

To create A RRs, alias RRs, and PTR RRs for existing reverse zones in a single operation, use **zone** *name* **addHost** *hostname address alias* for each host. To list the created zones, use **zone** *name* **listHosts**.

Adding Additional RRs for the Host

You add additional RRs for the host based on the zone edit mode you chose, either staged or synchronous. For details, see the "Adding Resource Records" section on page 16-2.

Reload the DNS server if you want these RRs to become active server RRs.

Local Basic or Advanced Web UI

For example, to add additional CNAME RRs, add the alias hostname in the Name field of the List/Add DNS Server RRs for Zones page (see Figure 5-7 on page 5-20), choose **CNAME** from the Type drop-down list, add the canonical name of the host in the Data field, then click **Add Resource Record**. Note that the DNS specification does not allow a CNAME RR with the same name as that of another RR.

For an MX RR, add the origin hostname in the Name field; choose **MX** from the Type drop-down list; add the integer preference value, a space, and the domain name of the mail exchanger for the origin host in the Data field; then click **Add Resource Record**. These entries should appear in the list at the bottom of the page.

CLI Commands

To create a CNAME record, use **zone** *name* **addRR** *alias* **CNAME** *canonical* for protected RRs or **zone** *name* **addDNSRR** *alias* **CNAME** *canonical* for unprotected RRs. To create an MX record, use **zone** *name* **addRR** *hostname* **MX** *preference mxname* for protected RRs or **zone** *name* **addDNSRR** *hostname* **MX** *preference mxname* for unprotected RRs.

Editing Hosts

Editing a host involves modifying its RRs.

Local Basic or Advanced Web UI

Step 1 Click **Hosts**. If you get the List Zones page, click the zone name to open the List/Add Hosts for Zone page. If there is only one zone, you immediately go to the List/Add Hosts for Zone page. Another alternative, if you are on the List/Add Zones page, is to click the View icon ( ) in the Hosts column in the row of the zone name to open the List/Add Hosts for Zone page.

Step 2 Click the hostname to open the Edit Host page.

Step 3 You can add additional addresses, aliases, and other RRs for the host. Click the appropriate button in each case. You can also delete an RR by clicking the Delete icon ( ) next to its name (without a confirmation requested).

Step 4 Click **Modify Host**.

CLI Commands

To edit the host, you must remove and reenter its RRs by using **zone** *name* **removeRR** *name type data* or **zone** *name* **removeDNSRR** *name type data*, then **zone** *name* **addRR** *name ttl class type data* or **zone** *name* **addDNSRR** *name ttl type data*.

Removing Hosts

Removing a host removes all A, CNAME, and PTR RRs for that host.

Local Basic or Advanced Web UI

On the List/Add Hosts in Zone page (see the "Editing Hosts" section on page 10-3 for the possible ways to get there), click the Delete icon ( ) next to the host you want to remove, then confirm the deletion.

CLI Commands

Remove the host by using **zone** *name* **removeHost**, then re-add it by using **zone** *name* **addHost**.

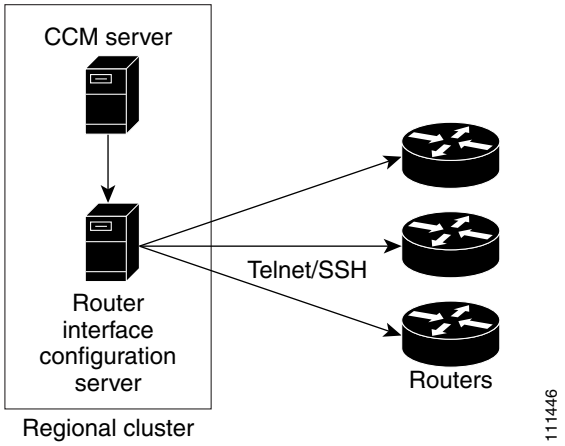


CHAPTER 11

Managing Router Interface Configurations

The regional Router Interface Configuration (RIC) server is used to manage router interfaces on Cisco Systems Universal Broadband Routers (uBRs) that manage cable modem termination systems (CMTSs). This module interacts with the CMTS servers to push the required cable modem configuration to edge devices, such as routers and switches (see Figure 11-1). The RIC server module is accessible only if you are assigned the ric-management subrole of the cfg-admin or central-cfg-admin role.

Figure 11-1 Router Interface Configuration (RIC) Server Module



Tip Add routers before you add any other subnets. This prevents the subnets that the router creates from possibly overlapping with those explicitly added, and prevents router synchronization errors later on.

The view of the routers is available on the View Tree of Routers page. The tree levels are the routers, their router interfaces, and any child interfaces. Parent/child relationships can be either physical/virtual (as in Cable2/0 and Cable2/0.1) or primary/secondary (as in router interface bundling, where the bundle is identified by one of the interfaces; see the "Bundling Interfaces" section on page 11-5). This listing of router interfaces is available only once you create routers in the system and synchronize with them.

See Also

Adding Routers
Editing Routers, page 11-3
Resynchronizing Routers, page 11-4
Pushing and Reclaiming Subnets for Routers, page 11-4
Viewing and Editing the Router Interfaces, page 11-4

Adding Routers

The routers that the RIC server manages can be Cisco Universal Broadband Routers in the family uBR72*xx* and uBR10*xxx*. (For an example of adding a router, see the "Add a Router and Modify an Interface" section on page 5-28.)

Local Advanced and Regional Web UI

Step 1 Click **Routers**, then **Router List**. This opens the List Routers page.

Step 2 Click **Add Router**. This opens the Add Router page (see Figure 5-14 on page 5-29).

Step 3 You can set a router to managed or virtual (see the "Managed Versus Virtual Routers" section on page 11-2). If the router is managed, you must enter its type and IP address. The selections for the Type field are **Ubr72xx** or **Ubr10k**. If managed, you also need to check with the router administrator about the username, password, and enable password, and enter these values.

Step 4 Click **Add Router**.

CLI Commands

Add a router using **router** *name* **create**. For example:

```
nrcmd> router router-1 create 192.168.121.121
```

Managed Versus Virtual Routers

Managed routers are updated in the database as well as being physically updated and synchronized. When you edit a managed router in the web UI or CLI, the router is also automatically updated and synchronized with the newest data.

Virtual routers are updated in the Network Registrar database only. However, you can create, push, and reclaim subnets for a virtual router. You might define a virtual router when the RIC server cannot directly manage the router, but the virtual router should still be considered part of the topology.

You define a virtual router by omitting the router type or connection credentials on the Add Router page (see Figure 5-14 on page 5-29) or Edit Router page.

Secure Mode Connections with Routers

To enable secure communication between the RIC server and the routers, you must have the Cisco Network Registrar Communications Security Option Release 1.1 installed. By default, secure connectivity is disabled and accessible over Telnet. However, you can specify whether you require or desire a Secure Shell (SSH) connection. Use the *use-ssh* attribute in the (expandable) Reserved attributes section of the Edit Router page in the web UI. This attribute has the following values:

- **disabled** (preset value)—Uses simple Telnet for the connection.
- **required**—The router communicates with the edge device using SSH only, and not Telnet.
- **desired**—The router tries to communicate using SSH, but if it cannot, it uses Telnet.

Note The SSH server should be set up so that the key length (modulus) is at least 1024 bits, using, for example, the command **crypto key generate rsa general-keys modulus 1024**.

Alternative Login Method to Routers

There are two types of login mechanisms provided in the RIC server that you can affect using the *login-template* attribute on the Add Router page:

- **Discovery mode**—The default mechanism, designed to understand login prompts on edge devices and respond to those dynamically. It does not force a particular login sequence, but supports the various login sequences and login prompts most customers use with these default prompts:

```
Username prompt - Username:
Password prompt - Password:
Login-prompt - >
Enable password prompt - Password:
Enable prompt - #
```

- **Template mode**—Use this in case the RIC server cannot log in using the discovery mechanism for some reason, such as nonstandard prompts or a login sequence that the discovery mechanism does not understand. The *login-template* is the name of an optional login template to use to further customize the RIC server login and enable interactive sessions. To create this template you must:
  1. In the API, create an ScpObj of class CCMRouterLoginTemplate.
  2. Add the object to the database using the RICAdminSession.addRouterLoginTemplate method.
  3. Enter the name of the added template (CCMRouterLoginTemplate.name) as the value of the *login-template*.

Editing Routers

Editing routers involves modifying some of the router attributes.

Local Advanced and Regional Web UI

Click the router name on the View Tree of Routers page or List Routers page. The Edit Router page is essentially the same as the Add Router page, except for an additional attribute unset function. Make your changes, then click **Modify Router**.

CLI Commands

Edit a router attribute using **router** *name* **set** *attribute*. For example:

```
nrcmd> router router-1 set owner=owner-1
```

Resynchronizing Routers

As soon as you add the router to the regional cluster, it is synchronized over the network. You can also explicitly resynchronize the router if you know that changes occurred. On the List Routers page, click the Resynchronize icon () next to the router name. If the synchronization could not occur or timed out, you get an error message to that effect.

Pushing and Reclaiming Subnets for Routers

You can push subnets to, and reclaim subnets from, a router interface (see the "Reclaiming Subnets" section on page 9-10). When you push or reclaim a subnet with a managed or virtual router, all primary and secondary relationships that are set for the router are also set for the related subnets and scopes.

Viewing and Editing the Router Interfaces

Editing a router interface involves modifying some of its attributes.

Local Advanced and Regional Web UI

If you click the Interface icon () associated with the router on the List Routers page, the list of related cable or Ethernet interfaces appears on the List Router Interfaces page. Both from this page and the View Tree of Routers page, you can click the interface name to edit it. The List Router Interfaces page includes an additional attribute unset function and the ability to delete the interface. You can add, edit, or delete interfaces for virtual routers without restrictions. There are restricted attributes for managed routers, described in the "Changeable Router Interface Attributes" section on page 11-5.

CLI Commands

Edit a router interface attribute using **router-interface** *name* **set** *attribute*. For example:

```
nrcmd> router-interface Ethernet1/0 set ip-helper=192.168.121.122
```

See Also

Changeable Router Interface Attributes, page 11-5
Bundling Interfaces, page 11-5

Changeable Router Interface Attributes

Editing the router interface opens the Edit Router Interface page. You cannot change the name, state, or MAC address of the interface on this page. However, you can change the following attributes:

- Description
- Address of the primary subnet address on the interface
- Addresses of the secondary subnets on the interface
- Address of any IP helper (DHCP relay agent) for the interface
- Address of any cable helper of the DHCP server to accept unicast packets for the interface

Bundling Interfaces

An interface bundle provides load balancing among the router interfaces. When you define a bundle, all the participating interfaces in the bundle must have the same bundle identifier (ID), which is the name of the interface specified as the master.

If you want to use bundling, the following attributes are in the Interface Bundling Settings section of the Edit Router Interface page, or set them using the **router-interface** command in the CLI:

- ***bundle-id***—Interface bundle identifier, the name of the master interface. All participating interfaces in the bundle must have the same bundle ID.
- ***is-master***—This interface is the master interface in the bundle.
- ***is-virtual***—This interface is a virtual interface in the bundle.



CHAPTER 12

Managing Owners and Regions

This chapter explains how to configure owners and regions that can be applied to DHCP address blocks, subnets, and zones.

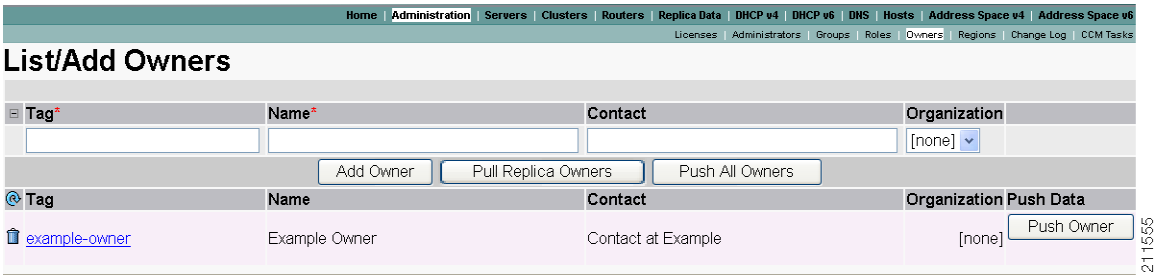
Managing Owners

You can create owners to associate with address blocks, subnets, and zones. You can list and add owners on a single page. Creating an owner involves creating a tag name, full name, and contact name.

Local Advanced and Regional Web UI

Step 1 Click **Administration**, then **Owners** to open the List/Add Owners page (see Figure 12-1 for the regional cluster version that also includes pull and push functions).

Figure 12-1 List/Add Owners Page (Regional)



Step 2 Enter a unique owner tag.

Step 3 Enter an owner name.

Step 4 Enter an optional contact name.

Step 5 Click **Add Owner**.

Step 6 To edit an owner, click its name to open the Edit Owners page.

CLI Commands

Use **owner** *tag* **create** *name* to create an owner. For example:

```
nrcmd> owner owner-1 create "First Owner" contact="Contact at owner-1"
```

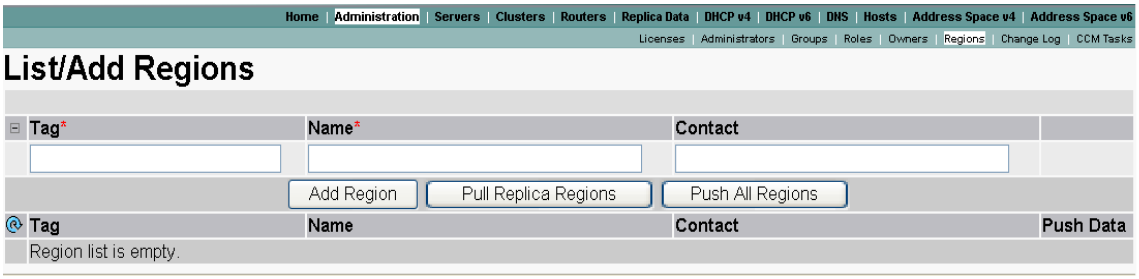
Managing Regions

You can create regions to associate with address blocks, subnets, and zones. You can list and add regions on a single page. Creating a region involves creating a tag name, full name, and contact name.

Local Advanced and Regional Web UI

Step 1 Click **Administration**, then **Regions** to open the List/Add Regions page (see Figure 12-2 for the regional cluster version that also includes pull and push functions).

Figure 12-2 List/Add Regions Page (Regional)



Step 2 Enter a unique region tag.

Step 3 Enter a region name.

Step 4 Enter an optional contact name.

Step 5 Click **Add Region**.

Step 6 To edit a region, click its name to open the Edit Regions page.

CLI Commands

Use **region** *tag* **create** *name*. For example:

```
nrcmd> region region-1 create "Boston Region" contact="Contact at region-1"
```



CHAPTER 13

Managing Reports

This chapter explains how to manage the Cisco CNS Network Registrar address space reporting tool, which is available from a regional cluster by using the web UI. Before you proceed with this chapter, become familiar with the concepts in the previous chapters of this part of the *User's Guide*.

See Also

ARIN Reports and Allocation Reports
Managing ARIN Reports, page 13-1

ARIN Reports and Allocation Reports

Using the Network Registrar web UI, you can generate:

- American Registry of Internet Numbers (ARIN) reports, including:
  - Organization and point of contact (POC) reports
  - IPv4 address space utilization reports
  - Shared WHOIS project (SWIP) allocation and assignment reports
- Allocation reports that show how addresses are deployed across the routers and router interfaces of your network, including:
  - Allocation by owner reports
  - Allocation by router interface or by network reports

Managing ARIN Reports

ARIN, which is one of several regional Internet Registries (IRs), manages IP resources in North America, parts of the Caribbean, and subequatorial Africa.

ARIN allocates blocks of IP addresses to Internet Service Providers (ISPs), which, in turn, reassign blocks of address space to their customers. ARIN distinguishes between *allocating* IP address space and *assigning* IP address space. It allocates address space to smaller IRs for subsequent distribution to the IRs' members and customers. It assigns address space to an ISP, or other organization, for use only within the network of that organization and only for the purposes documented in its requests and reports to ARIN.

Note ARIN manages IP address resources under the auspices of the Internet Corporation for Assigned Names and Numbers (ICANN). In other geographies, ICANN has delegated authority for IP resources to different regional Internet Registries. Network Registrar does not currently support the reports that these registries might require, nor does it now support IPv6 reports or autonomous system (AS) numbers.

ARIN maintains detailed documentation about its policies and guidelines on its website.

`http://www.arin.net`

Be sure that you are familiar with these policies and guidelines before proceeding with ARIN reports.

The three options that you can specify for ARIN reports are:

- **New**—For a newly added POC or organization.
- **Modify**—Includes changed POC or organization data, such as phone numbers and addresses.
- **Remove**—Signals that you want to remove the POC or organization from the ARIN database.

See Also

Managing Point of Contact and Organization Reports
Managing IPv4 Address Space Utilization Reports, page 13-7
Managing Shared WHOIS Project Allocation and Assignment Reports, page 13-8

Managing Point of Contact and Organization Reports

Network Registrar provides reports that can submit Points of Contact (POC) and organizational information to ARIN. Once you fill in these reports, you need to e-mail the information to ARIN. Submit the POC report (also called a template) to ARIN before preparing other reports.

Each POC is uniquely identified by a name called a POC handle and is associated with one or more Organization Identifiers (Org IDs) or resource delegations, such as an IP address space allocation or assignment. A POC handle, which ARIN assigns, can represent either an individual or a role.

The Organization report creates an Org ID and associates POC records with it. Create the Organization report after you create the POC report.

To manage POC and organization reports, log in to the Network Registrar regional web UI as a member of an administrator group assigned to the regional-addr-admin role.

See Also

Creating a Point of Contact Report, page 13-3
Registering a Point of Contact, page 13-4
Editing a Point of Contact Report, page 13-4
Creating an Organization Report, page 13-5
Registering an Organization, page 13-6
Editing an Organization Report, page 13-6

Creating a Point of Contact Report

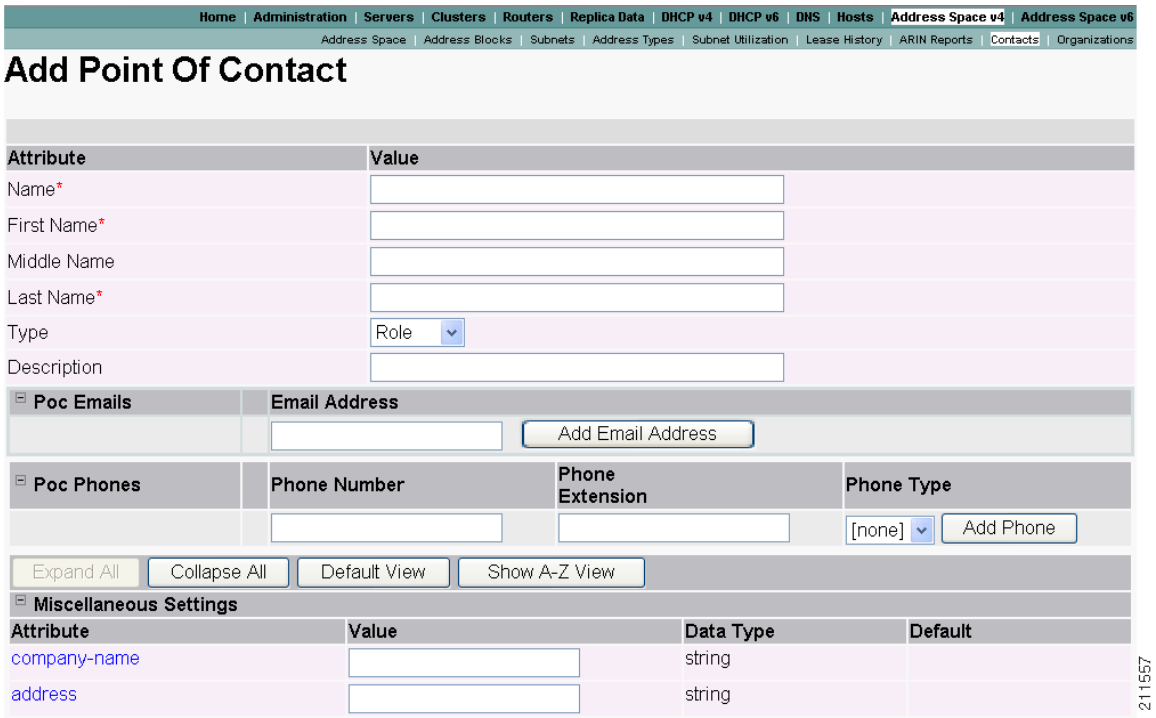
You create POCs so that managers can interact with ARIN to request and administer IP resources and so that network professionals can manage network operation issues.

Regional Web UI

Step 1 Click **Address Space**, then **Contacts** to open the List Points of Contact page.

Step 2 Click **Add Point of Contact** to open the Add Point of Contact page (see Figure 13-1 for a partial view).

Figure 13-1 Add Point of Contact Page (Regional)



Step 3 Enter data in the fields on the page:

- **Name**—A unique identifier for the POC (required).
- **First Name**—The first name of the point of contact (required).
- **Middle Name**—The middle name of the point of contact (optional).
- **Last Name**—The last name of the point of contact (required).
- **Type**—From the drop-down list, choose Person or Role (optional, with preset value Person).
- *Description*—A text description for the POC (optional).

Step 4 To expand the optional Poc Emails field, click the plus (+) sign.

a. Enter the e-mail address for the POC.

b. Click **Add Email Address** to add additional e-mail addresses.

Step 5 To expand the optional Poc Phones field, click the plus (+) sign.

Step 6 Enter a phone number and extension, if applicable, then choose a type (Office, Mobile, Fax, or Pager) from the drop-down list, then click **Add Phone** to add additional telephones. This is an optional field.

Step 7 Miscellaneous Settings. Add these additional attributes as strings or lists of text (optional).

Step 8 When you are finished, click **Add Point of Contact**. The browser displays the List Points of Contact page.

Registering a Point of Contact

You must register the POC with ARIN to receive a POC handle.

Regional Web UI

Step 1 Click **Reports**, then **Contacts** to open the List Points of Contact page.

Step 2 In the Register Report column on the List Points of Contact page, click the Report icon ( ) next to the report you want to register. The browser displays an ARIN template file, as shown in Figure 13-2.

Figure 13-2 ARIN POC Template Report (Regional)

```
Template: ARIN-POC-3.2.1
1. Registration Action: N
2. Handle:
3. Contact Type: P
4. Last Name Or Role Account: Last
5. First Name: First
6. Middle Name:
7. Additional Information:
8. Company Name: Company B
9a. Address: Main St.
9b. Address:
10. City:  Anytown
11. State/Province: AK
12. Postal Code:
13. Country Code: US
14. Phone Type:
15. Phone Number:
16. Phone Extension:
17. E-mail Address: person@example.com
18. Public Comments:

END OF TEMPLATE
```

149392

Step 3 Copy and paste the template file into an e-mail and send the file to ARIN.

Editing a Point of Contact Report

Edit a POC report after ARIN returns a POC handle to your organization or if your POC has changed.

Regional Web UI

Step 1 Click **Reports**, then **Contacts** to open the List Points of Contact page.

Step 2 Click the POC that you want to edit. The browser displays the Edit Point of Contact page.

Step 3 Enter data in the fields, according to whether you are adding a POC handle or changing your POC.

Step 4 When you are finished, click **Modify Point of Contact**.

Creating an Organization Report

Each organization is represented in the ARIN WHOIS database by a unique Org ID, consisting of an organization name, its postal address, and its POCs. While organizations may have more than one Org ID, ARIN recommends consolidating IP address resources under a single Org ID.

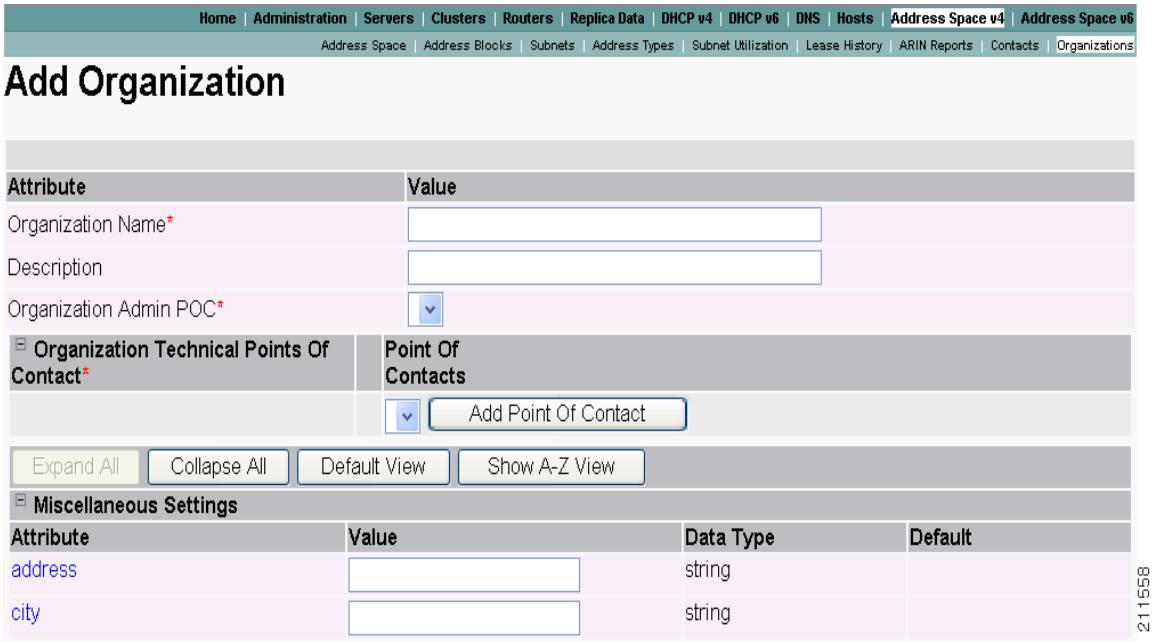
If you do not have an Org ID with ARIN, or you are establishing an additional Org ID, you must first create and submit a POC report. When ARIN confirms it has received your POC information, use Network Registrar to complete an Organization form and submit that information.

Regional Web UI

Step 1 Click **Address Space**, then **Organizations** to open the List Organizations page.

Step 2 Click **Add Organization**. The browser displays the Add Organization page (see Figure 13-3 for a partial view).

Figure 13-3 Add Organization Page (Regional)



Step 3 Enter data in the fields on the page:

- **Organization Name**—Name of the organization that you want to register with ARIN.
- **Description**—A text description of the organization.
- **Organization Admin POC**—From the drop-down list, choose the POC who administers IP resources from the drop-down list.
- **Organization Technical Points Of Contact**—From the drop-down list, choose one or more POCs who manage network operations, or click **Add Point of Contact** to add new contact information.
- **Miscellaneous Settings**—Add these additional attributes as strings or lists of text.
- **Organization Abuse Points of Contact**—From the drop-down list, choose one or more POCs who handle network abuse complaints, or click **Add Point of Contact** to add new contact information.
- **Organization NOC Points of Contact**—From the drop-down list, choose one or more POCs in network operations centers, or click **Add Point of Contact** to add new contact information.

Step 4 When you are finished, click **Add Organization**. The browser displays the List Organizations page.

Registering an Organization

You must register your Organization with ARIN to receive an Organization ID.

Regional Web UI

Step 1 Click **Reports**, then **Organization** to open the Organizations page.

Step 2 In the Register Report column of the List Organizations page, click the Report icon () next to the report you want to register. The browser displays an ARIN template file.

Step 3 Copy and paste the template file into an e-mail and send the file to ARIN.

Editing an Organization Report

You might need to change organizational information that you have registered with ARIN.

Regional Web UI

Step 1 Click **Reports**, then **Organization** to open the List Organizations page.

Step 2 Click the organization that you want to edit. The browser displays the Edit Organization page.

Step 3 Enter or change data in the fields.

Step 4 When you are finished, click **Modify Organization**.

Step 5 Submit the updated report to ARIN as described in the "Registering an Organization" section on page 13-6.

Managing IPv4 Address Space Utilization Reports

Address space utilization reports serve two purposes:

- To make an initial request for IPv4 address space after you receive a POC handle and an Org ID.
- To support a request for an additional allocation of IPv4 addresses when your business projections show that you are running out of IP addresses.

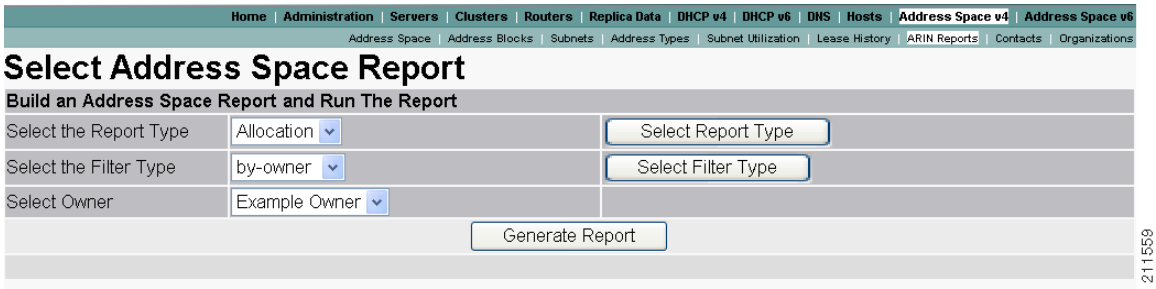
Note The ARIN website contains extensive information about how it initially allocates address space and its threshold criteria for requesting additional address space. In general, for a single-homed organization, the minimum allocation from ARIN is a /20 block of addresses. For a multihomed organization, the minimum allocation is a /22 block of addresses. ARIN recommends that an organization requiring a smaller block of addresses contact an upstream ISP to obtain addresses.

The Network Registrar utilization report corresponds to the ARIN ISP Network Request template (ARIN-NET-ISP-3.2.2).

Regional Web UI

Step 1 Click **Address Space**, then **ARIN Reports** to open the Select Address Space Report page (see Figure 13-4).

Figure 13-4 Select Address Space Report Page (Regional)



Step 2 In the Select the Report Type field, choose Utilization from the drop-down list and click **Select Report Type**.

Step 3 In the Select the Filter Type field, choose by-owner from the drop-down list and click **Select Filter Type**. The browser redisplays the Select Address Space Report page with two new fields: Network Name and Network Prefix Length.

Step 4 In the Select Owner field, choose the owner of this address block from the drop-down list.

Step 5 Enter the network name and select the network prefix length from the drop-down list.

Step 6 Click **Generate Report**. The browser displays an ARIN template file (ARIN-NET-ISP-3.2.2).

Several sections of the report require that you manually enter data because the information is generated and maintained outside the Network Registrar application.

Step 7 Click **Save Report**. The browser displays the Address Space Utilization Report as an unformatted text file.

Step 8 Copy the Address Space Utilization Report to a text editor to manually enter the data that Network Registrar does not generate.

Step 9 Copy and paste the edited report into an e-mail and send the file to ARIN.

Managing Shared WHOIS Project Allocation and Assignment Reports

The ARIN shared WHOIS project (SWIP) provides a mechanism for finding contact and registration information for resources registered with ARIN. The ARIN database contains IP addresses, autonomous system numbers, organizations or customers that are associated with these resources, and related POCs.

The ARIN WHOIS does not locate any domain- or military-related information. Use whois.internic.net to locate domain information, and whois.nic.mil for military network information.

The regional web UI also provides two allocation and assignment report pages:

- View ARIN SWIP Reallocated Report
- View ARIN SWIP Reassigned Report



PART 4

Domain and Zone Administration



CHAPTER 14

Introduction to the Domain Name System

The Domain Name System (DNS) handles the growing number of Internet users. DNS translates names, such as www.cisco.com, into IP addresses, such as 192.168.40.0 (or the more extended IPv6 addresses), so that computers can communicate with each other. DNS makes using Internet applications, such as the World Wide Web, easy. The process is as if, when phoning your friends and relatives, you could autodial them based on their names instead of having to remember their phone numbers.

See Also

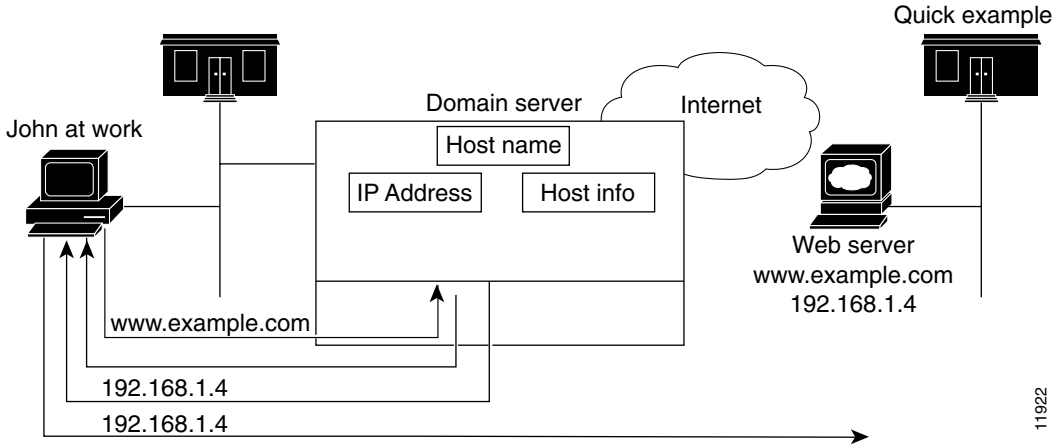
How DNS Works
Domains, page 14-2
Nameservers, page 14-5
Reverse Nameservers, page 14-6
High-Availability DNS, page 14-7

How DNS Works

To understand how DNS works, imagine a typical user, John, logging in to his computer. He launches his web browser so that he can view the website at a company, ExampleCo (see Figure 14-1 on page 14-2). He enters the name of their website—http://www.example.com. Then:

1. John's workstation sends a request to the DNS server about the IP address of www.example.com.
2. The DNS server checks its database to find that www.example.com corresponds to 192.168.1.4.
3. The server returns this address to John's browser.
4. The browser uses the address to locate the website.
5. The browser displays the website on John's monitor.

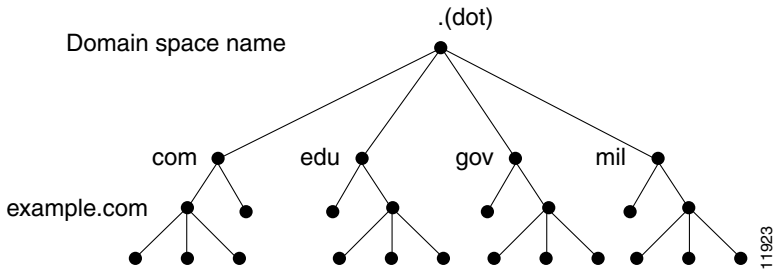
Figure 14-1 Domain Names and Addresses



Domains

John can access the ExampleCo website because his DNS server knows the www.example.com IP address. The server learned the address by searching through the domain namespace. DNS was designed as a tree structure, where each named domain is a node in the tree. The top-most node of the tree is the DNS root domain (.), under which there are subdomains, such as .com, .edu, .gov, and .mil (see Figure 14-2).

Figure 14-2 Domain Name System Hierarchy



The fully qualified domain name (FQDN) is a dot-separated string of all the network domains leading back to the root. This name is unique for each host on the Internet. The FQDN for the sample domain is example.com., with its domain example, parent domain .com, and root domain "." (dot).

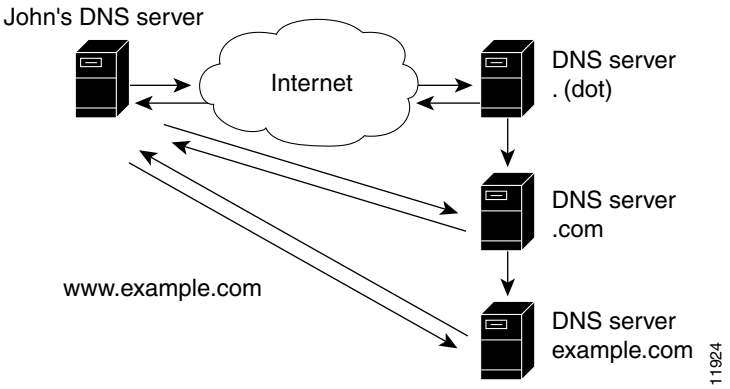
See Also

Learning ExampleCo Address, page 14-3
Establishing a Domain, page 14-3
Difference Between Domains and Zones, page 14-3

Learning ExampleCo Address

When John's workstation requests the IP address of the website www.example.com (see Figure 14-3):

Figure 14-3 DNS Hierarchical Name Search



1. The local DNS server looks for the www.example.com domain in its database, but cannot find it, indicating that the server is not authoritative for this domain.
2. The server asks the authoritative root nameserver for the top-level (root) domain "." (dot).
3. The root nameserver directs the query to a nameserver for the .com domain that knows about its subdomains.
4. The .com nameserver determines that example.com is one of its subdomains and responds with its server address.
5. The local server asks the example.com nameserver for the www.example.com location.
6. The example.com nameserver replies that its address is 192.168.1.4.
7. The local server sends this address to John's Web browser.

Establishing a Domain

ExampleCo has a website that John could reach because it registered its domain with an accredited domain registry. ExampleCo also entered its domain name in the .com server database, and requested a network number, which defines a range of IP addresses. In this case, the network number is 192.168.1.0, which includes all addresses in the range 192.168.1.1 through 192.168.1.255. You can only have the numbers 0 through 256 (2^8) in each of the address fields, known as octets. However, the numbers 0 and 256 are reserved for network and broadcast addresses, respectively, and are not used for hosts.

Difference Between Domains and Zones

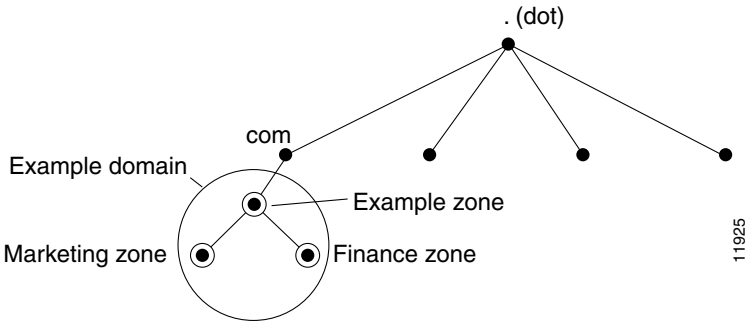
The domain namespace is divided into areas called zones that are points of delegation in the DNS tree. A zone contains all domains from a certain point downward, except those for which other zones are authoritative.

A zone usually has an authoritative nameserver, often more than one. In an organization, you can have many nameservers, but Internet clients can query only those that the root nameservers know. The other nameservers answer internal queries only.

The ExampleCo company registered its domain, example.com. It established three zones—example.com, marketing.example.com, and finance.example.com. ExampleCo delegated authority for marketing.example.com and finance.example.com to the DNS servers in the Marketing and Finance groups in the company. If someone queries example.com about hosts in marketing.example.com, example.com directs the query to the marketing.example.com nameserver.

In Figure 14-4, the domain example.com includes three zones, with the example.com zone being authoritative only for itself.

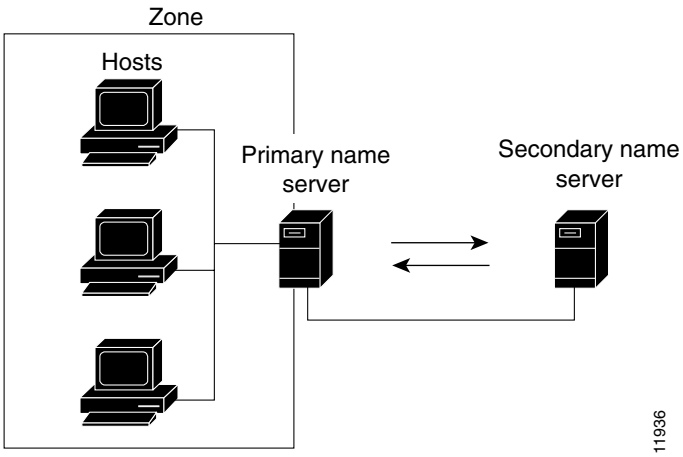
Figure 14-4 Example.com With Delegated Subdomains



ExampleCo could choose not to delegate authority to its subdomains. In that situation, the example.com domain is a zone that is authoritative for the subdomains for marketing and finance. The example.com server answers all outside queries about marketing and finance.

As you begin to configure zones by using Network Registrar, you must configure a nameserver for each zone. Each zone has one primary server, which loads the zone contents from a local configuration database. Each zone can also have any number of secondary servers, which load the zone contents by fetching the data from the primary server. Figure 14-5 shows a configuration with one secondary server.

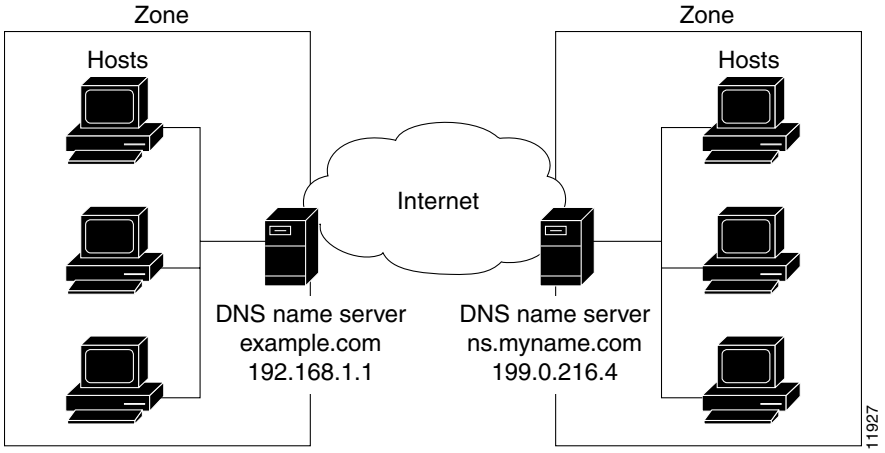
Figure 14-5 Primary and Secondary Servers for Zones



Nameservers

DNS is based on a client/server model. In this model, nameservers store data about a portion of the DNS database and provide it to clients that query the nameserver across the network. Nameservers are programs that run on a physical host and store zone data. As administrator for a domain, you set up a nameserver with the database of all the resource records (RRs) describing the hosts in your zone or zones (see Figure 14-6). For details about DNS RRs, see Appendix A, "Resource Records."

Figure 14-6 Client/Server Name Resolution



The DNS servers provide name-to-address translation, or name resolution. They interpret the information in a fully qualified domain name (FQDN) to find its address. If a local nameserver does not have the data requested in a query, it asks other nameservers until it finds it. For commonly requested names, this process can go quickly, because nameservers continuously cache the information they learn from queries about the domain namespace.

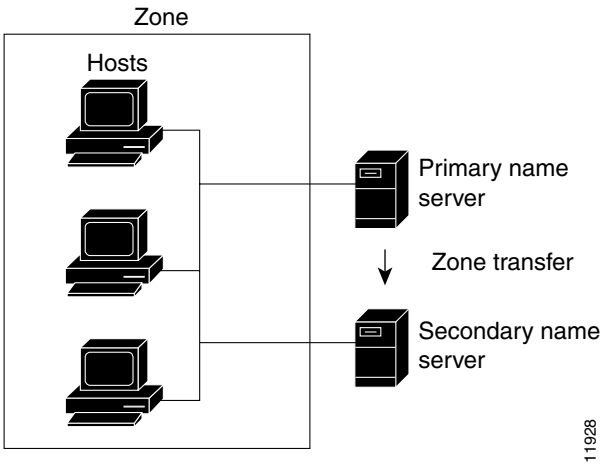
Each zone must have one primary nameserver that loads the zone contents from a local database, and a number of secondary servers, which load a copy of the data from the primary server (see Figure 14-7 on page 14-6). This process of updating the secondary server from the primary server is called a zone transfer.

Even though a secondary nameserver acts as a kind of backup to a primary server, both types of servers can be authoritative for the zone. They both learn about hostnames in the zone from the zone authoritative database, not from information learned while answering queries. Clients can query both servers for name resolution.

As you configure the Network Registrar DNS nameserver, you specify what role you want the server to perform for a zone—primary, secondary, or caching-only. The type of server is meaningful only in context to its role. A server can be a primary for some zones and a secondary for others. It can be a primary or secondary only, or it can serve no zones and just answer queries by means of its cache.

Although all servers are caching servers, because they save the information until it expires, a caching-only server is one that is not authoritative for any zone. This server answers internal queries and asks other authoritative servers for the information. Sites create caching-only servers to unburden the authoritative servers so that they do not need to have every query directed to the authoritative servers.

Figure 14-7 DNS Zone Transfer



To configure the:

- Primary nameserver, see the "Managing Primary DNS Servers" section on page 15-5.
- Secondary server, see the "Managing Secondary Servers" section on page 15-14.
- Caching-only server, see the "Configuring Caching-Only DNS Servers" section on page 17-6.

Reverse Nameservers

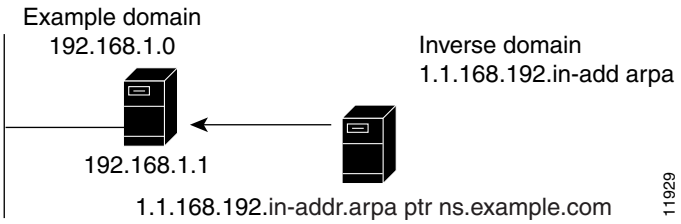
The DNS servers described so far perform name-to-address resolution. They can do this easily by searching through their database for the correct address, because they index all the data by name. However, there are times when you need address-to-name resolution so that you can interpret certain output, such as computer log files.

Finding a domain name when you only know the address, however, would require searching the entire namespace. DNS solves this problem by supporting a domain namespace that uses addresses as names, known as the in-addr.arpa domain. This reverse zone contains subdomains for each network based on the network number. For consistency and natural grouping, the four octets of a host number are reversed.

The IP address as a domain name appears backward, because the name is in leaf-to-root order. For example, the ExampleCo example domain network number is 192.168.1.0. Its reverse zone is 1.168.192.in-addr.arpa. If you only know the DNS server address (192.168.1.1), the query to the reverse domain would find the host entry 1.1.168.192.in-addr.arpa that maps back to example.com.

Reverse domains are handled through Pointer (PTR) RRs, as indicated in Figure 14-8.

Figure 14-8 Reverse Domains



High-Availability DNS

Because there can be only one primary DNS server per zone, you risk the failure of dynamic updates if the primary DNS server goes down. These updates can occur on the primary DNS server only; a secondary DNS server cannot record these changes, but must forward them to the primary. To solve this problem, a second primary server can become a hot standby that shadows the main primary. This is called High-Availability (HA) DNS (see Chapter 18, "Configuring High-Availability DNS Servers"). Both servers in this failover configuration must synchronize so that their primary zones and related attributes are identical. Network Registrar provides settings on the main server to identify the main and backup for synchronization, and the timeout period to go over into failover mode.



CHAPTER 15

Managing Zones

The Domain Name System (DNS) is a distributed database for objects in a computer network. By using a nameserver approach, the network consists of a hierarchy of autonomous domains and zones. The namespace is organized as a tree that often resembles the organizations that are responsible for the administration boundaries. For an introduction to the protocol, see Chapter 14, "Introduction to the Domain Name System."

The basic function of DNS nameservers is to provide data about network objects by answering queries. You can configure the Cisco Network Registrar DNS server and zones by accepting the system defaults or changing them.

This chapter describes the basics of configuring the Network Registrar DNS servers, and their primary and secondary zones. Chapter 16, "Managing Resource Records," describes how to manage DNS resource records (RRs) and hosts, and Chapter 17, "Managing DNS Server Properties," describes how to set some of the more advanced zone and DNS server properties.

See Also

Staged and Synchronous Modes
Creating and Applying Zone Templates, page 15-2
Managing Primary DNS Servers, page 15-5
Managing Secondary Servers, page 15-14
Adding Subzones, page 15-16
Enabling DNS Updates, page 15-19
Managing Zone Distributions, page 15-19

Staged and Synchronous Modes

You can perform additions or edits to DNS zones, RRs, and hosts in one of two modes—staged or synchronous:

- **Staged (or CCM)**—Changes to zones (and their hosts and protected server RRs) are written to the CCM database, but not immediately propagated to the DNS server until a synchronization is requested. This mode is reflected on the List/Add CCM Server Protected RRs for Zone page.
- **Synchronous (or DNS)**—After committing changes to CCM, hosts and protected RRs are immediately propagated to the DNS server. If propagation cannot occur because of an unreachable server, RRs are propagated at the next synchronization. This mode is reflected on the List/Add DNS Server RRs for Zone page (see Figure 16-1 on page 16-3).

Note Synchronous zone edit mode is the default value for the local cluster. Because of this, the procedures in this *User Guide* do not include a specific step to reload the DNS server. If Staged mode is in effect, assume an implicit server reload as part of most procedures.

Synchronizations can occur on a zone basis or by creating a zone distribution. In synchronous mode, changes are written to the DNS server right away, even though a server reload is necessary for the zone to be published on the network.

Local Basic or Advanced and Regional Web UI

Staged or synchronous zone modes are preset based on the Set Zone Edit mode setting in Session Settings on the Main Menu page:

- The regional web UI is preset to **staged**.
- The local web UI is preset to **synchronous**.

CLI Commands

Set the session *zone-edit-mode* attribute to staged or synchronous. For example:

```
nrcmd> session set zone-edit-mode=sync
```

Creating and Applying Zone Templates

A zone template is a convenient way to create a boilerplate for primary zones that share many of the same attributes. You can apply a zone template to any zone, and override the zone attributes with those of the template. You can create zone templates in the local and regional cluster web UIs and in the CLI.

Caution Be careful applying a template to an existing zone. The template overwrites all explicitly set attributes for the zone (other than its name), possibly causing severe consequences if the zone is already configured in a network. To make a limited attribute change to multiple zones using a template, be sure to change only that attribute (or attributes), leaving the others unset, before you apply the template to the zones.

Local Basic or Advanced and Regional Web UI

Step 1 Click **DNS**, then **Zone Templates** (see Figure 15-1 on page 15-3 for the regional cluster version).

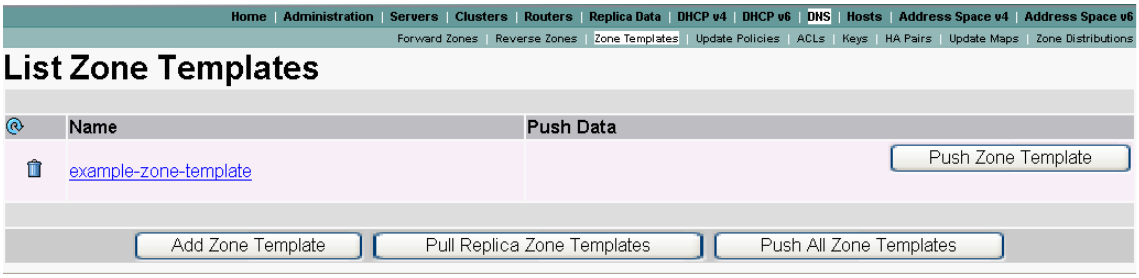
Step 2 You can add a zone template at the local and regional clusters, and you can also pull and push zone templates at the regional cluster in the web UI:

- To add a zone template at the local cluster or explicitly add one at the regional cluster, click **Add Zone Template**. This opens the Add Zone Template page, which is almost identical to the Add Zone page for the local cluster.

  To make the zone template meaningful, you would enter, in addition to its name, at least the suggested serial number, nameserver, contact e-mail address, and list of nameservers, because they are required for the zone itself. You might also want to specify any zone owners or zone distributions. You do not necessarily need to add these values for the zone template, because you

can do so for the zone once it is created from the template. However, the template name and zone default TTL are required. (For a description of the minimally required zone attributes, see the "Creating Primary Zones" section on page 15-6.)

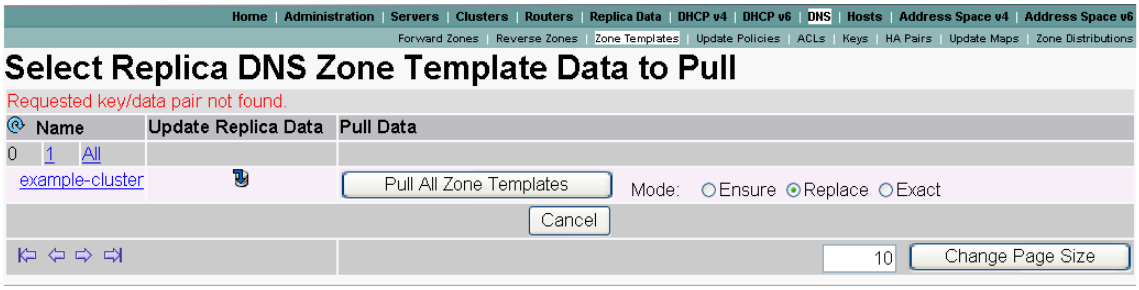
Figure 15-1 List Zone Templates Page (Regional)



Once you are done entering these value, click **Add Zone Template** at the bottom of the page.

- At the regional cluster, to pull a zone template from one or more local clusters, click **Pull Replica Zone Templates** on the List Zone Templates page. This opens the Select Replica DNS Zone Template Data to Pull page (see Figure 15-2 for an example that includes an error message).

Figure 15-2 Select Replica DNS Zone Template Data to Pull Page (Regional)



This page shows a tree view of the regional server replica data for the local clusters' zone templates. The tree has two levels, one for the local clusters and one for the templates in each cluster. You can pull individual templates from the clusters, or you can pull all of their templates:

- To pull individual zone templates, expand the tree for the cluster, choose a pull criterion next to its name, then click **Pull Zone Template**.
- To pull all the templates from a cluster, choose a pull criterion, then click **Pull All Zone Templates from Cluster**.
- To update all the replica data for a cluster, click the Replica icon ( ) next to its name.

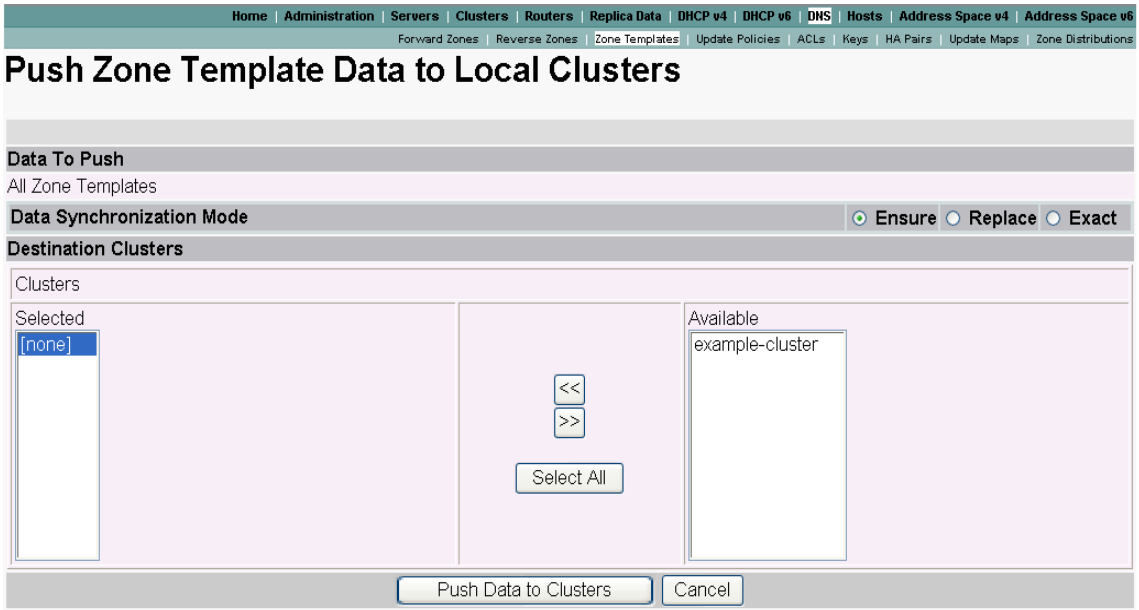
The pull selection criteria are:

- **Ensure**—Pulls each template, except if an existing template by that name already exists at the regional cluster, in which case it does not overwrite the regional cluster data.
- **Replace**—Pulls each template and overwrites the data for it if it already exists at the regional cluster, without affecting any additional templates at the regional cluster. This is the default and recommended setting.
- **Exact**—Pulls each template, overwrites the data for it if it already exists at the regional cluster, and removes any additional templates at the regional cluster.

- At the regional cluster, to push a zone template to one or more local clusters:
  - **To push all the zone templates on the page List Zone Templates page**—Click **Push All Zone Templates**.
  - **To push individual zone templates on the page List Zone Templates page**—Click **Push Zone Template** next to the template name.

  Both of these actions open a version of the Push Zone Template Data to Local Clusters page (see Figure 15-3).

Figure 15-3 Push Zone Template Data to Local Clusters Page (Regional)



This page provides a choice of the synchronization mode and the destination clusters. Move the desired cluster or clusters from the Available field to the Selected field, then click one of the data synchronization mode radio buttons:

- **Ensure**—Pushes each template, except if an existing template by that name already exists at the local cluster, in which case it does not overwrite the local cluster data. This is the default and recommended setting.
- **Replace**—Pushes each template and overwrites the data for it if it already exists at the local cluster, without affecting any additional templates at the local cluster.
- **Exact**—Available for "push all" operations only, it pushes each template, overwrites the data for it if it already exists at the local cluster, and removes any additional templates at the local cluster.

After making these choices, click **Push Data to Clusters**. This opens the View Push Zone Template Data Report page, where you can view the intended results of the push operation. Click **OK** to implement the push operation.

Step 3 You can apply the template to a new or existing zone:

a. **New zone**—Select the template from the Template drop-down list when you create the zone, as described in the "Configuring Primary Forward Zones" section on page 15-5.

b. **Existing zone**—After you create a zone (see the "Configuring Primary Forward Zones" section on page 15-5), you can apply the template when you edit the zone on the Edit Zone page. Click the template name in the Template drop-down list, then click **Apply Template**.

CLI Commands

Use **zone-template** *name* **create** to create the zone template. (See the "Configuring Primary Forward Zones" section on page 15-5 for how to apply the template to a zone.) For example:

```
nrcmd> zone-template zone-template-1 create serial=1
```

To apply a template to a zone, use **zone-template** *name* **apply-to** *zone*. Note that the syntax permits one or more comma-separated zones and also the **all** keyword for all zones. You can also clone a template from an existing template by using **zone-template** *clone-name* **create clone=***template*, and then make adjustments to the clone. For example:

```
nrcmd> zone-template zone-template-1 apply-to example.com,boston.example.com
nrcmd> zone-template cloned-template create clone=zone-template-1 owner=owner-1
```

Managing Primary DNS Servers

Adding a zone involves creating a domain name. You can also define an owner and use a zone template. If you do not use a template, you must also define the Start of Authority (SOA) and Name Server (NS) properties for the zone.

You do not need to create a loopback zone for the local host, because Network Registrar automatically creates one. A loopback zone is a reverse zone that a host uses to resolve its loopback address, 127.0.0.1, to localhost so that it can direct network traffic to itself. The loopback zone is 127.in-addr.arpa, which appears on the list of reverse zones.

See Also

Configuring Primary Forward Zones, page 15-5
Zone Lists and Zone Trees, page 15-12
Adding Primary Reverse Zones, page 15-12
Getting Zone Counts on the Server, page 15-13

Configuring Primary Forward Zones

This section explains how to configure a primary nameserver with a primary forward zone. When you are done with this procedure, follow the procedure in the "Adding Primary Reverse Zones" section on page 15-12 to configure a reverse zone for each network that you use.



Tip For an example of adding a forward zone, see the "Create the Zone Infrastructure" section on page 5-19.

See Also

Creating Primary Zones
Editing Primary Zones, page 15-8
Confirming Zone Nameservers, page 15-9
Synchronizing Zones and Zone Commands, page 15-9
Importing and Exporting Zone Data, page 15-9

Creating Primary Zones

Creating a primary zone requires, at a minimum, adding certain key Start of Authority (SOA) attributes and nameservers for the zone. The advantage of Basic mode in the web UI is that many of these settings are already done for you.

Local Basic Web UI

Step 1 Click **DNS**, then **Forward Zones** to open the List/Add Zones page (see Figure 5-6 on page 5-19).

Step 2 Enter the zone name (in domain name format).

Step 3 Enter the name of the nameserver host, such as **ns1**.

Step 4 Enter the contact e-mail name, such as **hostmaster**.

Step 5 Click **Add Zone**. Basic mode creates the zone with preset values:

- Zone default TTL—**24h**
- Start of Authority (SOA) serial number—**1**
- SOA secondary refresh time—**3h**
- SOA secondary retry time—**60m**
- SOA secondary expiration time—**1w**
- SOA minimum TTL—**10m**

Local Advanced and Regional Web UI

Step 1 Click **DNS**, then **Forward Zones** to open the List/Add Zones page. (At the regional cluster, these actions open the List Zones page.)

Step 2 Enter the zone name (in domain name format)

Step 3 Choose an owner or region, if necessary, from the drop-down list.

Step 4 Apply an existing zone template, if necessary (see the "Creating and Applying Zone Templates" section on page 15-2). Click the name of the configured template in the drop-down list.

Caution Be careful applying a template to a zone that is already live. Explicitly defined attributes on the template replace the existing ones defined for the zone.

Step 5 Click **Add Zone** to open the Add Zone page.

Step 6 Modify the top attributes, if necessary:

a. Owner and region

b. Preconfigured zone distribution (see the "Managing Zone Distributions" section on page 15-19)

c. Zone default TTL

Step 7 In the SOA attributes, enter a:

a. Serial number, such as **1**.

A primary DNS server uses a serial number to indicate when its database changes and uses any incrementing of this number to trigger a zone transfer to a secondary server. The serial number you can enter here is the *suggested* one only, and the DNS server does not always accept it. If you edit the serial number to be less than the actual serial number that the server maintains, the server logs a warning message and ignores the suggested serial number. The actual serial number always equals or is higher than the suggested one. You can get the actual serial number by using **zone** *name* **get serial** (if the DNS server is running; if the server is not running, or listing or showing the zone attributes, it always returns the suggested serial number), or by refreshing the DNS Server Value for the zone Serial Number attribute. You must explicitly enter this suggested serial number when creating a zone.

b. Nameserver host, such as **ns1**.

Enter either just the hostname or its fully qualified name (such as **ns1.example.com.**, but you must end it with a trailing dot). Use the fully qualified name if the primary nameserver is in a different zone. The primary DNS server becomes the *ns* value in the zone SOA record. You must also specify one or more authoritative nameservers for the zone—these become the Name Server (NS) records for the zone. In the CLI, the primary DNS server automatically becomes the first NS record and also appears as the first entry in the *nameservers* attribute list.

c. Contact e-mail name, such as **hostmaster**.

The fully qualified contact e-mail name becomes a slightly altered version of the e-mail address in that dots (.) are substituted for the at symbol (@). If using the fully qualified value, end the address with a trailing dot (for example, enter hostmaster@example.com as **hostmaster.example.com.**). Precede any dot before the @ in the original address with a backslash (\) (for example, enter hostmaster.marketing@example.com as **hostmaster\.marketing.example.com.***)*.

Step 8 Enter an authoritative nameserver name under Nameservers further down the page, then click **Add Nameserver**.

Authoritative nameservers validate the data in their zones. Both primary and secondary servers can be authoritative. The crucial difference is where they get their zone data. A primary server obtains its data from an administrator, as stored in the server configuration database, and from DNS updates, typically from a DHCP server. A secondary server obtains the zone data from its designated master servers by way of a zone transfer.

You must add at least one nameserver for a zone—Network Registrar does not consider the zone data complete unless you do so. The nameservers you list should be those that you want people outside your domain to query when trying to resolve names in your zone. You must add the authoritative nameservers in addition to the primary server for the zone. If the primary DNS server for the zone is in the zone, you must create a host address for it.

For every DNS internal-to-zone nameserver, you must create an Address (A) resource record (RR) to associate the server domain name with an IP address:

a. Click **Host** to open the List Zones page.

b. Click the zone name to open the List/Add Hosts for Zone page.

c. Enter the hostname of the authoritative server.

d. Enter its IP address.

e. Click **Add Host**. The server hostname and address appear in the list.

f. To edit the host, click its name to open the Edit Host page. Click **Modify Host** to implement the changes.

Step 9 Configure additional attributes as needed.

Step 10 Click **Add Zone**.

CLI Commands

To create a primary zone, use **zone** *name* **create primary** *nameserver contact*. You must specify a primary DNS server; this server becomes the first authoritative DNS nameserver. For example:

```
nrcmd> zone example.com create primary ns1 hostmaster
```

The serial number defaults to 1. You can get the actual serial number by using **zone** *name* **get serial** (if the DNS server is running; if the server is not running, or listing or showing the zone attributes, it always returns the suggested serial number).

To add additional authoritative nameservers for the zone, enter a comma-separated list of fully qualified domain names using **zone** *name* **set nameservers=***list*. Note that only the first server entered is confirmed by the command. Use **zone** *name* **show** to show all the server names.

Use **zone** *name* **addRR** *hostname* **A** *address* to add the authoritative server hostname and address. To list the host, use **zone** *name* **listHosts**. To remove the host, use **zone** *name* **removeRR** *hostname* **A**.

If you want to apply an existing template while creating a zone, use the *template* attribute. For example:

```
nrcmd> zone example.com create primary ns1 hostmaster template=zone-template-1
```

Note In this example, even though you need to specify the nameserver and contact as part of the syntax, the template definition (if any) overwrites them.

To apply a template after creating the zone, use **zone** *name* **applyTemplate** *template*. For example:

```
nrcmd> zone example.com applyTemplate zone-template-1
```

Editing Primary Zones

You can edit a primary zone to modify its properties, apply a template to it, or use the zone definition to create a template from it.

Local Advanced and Regional Web UI

Step 1 Click **DNS**, then **Forward Zones** to open the List/Add Zones page (or List Forward Zones page in the regional web UI).

Step 2 Click the name of the zone to edit to open the Edit Zone page.

Step 3 Make attribute changes as necessary.

Step 4 To apply a template to the zone, choose a template name from the drop-down list at the bottom of the page, then click **Apply Template**.

Caution Be careful applying a template to a zone that is already live. Explicitly defined attributes on the template replace the existing ones defined for the zone.

Step 5 To use the zone definitions to create a template from them while modifying the zone, click **Modify Zone and Save Template**. On the Save New Zone Template page, give the template a name in the Value field, then click **Save Zone Template**. You return to the List/Add Zones page.

Confirming Zone Nameservers

Confirm your zone NS RR configuration by looking at the RRs that you created.

Local Advanced and Regional Web UI

Click the View icon () in the RRs column of the zone name on the List/Add Zones page (List Forward Zones page in the regional web UI) to open the List/Add CCM Server Protected Server RRs for Zone page (in the regional web UI) or List/Add DNS Server RRs for Zone page (in the local web UI). There should be an A record for each nameserver host in the zone. Edit these records or add more on this page.

See the "Adding Resource Records" section on page 16-2.

CLI Commands

Use **zone** *name* **listRR** to check the RRs you added.

Synchronizing Zones and Zone Commands

If a zone needs to be synchronized, the List/Add Zones page shows an icon in the Sync? column. Click this icon to open a Synchronize Zone page. Expert mode includes an additional **Sync CCM Hosts from RR Data** button. The CLI provides a **zone** *name* **syncToDns** command.

The List/Add Zones page also includes a Run icon ( ) in the Commands column. When clicked, this opens the Zone Commands page. These commands serve specific purposes:

- **Checkpoint zone**—See the "Changesets and Checkpointing" section on page 17-10.
- **Scavenge zone**—See the "Scavenging Dynamic Records" section on page 28-18.
- **Get scavenge start time**—See the "Scavenging Dynamic Records" section on page 28-18.

Importing and Exporting Zone Data

The easiest and quickest way to create a primary zone is to import an existing BIND format zone file, defined in RFC 1035. You can also export these same kinds of files to another server. BIND 4.*x.x* uses a boot file, called named.boot, to point the server to its database files. You can import your entire BIND 4.*x.x* configuration using the **import** command in the CLI. BIND 8 and BIND 9 use a configuration file, called named.config, with a different syntax.

You can import and export zone data only by using the CLI.

When a BIND file contains an $INCLUDE directive, BIND searches for the include file relative to the directory that the directory directive in the named.boot file specifies. In contrast, the **nrcmd** program searches for the include file relative to the directory containing the zone file being processed.

To avoid this problem, ensure that the BIND configuration uses absolute paths whenever specifying an include file in a zone file. If your zone files contain relative paths when specifying include files, and the directory containing the zone file is not the same as the directory that the directory directive in the named.boot file specifies, your configuration cannot load properly. You need to convert the relative paths in your zone files to absolute paths so that you can import your BIND configuration into Network Registrar. Here is an example of a configuration and how to fix paths in directory hierarchy, configuration files, and zone files:

- Directory hierarchy:

```
/etc/named.conf
/etc/named.boot
/usr/local/domain/primary/db.example
/usr/local/domain/primary/db.include
/usr/local/domain/secondary
```

- Configuration file (/etc/named.conf):

```
#BIND searches for zone files and include files relative to /usr/local/domain
option directory /usr/local/domain
#BIND finds zone file in /usr/local/domain/primary
zone example.com {
    type master ;
    file primary/db.example ;
#end of /etc/named.conf
```

- Configuration file (/etc/named.boot):

```
#BIND searches for zone files and include files relative to /usr/local/domain
directory /usr/local/domain
#BIND finds zone file in /usr/local/domain/primary
primary example.com primary/db.example
#end of /etc/named.boot
```

- Incorrect zone file (/usr/local/domain/primary/db.example):

```
#BIND searches for include file relative to /usr/local/domain
$INCLUDE primary/db.include
#end of /usr/local/domain/primary/db.example
```

  To make the configuration loadable, change the relative path ($INCLUDE primary/db.include) in the file db.example to an absolute path ($INCLUDE /usr/local/domain/primary/db.include).

Table 15-1 describes the named.boot and named.conf file directives that BIND 4 and BIND 9 support, and the corresponding Network Registrar user interface location or syntax, if any.

Table 15-1 BIND-to-CLI Command Mappings

BIND 4 Command	BIND 9 Command	Mapping to User Interface
—	acl *name* { *addr-match-list* };	Web UI: List/Add Access Control Lists page fields (see the "Access Control Lists" section on page 28-9). CLI: **acl** *name* **create** *value* **match-list**=*addr-match-list*
forwarders *addrlist*	options { forwarders { *addr*; *addr*;... }; };	Web UI: Edit DNS Server page, set Forwarders: IP Address field. CLI: **dns addForwarder** *addr*[**,***addr*...]
—	key *id* { algorithm *string*; secret *string*; };	Web UI: List/Add Encryption Keys page fields. CLI: **key** *name* **create** *secret* **algorithm**=*alg*

Table 15-1 BIND-to-CLI Command Mappings (continued)

BIND 4 Command	BIND 9 Command	Mapping to User Interface
limit transfers-in *num*	options { transfers-in *num* ;};	Web UI: Edit DNS Server page, set *xfer-client-concurrent-limit*. CLI: **session set visibility=3** **dns set xfer-client-concurrent-limit**=*number*
—	options { allow-query *addr-match-list* ;};	Web UI: Edit DNS Server page, enable *restrict-query-acl* CLI: **dns set restrict-query-acl**
options allow-recursion *addr-match-list*	options { allow-recursion *addr-match-list* ;};	Web UI: Edit DNS Server page, enable *restrict-recursion-acl*. CLI: **dns set restrict-recursion-acl**
options forward-only	options { forward only ;};	Web UI: Edit DNS Server page, enable *Slave mode*. CLI: **dns enable slave-mode**
options listen-on *port*	options { listen-on *port* {*addr-match-list*} ;};	Web UI: Edit DNS Server page, set *Listening port*. CLI: **dns set local-port-number**=*port*
options max-cache-ttl *num*	options { max-cache-ttl *num* ;};	Web UI: Edit DNS Server, set *Max. RR caching TTL*. CLI: **dns set max-cache-ttl**=*num*
options no-fetch-glue	options { fetch-glue no ;};	Web UI: Edit DNS Server page, enable *Don't fetch missing glue records*. CLI: **dns enable no-fetch-glue**
options no-recursion	options { recursion no ;};	Web UI: Edit DNS Server page, enable *Recursive queries*. CLI: **dns enable no-recurse**
options notify yes	options { notify yes ;};	Web UI: Edit DNS Server page, enable *Send zone change notification (NOTIFY)*. CLI: **dns enable notify**
options rrset-order order order ...	options { rrset-order *order* ; *order* ; ... ;};	Web UI: Edit DNS Server page, enable *Enable round-robin*. CLI: **dns enable round-robin**
options support-ixfr yes	options { request-ixfr yes ;};	Web UI: Edit DNS Server page, enable *Request incremental transfers (IXFR)*. CLI: **dns enable ixfr-enable**
options transfer-format many-answers	options { transfer-format many-answers ;};	Web UI: Edit DNS Server page, enable *Use multirec format for zone transfers*. CLI: **dns enable axfr-multirec-default**
primary *zonename file*	zone "*name*" { type master; };	Web UI: Add Zone page fields. CLI: **zone** *name* **create primary file**=*file*
secondary *zonename addr list* [*backupfile*]	zone "*name*" { type slave; };	Web UI: Add Secondary Zone page fields. CLI: **zone** *name* **create secondary** *ip-addr* [**,***ip-addr...*]
slave	zone "*name*" { type slave; };	Web UI: Edit DNS Server page, enable *Slave mode*. CLI: **dns enable slave-mode**

Table 15-1 BIND-to-CLI Command Mappings (continued)

BIND 4 Command	BIND 9 Command	Mapping to User Interface
—	zone "*name*" { allow-query { *addr*; ... }};	Web UI: Edit Zone page, set *restrict-query-acl*. CLI: **zone** *name* **set restrict-query-acl=***addr*[**,***addr*...]
tcplist *addrlist* **xfernets** *addrlist*	zone "*name*" { allow-transfer { *addr*; ... }};	Web UI: Edit Zone page, enable *restrict-xfer* and set *restrict-xfer-acl*. CLI: **zone** *name* **enable restrict-xfer** **zone** *name* **set restrict-xfer-acl=***addr*[**,***addr*...]

Zone Lists and Zone Trees

You can display forward and reverse zones as lists or as trees. The List/Add Zones page has a button with which you can toggle between these two views: **Show Forward Zone List** and **Show Forward Zone Tree**. The List/Add Reverse Zones page has similar toggle buttons: **Show Reverse Zone List** and **Show Reverse Zone Tree**.

Adding Primary Reverse Zones

For a correct DNS configuration, you must create a reverse zone for each network that you use. A reverse zone is a primary zone that DNS clients use to convert IP addresses back to hostnames, and resides in a special in-addr.arpa domain. You can create a reverse zone manually or import it from BIND. You can also create reverse zones from subnets (see the "Adding Reverse Zones from Subnets" section on page 15-13).

See Also

Adding Reverse Zones as Zones, page 15-12
Adding Reverse Zones from Subnets, page 15-13

Adding Reverse Zones as Zones

You can manually add a reverse zone as a zone.

Local Basic or Advanced and Regional Web UI

Click **Zone**, then **Reverse Zones** to open the List/Add Reverse Zones page (or List Reverse Zones page in the regional web UI). This page is almost identical to the List/Add Zones page (see Figure 5-6 on page 5-19). Then, add a reverse zone the same way you would add a forward zone, as described in the "Configuring Primary Forward Zones" section on page 15-5, except use the reverse of the forward zone network number added to the special in-addr.arpa domain as the zone name. Use the same template or SOA and nameserver values as you used for the related forward zone.

As of Network Registrar 7.0, you can enter a DHCPv4 subnet or DHCPv6 prefix value in the Name field, which converts the subnet or prefix into an appropriate reverse zone name.

CLI Commands

Use **zone** *name* **create primary** and **zone** *name* **addRR PTR** to add the primary reverse zone and pointer records for the server. You can also apply a zone template.

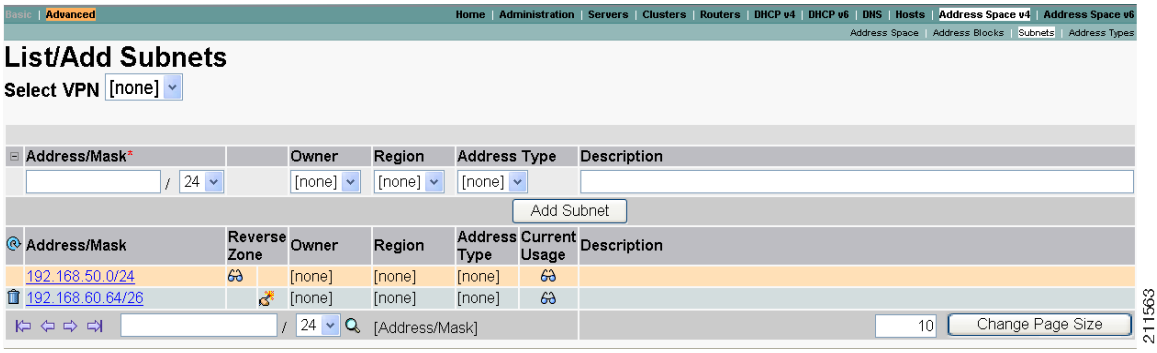
Adding Reverse Zones from Subnets

An alternative to creating reverse zones manually is to create them from existing subnets. You can do this in the web UI only.

Local Advanced and Regional Web UI

Step 1 Click **Address Space**, then **Subnets** to open the List/Add Subnets page (see Figure 15-4).

Figure 15-4 List/Add Subnets Page (Local Advanced)



Step 2 Create a subnet for the reverse zone, or use one of the existing subnets. If the subnet already has a reverse zone created from it, the Reverse Zone column shows the View icon (), which opens the List/Add Reverse Zones page. If the subnet has the Create icon () in the Reverse Zone column, click the icon to open the Create Reverse Zone(s) for Subnet page.

Step 3 On the Create Reverse Zone(s) for Subnet page, you must choose an existing zone template.

Step 4 Click **Report** to show the changesets for the creation.

Step 5 Click **Run** to run the creation.

Step 6 Click **Return** to return to the List/Add Subnets page.

Step 7 Confirm the creation by clicking **DNS**, then **Reverse Zones** to see the newly created zone on the List/Add Reverse Zones page.

Getting Zone Counts on the Server

You can view the created zones associated with the DNS server, hence obtain a count, in the web UI.

Using the CLI, you can get an exact count of the total zones for the DNS server by using **dns getZoneCount** [**forward** | **reverse** | **primary** | **secondary** | **published** | **unpublished** | **all**]. With no options specified, the command returns the total number of published zones only.

Managing Secondary Servers

When you configure a zone, choose at least one secondary server. If you have only one nameserver and it becomes unavailable, there is nothing that can look up names. A secondary server splits the load with the primary or handles the whole load if the primary is unavailable. When a secondary server starts up, it contacts the primary and pulls the zone data over. This is known as a zone transfer.

Tip If the authoritative server for your secondary zones is also running Network Registrar 6.0 or later, see the "Managing Zone Distributions" section on page 15-19 for how to avoid entering these zones manually. If you have only one secondary server, remove it geographically from the primary. They should not be on the same network segment, switch, or router, but on a different cluster entirely.

You can configure a secondary DNS server to be responsible for a secondary zone, which makes the server a secondary for that zone. You also need to give the address of the master server from which to perform zone transfers. Network Registrar must know about this master server.

See Also

Adding Secondary Forward Zones
Adding Secondary Reverse Zones, page 15-15
Enabling Zone Transfers, page 15-15

Adding Secondary Forward Zones

You can add a secondary forward zone at the local cluster.

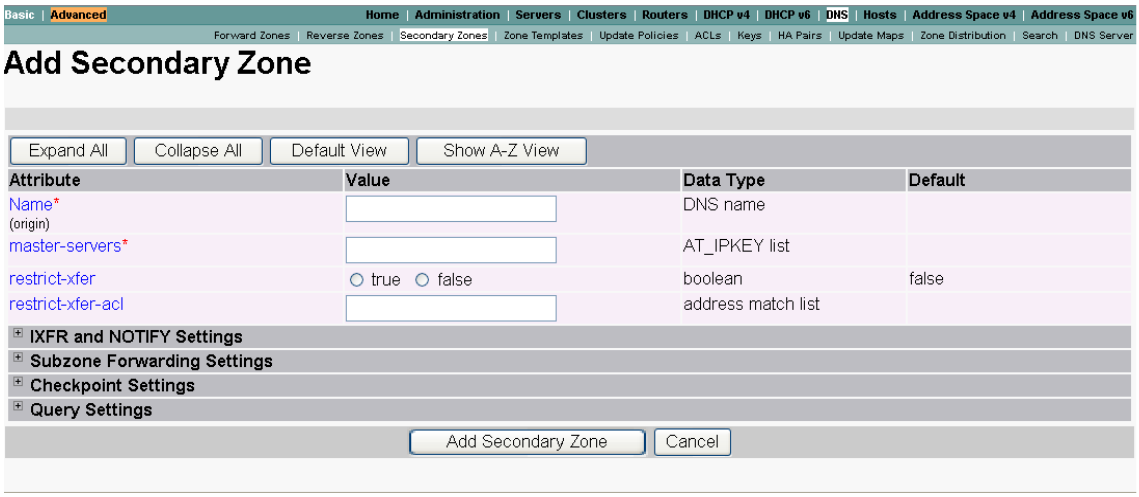
Local Basic or Advanced Web UI

Click **DNS**, then **Secondary Zones** to open the List Secondary Zones page. Then click **Add Secondary Zone** to open the Add Secondary Zone page (see Figure 15-5 on page 15-15).

A secondary zone requires a name and a list of one or more master servers. You can also enable restricting zone transfers to a set of hosts, then enter the access control list (ACL) of the restricted hosts in the restrict-xfer-acl field. Enter other attribute values as necessary, then click **Add Secondary Zone**.

Clicking the name of the secondary zone on the List Secondary Zones page opens the Edit Secondary Zone page where you can edit the secondary zone. Click **Modify Secondary Zone** on this page.

Figure 15-5 Add Secondary Zone Page (Local Advanced)



CLI Commands

Use **zone** *name* **create secondary**. The IP address you include is that of the nameserver from which data is expected, typically a primary nameserver. You cannot apply a template to a secondary zone.

Adding Secondary Reverse Zones

You should add a secondary reverse zone, just as you added a secondary forward zone.

Local Basic or Advanced Web UI

Step 1 Add the secondary reverse zone the same way you do a secondary forward zone, except that the address must be a reverse zone address. (See the "Adding Secondary Forward Zones" section on page 15-14.)

Step 2 Make the secondary zone domain name an in-addr.arpa reverse domain.

Step 3 Add the nameserver address for the secondary forward zone and set any zone transfer address restrictions, as in the "Adding Secondary Forward Zones" section on page 15-14.

Enabling Zone Transfers

A secondary server periodically contacts its master server for changes, called a zone transfer. The interval is defined in the server SOA record as the secondary refresh time. You can restrict zone transfers by setting the *restrict-xfer* attribute to true (the preset value is false) on the master server.

Note If you restrict zone transfers, the **nslookup** utility **ls** command may fail because it tries to do a full zone transfer, unless you include the IP address that **ls** runs from in the zone *restrict-xfer-acl* list.

Local Advanced and Regional Web UI

Step 1 On the List/Add Zones page (or the List Forward Zones page in the regional web UI), click the name of the primary zone to open the Edit Zone page.

Step 2 In the zone attributes area, you can set the *restrict-xfer* attribute to false (the preset value). If you set the attribute to *true*, you can also specify a list of servers to which to restrict the zone transfers by using the *restrict-xfer-acl* attribute, separating the IP addresses with commas.

Secondary zones can also restrict zone transfers from other secondary zones, so that the *restrict-xfer* and *restrict-xfer-acl* attributes are also available for secondary zone configurations.

Step 3 Click **Modify Zone**.

Step 4 You can force zone transfers for the DNS server in two ways:

- On the List Secondary Zones page, click the Run icon ( ) in the Force Zone Transfer column.
- To force all zone transfers from the primary server, on the DNS Server Commands page (see Figure 7-3 on page 7-3), click the Run icon ( ) next to Force all zone transfers.

CLI Commands

In the CLI, zone transfers are enabled by default, unless you restrict them using **zone** *name* **enable restrict-xfer**. If you want to force a zone transfer, use **zone** *name* **forceXfer secondary**.

Adding Subzones

As the zone grows, you might want to divide it into smaller pieces called subzones. You can delegate administrative authority for these subzones, and have them managed there or served by separate servers. This partitioning is called subzone delegation. Establish subzone delegation by performing these tasks:

1. Choose a subzone name.
2. Specify a nameserver name.
3. Specify a nameserver address.

See Also

Choosing Subzone Names and Servers, page 15-16
Creating and Delegating Subzones, page 15-17
Undelegating Subzones, page 15-18
Editing Subzone Delegation, page 15-18

Choosing Subzone Names and Servers

After you decide to divide the zone into subzones, you must create names for them. Involve the people responsible for the subzones in deciding their names, and try to maintain a consistent naming scheme.

These suggestions can help you avoid subzone naming problems:

- Consider not naming a subzone by its organizational name. In a changing business environment, organizations merge and are renamed. Naming a subzone after an organization could result in a name that is no longer meaningful over time.
- Consider not using geographical names that indicate the subzone location. Geographical names are meaningless to people outside your organization.
- Do not use cryptic names; make them obvious.
- Do not use existing or reserved top-level domain names as subzones. Using existing names can result in routing problems.

After you choose a subzone name, specify its nameservers, the ones the parent domain nameservers use when queried about the subzone. To ensure that the subzone is always reachable, you should specify two nameservers. They must be authoritative for this zone as either primary or secondary, or lame delegation will result (see the "Reporting Lame Delegation" section on page 17-14).

Whenever a subzone nameserver changes its name or address, the subzone administrator must inform its parent zone so that the parent zone administrator can change the subzone nameserver and *glue records*. A glue record is an A record with the address of a subzone authoritative nameserver. If the subzone administrator fails to inform its parent, the glue records are invalid. The common symptom is that a host cannot reach a host in another domain by its name, only by its address.

Note Network Registrar detects lame delegation by reporting missing subzone NS records in the parent zone, if NS record addresses do not match, and if glue A records are required.

Creating and Delegating Subzones

You delegate a subzone by creating it in the parent zone. There should be one NS record for each nameserver to which the subzone is delegated. Each NS record requires a corresponding A record describing the address of the nameserver, unless the nameserver is outside the parent zone or subzone. This A record is called a *glue* record.

See also the "Choosing Subzone Names and Servers" section on page 15-16.

Local Basic or Advanced Web UI

Step 1 Create a zone as a subdomain of the parent domain on the List/Add Zones (or List Forward Zones) page:

- If applying a zone template, go to Step 2.
- If not applying a zone template, on the Add Zone page, add the SOA records and the nameserver with its address, then click **Add Zone**.

Step 2 If Network Registrar detects a parent zone based on the subzone name, the Create Subzone in Parent Zone page appears. Click **Create as Subzone** (or **Create as Unparented Zone** if you do not want it to be a subzone) on this page.

Step 3 If you configured a nameserver in the subzone, you need to create a glue Address (A) record for it. In the field provided, enter the IP address of the nameserver, then click **Specify Glue Records**. (If there are multiple subzone nameservers, there are multiple fields for the glue records.)

Step 4 Click **Report** to show the intended changesets for the added records, then click **Run**.

Step 5 Click **Return** after viewing the actual changesets implemented.

Step 6 To confirm the added records for the subzone, click the View icon () in the RRs column for the subzone. The glue A record or records for the subzone nameserver should appear. Click **Return to Zone List**.

Step 7 To confirm the added records for the parent zone, click the View icon () in the RRs column for the parent zone. The subzone nameserver (NS) record or records plus the glue A record or records for them should appear. Click **Return to Zone List**.

CLI Commands

On the subzone primary nameserver machine, create the subdomain:

```
nrcmd> zone boston.example.com. create primary bostonDNSserv1 hostmaster
```

On the parent zone nameserver machine, add an NS record for the subzone nameserver, then Create a glue A record for the subzone nameserver:

```
nrcmd> zone example.com. addRR boston NS bostonDNSserv1.boston.example.com.
nrcmd> zone example.com. addRR bostonDNSserv1.boston.example.com. A 192.168.40.1
```

Undelegating Subzones

If you undelegate a subzone, you need to remove any associated NS and glue A records from the parent zone.



Note If you delete the subzone, Network Registrar cleans up the delegation records automatically.

Local Basic or Advanced and Regional Web UI

On the regional List/Add CCM Server Protected RRs for Zone page or local List/Add DNS Server RRs for Zone page, delete the NS record for the subzone, then delete the glue A record for the subzone server host.

CLI Commands

Use **zone** *name* **removeRR NS** and **zone** *name* **removeRR A** to remove the subzone NS and glue A records.

Editing Subzone Delegation

You can edit the subzone RRs.

Local Basic or Advanced and Regional Web UI

Step 1 On the regional List/Add CCM Server Protected RRs for Zone page or local List/Add DNS Server RRs for Zone page, edit the NS RR for the subzone by clicking the Edit icon () next to the record to open the Edit RR in Zone page.

Step 2 Edit the NS record data.

Step 3 Click **Modify Resource Record**.

Step 4 Edit the glue A RR for the subzone server in the same way as in the previous steps.

CLI Commands

Use **zone** *name* **removeRR** to delete the NS and glue A records, then use **zone** *name* **addRR** to replace them.

Enabling DNS Updates

DNS Update (RFC 2136) integrates DNS and DHCP so that they can work together. DNS update automatically records the association between the hosts and their DHCP-assigned addresses. Using DHCP and DNS update, you can configure a host automatically for network access whenever it attaches to the network. You can locate and access the host using its unique DNS hostname.

DNS update is described more fully in Chapter 28, "Configuring DNS Update." The chapter includes sections on the following:

- **Update policy (the Update Policies tab)**—Determines what kind of RRs you want updated when name-to-address associations change through DHCP.
- **Update map (the Update Maps tab)**—Defines an update relationship between a DNS server or HA DNS pair and a DHCP failover pair, DHCP policies, client-class, or access control list. (See the "Creating DNS Update Maps" section on page 28-7.)

Managing Zone Distributions

Creating a zone distribution simplifies creating multiple zones that share the same secondary zone attributes. The zone distribution requires adding one or more predefined secondary servers. Running a zone distribution synchronization adds secondary zones managed by secondary (slave) servers for each primary zone managed by a primary server. You can also use zone distributions to synchronize zone data from the CCM database to the local DNS server and regional and local cluster zone data.

The distribution must be in a star topology, that is, one primary server and multiple secondary servers. The authoritative (master) server can only be the local primary server where the zone distribution default is defined. Starting with Network Registrar 6.2, you can manage one zone distribution at the local cluster and multiple distributions at the regional clusters.

See Also

Preparing the Zone Distribution Map, page 15-20
Creating a Zone Distribution, page 15-21
Pulling Zone Distributions from Replica Data, page 15-24

Preparing the Zone Distribution Map

To prepare for creating a zone distribution, draw a zone distribution map diagram on paper.

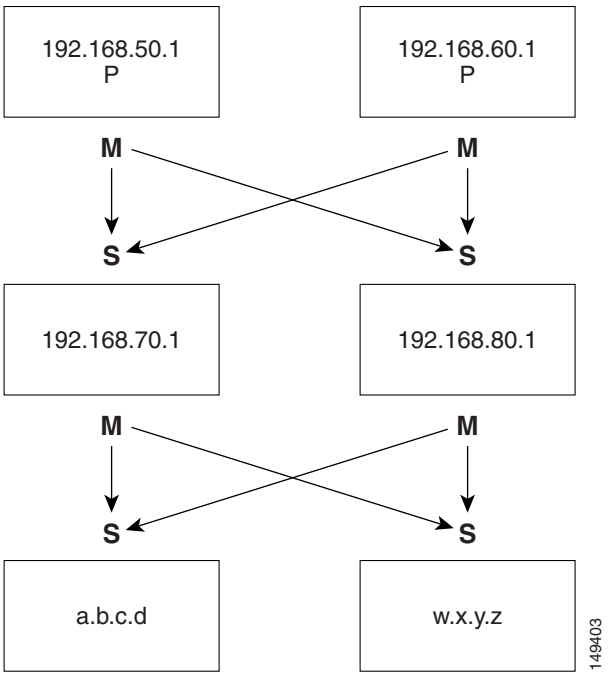
Step 1 Start by identifying the HA DNS pair that is primary (or the primary server if HA is not involved) for all the zones that you include in the map:

a. Create a box for each server in the HA DNS pair. For example, the server pair for the Chicago-cluster consists of the servers at 192.168.50.1 and 192.168.60.1.

b. Write the IP addresses of each server in each box.

c. Write a **P** (for Primary) inside each box (see Figure 15-6).

Figure 15-6 Diagramming a Zone Distribution Map



Step 2 Identify the role as master for each server by writing an **M** below the box. In the example, both primary servers are, by definition, also masters that will send copies of their zones to other servers over zone transfers. Even so, write the **M** below the box to make later steps easier.

Step 3 Identify all slave servers that will receive zone transfers directly from these masters. Below the master server boxes on the page, add a box for each slave, and write its IP address inside the box. For example, the slave servers at 192.168.70.1 and 192.168.80.1 get zone transfers from the Chicago-cluster masters.

Step 4 Write an **S** above each slave server box.

Step 5 Draw arrows from the **M** to each **S** representing the zone transfer flow (see the diagram). In this HA DNS example, the arrows go from each master to both slaves.

Step 6 As you can see from the diagram, you can extend the boxes further so that the original slaves can become masters to another set of servers (a.b.c.d and w.x.y.z).

Step 7 Enter the IP address in each box with an **M** below it in the Master Servers list when creating the zone distribution (see Figure 15-8 on page 15-22).

In the CLI, set the master-servers attribute to the list of IP addresses; for example:

```
nrcmd> zone-dist dist-1 create Chicago-cluster master-servers=192.168.50.1,192.168.60.1
```

Step 8 From the Secondary Servers drop-down list on the Add or Edit Zone Distribution Secondary Server page, choose the cluster associated with the slave server IP addresses in the boxes that have an **S** above them.

In the CLI, use **zone-dist** *name* **addSecondary** *cluster*; for example:

```
nrcmd> zone-dist dist-1 addSecondary Boston-cluster
```

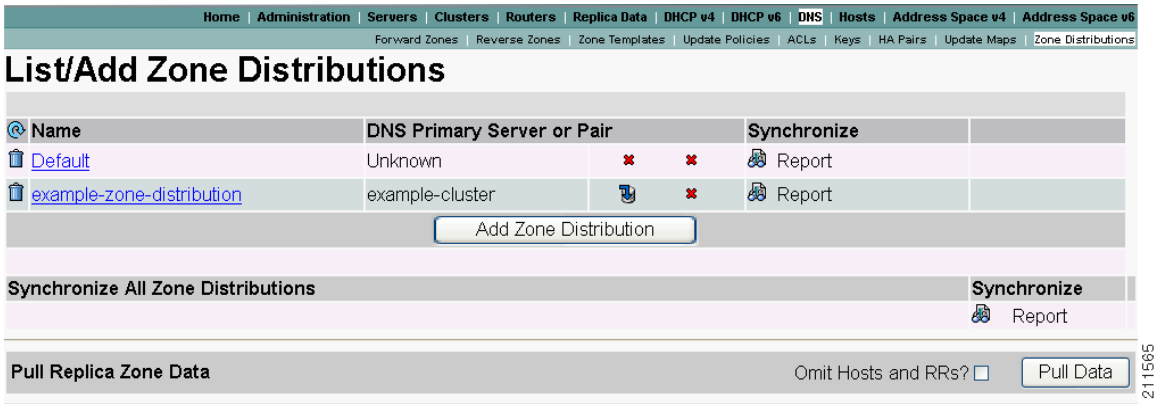
Creating a Zone Distribution

Note If you move a zone from one zone distribution to another, synchronize the first zone distribution, move the zone, then synchronize the second zone distribution.

Local Basic or Advanced and Regional Web UI

Step 1 Click **DNS**, then **Zone Distributions** (for the regional cluster) or **Zone Distribution** (for the local cluster). This opens the regional List/Add Zone Distributions page (see Figure 15-7) or the local List Zone Distributions page. Note that the default zone distribution is predefined at both clusters; however, the default cluster is the only one available at the local cluster.

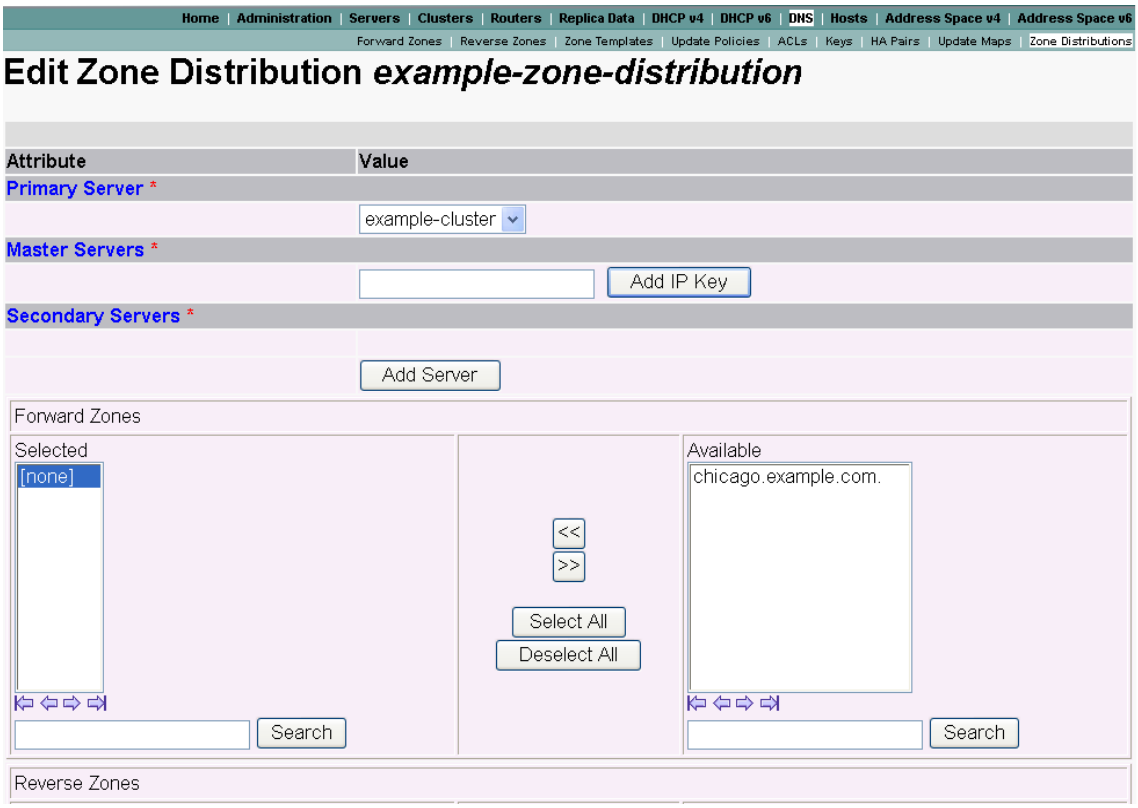
Figure 15-7 List/Add Zone Distributions Page (Regional)



Step 2 To add a new zone distribution, click **Add Zone Distribution** to open the Add Zone Distribution page. To edit an existing zone distribution, click its name to open the Edit Zone Distribution page (see Figure 15-8 on page 15-22 for a partial view). The Add Zone Distribution page and Edit Zone Distribution page are functionally equivalent.

Step 3 In the Primary Server field, enter the cluster (or configured HA DNS pair) that has the primary server. This primary server is authoritative for the zones that you will determine further down the page. This selection is subtractive: the next zone distribution you create will no longer have the cluster that you set here as one of the choices.

Figure 15-8 Edit Zone Distribution Page (Regional)



Step 4 In the Master Servers list, add the IP address (and optional key) for each master server. The master server is generally the primary server. However, you might want to set up a hierarchy of primaries and secondaries where you need to define the master servers for each of the secondary relationships. You might also want to determine the HA DNS server pairs from the master server list. You can also add an optional TSIG key (see the "Transaction Security" section on page 28-11) to the master server address by hyphenating the entry in the format *address–key*. For each entry, click **Add IP Key**.

Step 5 For a zone distribution, you need to add at least one secondary server. Click **Add Server** on the Edit Zone Distribution page to open the Add Zone Distribution Secondary Server page. Here, choose the cluster of the secondary server. Optionally, if the master servers are other than the primary servers indicated for the zone distribution, add the master server addresses, separating multiple addresses with commas. After clicking **Add Secondary Server** returns you to the Add or Edit page, you can connect to the secondary server cluster, delete it, or edit it to change the master servers.

To manage the secondary servers in the zone distribution, click the View icon (🔭) in the Manage Servers column to open the List Secondary Servers page. You can also edit the secondary server on an Edit Zone Distribution Secondary Server page.

Step 6 Choose the forward and reverse zones for the zone distribution. The default zone distribution includes all the created forward and reverse zones. For all other created zone distributions, you must move the zone or zones into the Selected column.

Step 7 Click **Add Zone Distribution** or **Modify Zone Distribution**.

Step 8 Synchronize the zone distribution with the local cluster DNS servers. A synchronization:

- Pushes staged zone, RR, or host edits to the primary server cluster or HA DNS pair for the regional cluster in Ensure, Replace, or Exact modes, or from the local cluster in Exact mode.
- Creates secondary zones for secondary servers, in Exact mode.

Step 9 Choose a synchronization mode:

- **Update**—Adds new zones, RR sets, and hosts; replaces existing hosts if there are conflicts; and creates new secondary zones.
- **Complete**—Like Ensure mode, except that it always replaces existing RR sets and hosts, and modifies the master server list on existing secondary zones.
- **Exact**—Like Complete mode, except that it deletes extra zones, RR sets, hosts, and secondary zones no longer on the primary.

Step 10 Click the Report icon () in the Synchronize column (or the same icon in the Synchronize All Zone Distributions area of the page at the regional cluster). This opens the Sync Zone Distribution page that shows a preview of the data synchronized. Click **Run**.

CLI Commands

To create the zone distribution, use **zone-dist** *name* **create** *primary-cluster*. (The primary cluster can also be the HA DNS pair.) For example:

```
nrcmd> zone-dist dist-2 create Chicago-cluster
```

To set the master server or servers, use **zone-dist** *name* **set master-servers=***addressses*, separating the addresses with commas. For example:

```
nrcmd> zone-dist zone-dist-2 set master-servers=192.168.50.1,192.168.60.1
```

To add the secondary server, use **zone-dist** *name* **addSecondary** *secondary-cluster*. For example:

```
nrcmd> zone-dist zone-dist-2 AddSecondary Boston-cluster
```

You must associate the zone distribution directly with the zone or zone template. Use **zone** *name* **set dist-map=***zone-dist-list* or **zone-template** *name* **set dist-map=***zone-dist-list*, separating the zone distribution entries with commas. For example:

```
nrcmd> zone example.com set dist-map=zone-dist-2
nrcmd> zone-template zone-template-1 set dist-map=zone-dist-2
```

To synchronize the zone distributions, use **zone-dist** *name* **sync**. You can do a synchronization in update, complete, or exact mode, and you can exclude RRs and secondary zones:

- At the local cluster, this synchronizes staged edits to the DNS server and primary zones to secondaries. Regardless of the synchronization mode, this always synchronizes the exact list of authoritative zones.
- At the regional cluster, this synchronizes primary zones with the local clusters, and primaries to secondaries. This replaces primary zones at the local cluster in Update and Complete modes, and deletes extra primary zones at the local cluster in Exact mode.
- For secondary zones, the same synchronization logic occurs at the local and regional clusters. In Update mode, this ensures that corresponding secondary zones exist on the server. In Complete mode, existing zones are updated to use the master server list specified by the zone distribution map. In Exact mode, any zones not matching the distribution map are deleted.

For example:

```
nrcmd> zone-dist zone-dist-1 sync exact no-rrs no-secondaries
```

Pulling Zone Distributions from Replica Data

You can pull zone distributions from the local replica data instead of explicitly creating them.

Tip For an example of pulling local zone data to create a zone distribution, see the "Pull Zone Data and Create a Zone Distribution" section on page 5-30.

Regional Web UI

Step 1 On the List/Add Zone Distributions page, click **Pull Replica Zone Data**.

Step 2 Choose the data synchronization mode (**Update**, **Complete**, or **Exact**) on the Select Pull Replica Zone Data page. These modes are described in the table on that page.

Step 3 Click **Report** at the bottom of the page.

Step 4 Click **Run** on the Report Pull Replica Zone Data page.

Step 5 Click **OK** on the Run Pull Replica Zone Data page.



CHAPTER 16

Managing Resource Records

This chapter explains how to configure some of the more advanced DNS zone and server parameters by using the Network Registrar web UI and CLI. Before you proceed with the concepts in this chapter, read Chapter 15, "Managing Zones," which explains how to set up the basic properties of a primary and secondary DNS server and its zones.

See Also

Managing Resource Records
Managing Hosts in Zones, page 16-11

Managing Resource Records

Resource records (RRs) comprise the data within a DNS zone. Although there is no fixed limit to the number of RRs a zone may own, in general, a zone may own one or more RRs of a given type (the zone always has a Start of Authority, or SOA, record). There are some exceptions depending on the types involved. All RRs have the entries described in Table 16-1.

Table 16-1 Resource Record Common Entries

RR Entry	Description
Name	Owner of the record, such as a zone or hostname.
Class (not required for all formats)	Network Registrar supports only the IN (Internet) class.
TTL (time to live)	Amount of time to store the record in a cache, in seconds. If you do not include a TTL, Network Registrar uses the zone default TTL, defined as a zone attribute.
Type	Type of the record, such as A (AAAA for IPv6), NS, SOA, and MX. There are many types that various RFCs define, although fewer than ten are in common use.
Record data	Data types whose format and meaning varies with record type.

See Also

Adding Resource Records
Protecting Resource Record Sets, page 16-3
Editing Resource Records, page 16-4
Removing Resource Records, page 16-4
Removing Cached Records, page 16-5
Listing Records, page 16-5
Searching Server-Wide for Records and Addresses, page 16-6
Filtering Records, page 16-8
Deleting Leftover Zone Records After Recreating Zones, page 16-9
Using Service Location (SRV) Records, page 16-9
Using NAPTR Records, page 16-9
Adding IPv6 Records, page 16-11

Adding Resource Records

Before adding or modifying RRs, keep in mind the two distinct zone edit modes that you can set and work in: staged and synchronous (see the "Staged and Synchronous Modes" section on page 15-1). For details on the RR syntax, see Appendix A, "Resource Records."

Administrator roles required for RR management are the dns-admin role at the local cluster and the central-dns-admin role at the regional cluster. The host-admin role at the local cluster and the central-host-admin role at the regional cluster can view host records only.

Local Basic or Advanced and Regional Web UI

Step 1 Click **DNS**, then **Forward Zones** to open the local List/Add Zones or regional List Forward Zones page.

Step 2 Click the View icon ( ) in the RRs column of the zone name to open the List/Add DNS Server RRs for Zone page at the local cluster (see Figure 16-1 on page 16-3). Note that by default at the regional cluster, this action opens the List/Add CCM Server Protected RRs for Zone page, because the regional cluster defaults to staged (CCM) zone edit mode.

Tip You can toggle between the List/Add CCM Server Protected RRs for Zone page and the List/Add DNS Server RRs for Zone page. Either page appears initially depending on whether you are set up with protected or unprotected RR edit capabilities (see the "Protecting Resource Record Sets" section on page 16-3).

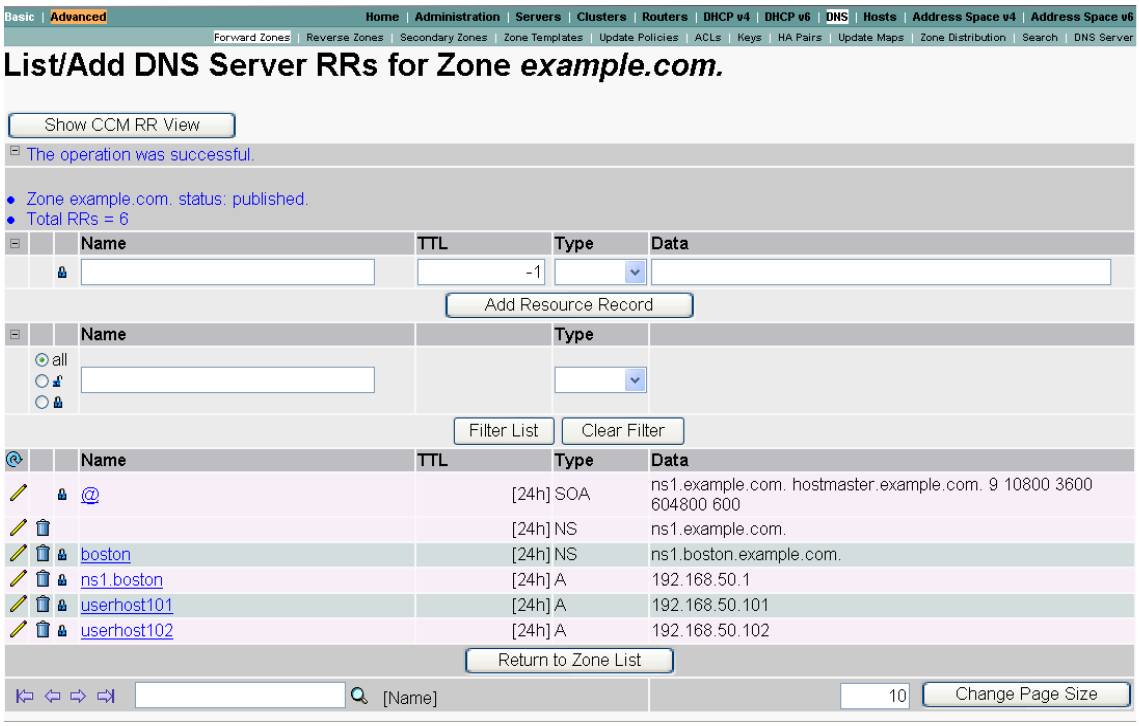
Records are listed in the formats that their respective RFCs specify, with only the first record in a set labeled with its name, and in DNSSEC order. To reduce or increase the items in the table, change the page size value at the bottom of the page, then click **Change Page Size**.

Step 3 Add the RR name, TTL (if not using the default TTL), type, and data as appropriate.

Step 4 By default, RRs are protected, which means that DNS Updates cannot overwrite them (see the "Protecting Resource Record Sets" section on page 16-3). To unprotect the RRs, click the Locked icon ( ) to the left of the record name to change it to the Unlocked icon ( ). Likewise, to protect the record, click the Unlocked ( ) icon to change it to the Locked icon ( ).

Step 5 Click **Add Resource Record**.

Figure 16-1 List/Add DNS Server RRs for Zone Page (Local Advanced)



CLI Commands

Use **zone** *name* **addRR** to add a protected RR of a certain type. You can specify the name as a relative name, if the owner is in the same domain, an absolute name (by supplying the FQDN), or the same name as the zone name (by using the at [@] symbol). You can specify the zone edit mode as part of the command by using the **–staged** or **–sync** switches.

For example:

```
nrcmd> zone example.com addRR -sync host101 A 192.168.50.101
```

Use **zone** *name* **addDNSRR** *type data* to add an unprotected RR.

Protecting Resource Record Sets

When an RR is protected, DNS Updates cannot modify the record. Most administratively created RRs are protected. However, RRs created by DNS Updates must be unprotected to allow the server to modify them. You can set this protection status for each RR set on the List/Add DNS Server RRs for Zone page.

Note that only the primary DNS server can recognize this protection status; secondary servers do not recognize the protection status of their RRs.

Local Basic or Advanced and Regional Web UI

On the local List/Add Zones or regional List Forward Zones page, click the View icon ( ) in the RRs column of the zone name. Then, on the List/Add DNS Server RRs for Zone page (see Figure 16-1 on page 16-3):

- To unprotect the RR set, click the Locked icon ( ) to the left of the RR set name to change it to the Unlocked icon ( ).
- To protect the RR set, click the Unlocked ( ) icon to change it to the Locked icon ( ).

Note that you cannot change the protection on the List/Add CCM Protected RRs for Zone page. The Locked icon ( ) always appears and you cannot change it.

CLI Commands

To protect the RR sets, use **zone** *name* **protect-name** *rrset-name*; to unprotect the zone, use the **unprotect-name** *rrset-name* action instead. For example:

```
nrcmd> zone example.com protect-name boston
100 Ok
protected boston
nrcmd> zone example.com unprotect-name boston
100 Ok
unprotected boston
```

Editing Resource Records

You can edit RRs as an individual record or as an RR set:

- **Individual RRs**—Click the Edit icon ( ) next to the record name to open the Edit RR in Zone page.
- **RR sets**—Click the name of the record to open the Edit RR Set in Zone page.

For a description of the fields to enter data, see the "Adding Resource Records" section on page 16-2.

Removing Resource Records

You can remove RRs from a zone.

Local Basic or Advanced and Regional Web UI

On the local List/Add DNS Server RRs for Zone page (see Figure 16-1 on page 16-3) or regional List/Add CCM Server Protected RRs for Zone page:

- To remove an entire record name set, click the Delete icon ( ) next to the record set name in the list, then confirm the deletion.
- To remove individual records from the set, click the name of the record set to open the Edit RR Set page, click the Delete icon next to the individual record in the list, then confirm the deletion.

CLI Commands

The CLI includes two removal commands, depending on the type of RR to remove:

- Use **zone** *name* **removeRR** to remove any RR. You must specify the owner. If you omit the data, Network Registrar removes all records of the specified type for the specified owner. Similarly, if you omit the type, Network Registrar removes all records for the specified owner.
- Use **zone** *name* **removeDNSRR** to remove unprotected RRs only.

Removing Cached Records

Removing cached records removes nonauthoritative RRs from both in-memory and persistent (nonauthoritative) cache. The DNS server must be running to remove cached records. Changes take effect immediately.

Local Basic or Advanced Web UI

Step 1 Click **DNS**, then **DNS Server** to open the Manage DNS Server page.

Step 2 Click the Run icon ( ) in the Commands column to open the DNS Server Commands page.

Step 3 For the Remove nonauthoritative RR set command, if you want to:

- Remove the entire cached RR set, enter just the name of the RR set; omit the type and data values.
- Remove the cached RR name, enter the name and type of RR.
- Remove the specific cached record, enter the name, type, and data.

Step 4 Click the Run icon ( ) for that command. You should get a confirmation message.

Step 5 Click **Return**.

CLI Commands

Use **dns removecachedRR** *name type data* to remove cached RRs in the memory and persistent caches. With the type and data omitted, this removes the entire RR set; if the type is included without the data, this removes the name set; with the name, type, and data included, this removes the specific record only.

Listing Records

You can display all the RRs, or only the staged or synchronized ones. The server must be operating to display the synchronized records.

Local Basic or Advanced and Regional Web UI

On the regional List/Add CCM Server Protected RRs for Zone page or local List/Add DNS Server RRs for Zone page (see Figure 16-1 on page 16-3), view the records on the page, then click **Return to Zone List**.

CLI Commands

Use **zone** *name* **listRR** to display RRs in the named zone. You can also specify whether you want all records or only staged (CCM) or synchronized (DNS) records (see the "Filtering Records" section on page 16-8 for details). For example:

```
nrcmd> zone example.com listRR dns
```

You can get an exact count of the total RRs for the DNS server by using **dns getRRCount** [**zone** *name* **| forward | reverse | primary | secondary | published | unpublished | all**]. Options let you request the RR count for a single zone or all zones of a given type.

Searching Server-Wide for Records and Addresses

Network Registrar 7.0 provides a new function with which you can search for RRs and IP addresses server-wide. The search is a filter mechanism whereby you can specify a combination of RR and address attributes to target one or more RRs or addresses configured for the network. The search function is available at the local cluster only.

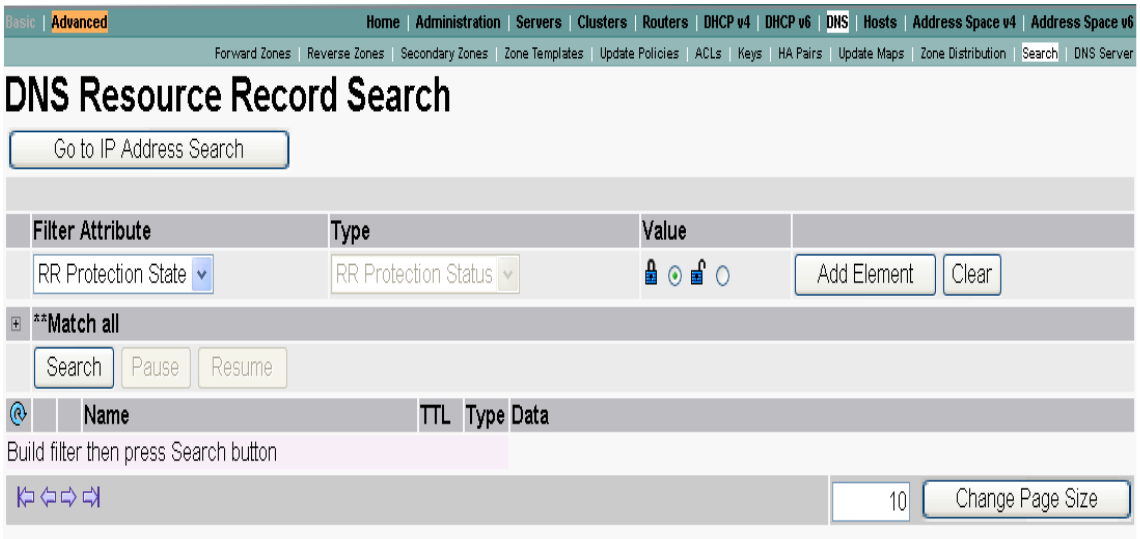
You can search RRs by:

- IP address
- Protection state
- Name prefix
- Type
- Zone

Local Advanced Web UI

Step 1 Click **DNS**, then **Search** to open the DNS Search page (see Figure 16-2).

Figure 16-2 DNS Resource Record Search Page (Local Advanced)



Step 2 Choose a filter attribute from the drop-down list, such as RR Protection State, or search by IP address.

To search by IP address, click **Go to IP Address Search** to open the IP Address Search page. Enter an IP address, then click **Search**.

Note In an IP address search, the DNS server does not search all forward zones for RRs that have the specified address in the data field. Instead, the server looks up the matching PTR record in the reverse zone and returns all the respective RRs in the forward zone.

Step 3 If you are not searching by IP address, choose a filter type from the drop-down list depending on the filter attribute you chose:

- **RR Protection State**—RR Protection Status, either locked or unlocked.
- **RR Name Prefix**—RR Name Prefix.
- **RR Type**—RR Type.
- **Zone**—
  - **Zone List**—Comma-separated list of DNS zones. The search returns the RRs for all the zones listed (if a zone does not exist, it ignores that zone).
  - **Regular Expression**—Value is a Regular Expression in regex syntax. (For common regex usage, see Table 5-4 on page 5-22).
  - **Zone Flags**—Forward or Reverse, Primary or Secondary, or Published or Unpublished. Click a check mark for one of these pairs, then click the radio button for one of the choices in the pair.

Step 4 Enter or select a Value, based on the Type selected. To clear the filter, click **Clear Filter**.

Step 5 Click **Add Element** to add the search element to the filter elements list. The Filter Elements heading changes to identify the filter attribute and value used for the filter. If you add more than one element, the heading identifies the ANDed values of the elements. For example, if you add an element for a name prefix search for user, then add another element for an RR type search for A records, the filter element heading will identify the search as ****RR Name Prefix = user AND RR Type = A**.

Step 6 You can add as many elements as you like (remembering that the search results are an intersection of the filter elements). View the filter elements list by clicking the plus sign (**+**).

Step 7 Click **Search**.

Step 8 Check the table of resulting RRs from the search, which shows for each RR its zone, hostname, TTL, type, and associated data. If necessary, change the page size to see more entries at one time (you might still need to page forward and back). The RRs are sorted in DNSSEC order.

Tip If the search results are less than expected due to the ANDing of the filter elements, look at the filter list for any element that might be compromising the search, delete it by clicking the Delete icon ( ) next to it, then redo the search.

CLI Commands

Use **dns findRR** to find RRs across the zones. The command syntax is of two kinds:

```
nrcmd> dns findRR -name {fqdn | domainaddr}
nrcmd> dns findRR [-namePrefix nameprefix]
    [-rrTypes RRtypelist]
    [-protected | -unprotected]
    [-zoneType {forward | reverse | primary |secondary | published | unpublished | ALL}]
```

You can search by domain or its address, or enter the beginning characters of the RR name (the name prefix). If you search by RR name prefix, you can narrow the search by a list of RR types, protection status, or zone type. The output clearly indicates the zone for each found entry. For example:

```
nrcmd> dns findRR -namePrefix user -rrTypes A
userhost101.example.com IN A 192.168.50.101
userhost102.example.com IN A 192.169.50.102
userhost103.boston.example.com IN A 192.168.50.103
```

Use **zone findRR** to search on RR name prefixes, RR types, or protection status:

```
nrcmd> zone findRR [-namePrefix nameprefix]
    [-rrTypes RRtypelist]
    [-protected | -unprotected]
```

Filtering Records

You might want to filter records to display only one type of record, such as an A (or IPv6 AAAA) or PTR record. (See also the "Searching Server-Wide for Records and Addresses" section on page 16-6.)

Local Basic or Advanced and Regional Web UI

You can filter RRs right from the regional List/Add CCM Server Protected RRs for Zone page or local List/Add DNS Server RRs for Zone page (see Figure 16-1 on page 16-3). Look for the Name and Type fields just below the **Add Resource Record** button.

By default, RRs are sorted alphabetically by name, starting with the top-of-zone records (marked with the @ symbol), and secondarily sorted by type, then data. You can also sort them by:

- **Protected state**—You can click All, Unprotected (), or Protected ().
- **Name prefix**—Starting characters in the name. Note that the * character is not a wildcard. For example, entering **al** returns alberta, allen.wrench, and allie, whereas entering **al*** returns al* and al*ert.
- **RR type**—Click one of the RR types in the drop-down list, such as A (or IPv6 AAAA) or TXT.

When the selection is complete, click **Filter List**. This returns just the filtered entries in the table below the fields. To return to the full, unfiltered list, click **Clear Filter**.

CLI Commands

Use **zone** *name* **listRR** *option* to filter records. This helps determine whether DNS Update is working and what dynamic entries are in the system. The options are:

- **all**—Displays all records (the default if omitted).
- **ccm**—Displays the CCM protected RRs only (the default for the local cluster).
- **dns**—Displays the DNS live RRs only (the default for the regional cluster).

Deleting Leftover Zone Records After Recreating Zones

You can delete leftover static zone records after you delete a zone and then recreate it. Dynamic RRs are automatically deleted when you recreate the zone. This function is currently not available in the web UI.

Use **zone** *name* **cleanRR** if you periodically delete and reimport zones, which can cause your database to grow. This command uses the DNS server historical zone data to determine what part to remove. It does not print a list of records to be deleted or prompt you for confirmation. You can safely run it any time.

Using Service Location (SRV) Records

Windows 2000 domain controllers use the service location (SRV) RR to advertise services to the network. This RR is defined in the RFC 2782, "A DNS RR for specifying the location of services (DNS SRV)." The RFC defines the format of the SRV record (DNS type code 33) as:

```
_service._protocol.name ttl class SRV priority weight port target
```

There should always be an A record associated with the SRV record target so that the client can resolve the service back to a host. In the Microsoft Windows 2000 implementation of SRV records, the records might look like this:

```
myserver.example.com A 10.100.200.11
_ldap._tcp.example.com SRV 0  0  389  myserver.example.com
_kdc._tcp.example.com SRV 0  0  88  myserver.example.com
_ldap._tcp.dc._msdcs.example.com SRV 0  0  88  myserver.example.com
```

An underscore (_) always precedes the service and protocol names. In the example, _kdc is the Key Distribution Center. The priority and weight help a client choose between target servers providing the same service (the weight differentiating those with equal priorities). If the priority and weight are all set to zero, the client orders the servers randomly. For more information on SRV records, see Appendix A, "Resource Records."

Note For a description of how Windows 2003 and XP clients interoperate with DNS and DHCP servers, including scavenging dynamic RRs, see the "Configuring DNS Update for Windows Clients" section on page 28-20.

Using NAPTR Records

Network Registrar supports Naming Authority Pointer (NAPTR) RRs. These records help with name resolution in a particular namespace and are processed to get to a resolution service. Because NAPTR records are a proposed standard, RFC 3403, Network Registrar only validates their numeric record fields. However, the proposed standard requires a value for each field, even if it is null (""), and there are no preset values. See Appendix A, "Resource Records," for the syntax of NAPTR records.

When using a NAPTR record to locate a Session Initiation Protocol (SIP) proxy, see the proposed standard, RFC 2916 or RFC 3263. In RFC 2916, the ENUM working group of the Internet Engineering Task Force specifies NAPTR records to map E.164 addresses to Universal Resource Identifiers (URIs). Using the NAPTR record resolves a name in the E.164 international public telecommunication namespace to a URI, instead of providing the name of a service to use as a resolver. The U flag was added to the NAPTR record for this purpose.

For example, to specify a SIP proxy for the phone number +4689761234, add a NAPTR record at the name 4.3.2.1.6.7.9.8.6.4.e164.arpa. with this content:

```
100 10 "u" "sip+E2U" "/^.*$/sip:info@tele2.se/" .
```

This sets these fields of the NAPTR record:

```
order = 100
preference = 10
flags = "u"
service = "sip+E2U"
regexp = "/^.*$/sip:info@tele2.se/"
replacement = .
```

After you configure these fields, the DNS client dealing with phone number +4689761234 can now find an SIP service URI by replacing the number with sip:info@tele2.se. The E.164 zone mostly uses the NAPTR record for wholesale replacement of the input telephone number. Section 3.2.3 of RFC 2916 includes an example of one transformation to a Lightweight Directory Access Protocol (LDAP) query that preserves some of the digits. The E.164 zone does not map to service location (SRV) records because it wants to obtain a SIP URL that is more humanly readable to the left of the at (@) symbol.

Local Basic or Advanced and Regional Web UI

Step 1 On the List/Add Zones page, click the View icon () in the RRs column.

Step 2 Enter the owner of the record in the Name field.

Step 3 Enter the TTL (if necessary).

Step 4 Click NAPTR in the Type drop-down list.

Step 5 Enter the data as a string embedded in quotes and separated by spaces:

a. Order

b. Preference

c. Flags

d. Service

e. Regular expression

f. Replacement string

For example:

```
"100 10 u sip+E2U /^.*$/sip:info@tele2.se/ ."
```

Step 6 Click **Add Resource Record**.

Step 7 Refresh the list if necessary.

CLI Commands

Use **zone** *name* **addRR**.

Adding IPv6 Records

The IPv6 address space includes some additional RRs. For details, see Chapter 26, "Managing DHCPv6 Addresses," and the RR descriptions in Appendix A, "Resource Records."

Managing Hosts in Zones

You can manage the RRs for a host by configuring the host record rather than the individual RRs. When you define a host, the DNS server automatically creates an Address (A) RR for IPv4, or an AAAA RR for IPv6, for it. If the reverse zone for the host exists, the server can also create the associated Pointer (PTR) RR for it.

See Chapter 10, "Managing Hosts," for details



CHAPTER 17

Managing DNS Server Properties

This chapter explains how to set the DNS server parameters. Before you proceed with the tasks in this chapter, read Chapter 15, "Managing Zones," which explains how to set up the basic properties of a primary and secondary zone.

See Also

Managing DNS Servers
Setting DNS Server Properties, page 17-2
Setting Advanced DNS Server Properties, page 17-11
Tuning DNS Properties, page 17-17
Troubleshooting DNS Servers, page 17-18

Managing DNS Servers

You can manage the DNS server, including viewing its health, statistics, and logs; starting, stopping, and reloading it; running certain commands (see the "Running DNS Commands" section); and editing the server attributes.

To view the server status and health, or stop, start, and reload the server, in the local cluster web UI, click **DNS**, then **DNS Server** to open the Manage DNS Server page.

See Also

Running DNS Commands, page 17-1
Configuring DNS Server Network Interfaces, page 17-2

Running DNS Commands

Access the commands by using the Run icon ( ) in the Commands column to open the DNS Server Commands page in the local web UI. Each command has its own Run icon (click it, then **Return** when finished):

- **Flush cache**—See the "Flushing DNS Cache" section on page 17-15.
- **Force all zone transfers**—See the "Enabling Zone Transfers" section on page 15-15.
- **Scavenge all zones**—See the "Scavenging Dynamic Records" section on page 28-18.
- **Remove non-authoritative RR set**—See the "Removing Cached Records" section on page 16-5.

Note If you find a server error, investigate the server log file for a configuration error, correct the error, return to this page, and refresh the page.

Configuring DNS Server Network Interfaces

You can configure the network interfaces for the DNS server from the Manage Servers page in the local web UI. To get there, click **Servers**, then **Manage Servers**. Click the Interfaces icon ( ) for the DNS server to open the Manage DNS Server Network Interfaces page.

This page shows the available network interfaces that you can configure for the server. By default, the server uses all of them. To configure an interface, click the Configure icon ( ) in the Configure column for the interface. This adds the interface to the Configured Interfaces table, where you can edit or delete it. Clicking the name of the configured interface opens the Edit DNS Server Network Interface page, where you can change the address and port of the interface. Click **Modify Interface** when you are done editing, then click **Return** to return to the Manage Servers page.

Note The IPv6 functionality in DNS requires IPv4 interfaces to be configured except if the DNS server is isolated and standalone (it is its own root and is authoritative for all queries).

Setting DNS Server Properties

You can set properties for the DNS server itself, along with those you already set for its zones. These include:

- **General server properties**—See the "Setting General DNS Server Properties" section on page 17-3.
- **Forwarders**—See the "Defining Forwarders for DNS Servers" section on page 17-3.
- **Subzone forwarding**—See the "Setting Subzone Forwarding" section on page 17-4.
- **Resolution exception**—See the "Using Resolution Exception" section on page 17-4.
- **Caching-only servers**—See the "Configuring Caching-Only DNS Servers" section on page 17-6.
- **Delegation-only zones**—See the "Specifying Delegation-Only Zones" section on page 17-7.
- **Root name servers**—See the "Defining Root Nameservers" section on page 17-7.
- **Recursive queries**—See the "Enabling Recursive Queries" section on page 17-8.
- **Round-robin server processing**—See the "Enabling Round-Robin" section on page 17-8.
- **Subnet sorting**—See the "Enabling Subnet Sorting" section on page 17-9.
- **Enabling incremental zone transfers**—See the "Enabling Incremental Zone Transfers (IXFR)" section on page 17-9.
- **Changesets and checkpointing**—See the "Changesets and Checkpointing" section on page 17-10.
- **Restricting zone queries**—See the "Restricting Zone Queries" section on page 17-10.
- **Enabling NOTIFY packets**—See the "Enabling NOTIFY" section on page 17-11.

Setting General DNS Server Properties

You can display DNS general server properties, such as the name of the server cluster or host machine and the version number of the Network Registrar DNS server software. You can change the internal name of the DNS server by deleting the current name and entering a new one. This name is used for notation and does not reflect the official name of the server. Network Registrar uses the server IP address for official name lookups and for DNS updates (see Chapter 28, "Configuring DNS Update").

The following subsections describe some of the more common property settings. They are listed in the "Setting DNS Server Properties" section.

Local Basic or Advanced Web UI

To access the server properties, click **DNS**, then **DNS Server** to open the Manage DNS Server page. Click the name of the server to open the Edit DNS Server page. (Or, click **Servers**, then **Manage Servers**, then the **Local DNS Server** link to get to the same page.) The page displays all the DNS server attributes. Then click **Modify Server** to save the DNS server attribute modifications.

CLI Commands

Use **dns** [**show**] to display the DNS server properties.

Defining Forwarders for DNS Servers

Sites that must limit their network traffic for security reasons can designate one or more servers to be forwarders that handle all off-site requests before the local server goes out to the Internet. Over time, the forwarders build up a rich data cache that can satisfy most requests.

Forwarders are useful in that they:

- **Reduce the load on the Internet connection**—Forwarders build up a cache and thus reduce the number of requests sent to external nameservers and improve DNS performance.
- **Improve the DNS response to repeated queries**—The forwarder cache can answer most queries.
- **Handle firewalls**—Hosts that do not have access to root nameservers can send requests to the forwarder that does.

Tip Restrict the nameserver even more by stopping it from attempting to contact an off-site server. This kind of server uses forwarders exclusively. It answers queries from its authoritative and cached data, but it relies completely on the forwarders for data not in its cache. If the forwarder does not provide an answer, then the slave mode server does not try to contact other servers.

You can have multiple forwarders configured. If the first forwarder does not respond after a preset value of eight seconds, Network Registrar asks each remaining forwarder in sequence until one answers, or until it gets to the end of the list. If the DNS server does not get an answer, the next step depends on whether you have slave mode on or off:

- If slave mode is on, the DNS server stops searching and responds that it cannot find the answer.
- If slave mode is off, the DNS server sends the query to the domain designated nameservers as if there were no forwarders listed.

You can modify the eight-second preset value for the forwarding retry interval through the DNS *forward-retry-time* attribute. Because these queries are recursive and can require more time for the forwarder to resolve, set this interval to the *request-expiration-time* (90 seconds by default) divided by one fewer than the total number of configured forwarders or resolution exception servers. (See also the "Enabling Recursive Queries" section on page 17-8.)

Note Subzone forwarding (see the "Setting Subzone Forwarding" section on page 17-4) and resolution exception (see the "Using Resolution Exception" section on page 17-4) override any forwarders you set.

Local Basic or Advanced Web UI

On the Edit DNS Server page, in Group view under the Forwarders category, enter the IP addresses of the forwarding servers, click **Add Forwarder**, click whether you want no-recurse or slave-mode enabled or disabled, then click **Modify Server**.

CLI Commands

To:

- Specify the address (or space-separated addresses) of nameservers to use as forwarders, use **dns addForwarder**.
- Designate the server as a slave, use **dns enable slave-mode**.
- List the current forwarders, use **dns listForwarders**.
- Edit your forwarder list, you must remove any offending forwarder and reenter it.
- Remove a forwarder or list of forwarders, use **dns removeForwarder**.

Setting Subzone Forwarding

For zones with forwarders set (see the "Defining Forwarders for DNS Servers" section on page 17-3), the normal Network Registrar behavior is to ignore delegation to subzone nameservers and forward queries to these forwarding servers. You would normally need to set a resolution exception (see the "Using Resolution Exception" section) to the subzone server. This might be impractical for large numbers of subzones. With the zone *subzone-forward* attribute set to **no-forward**, when the server receives a query for any of its subzones, it tries to find relevant subzone NS records, resolve their corresponding IP addresses, and delegate the query to those IP addresses. The default is to use the forwarders, if requested. However, subzone forwarding overrides any requested forwarders.

For example, with the example.com zone and its subzones boston.example.com and chicago.example.com, if you set the no-forward option, queries do not go to the forwarder, but their NS and corresponding A records are found. The preset value is *subzone-forward*=normal, which causes queries to be forwarded.

Using Resolution Exception

If you do not want the DNS servers to use the standard resolution method to query the root nameserver for certain names outside its domain, use resolution exception. This bypasses the root nameservers and targets a specific server (or list of servers) to handle name resolution.

For example, example.com has four subsidiaries: Red, Blue, Yellow, and Green. Each has its own domain under the .com domain. When users at Red want to access resources at Blue, their DNS server knows that it is not authoritative for Blue and appeals to the root nameservers. These queries cause unnecessary traffic, and in some cases fail because internal resources are often barred from external queries or sites that use unreachable private networks without unique addresses.

Resolution exception solves these problems. The Red administrator lists all the other example.com domains that users might want to reach and at least one corresponding nameserver. When a Red user wants to reach a Blue server, the Red server asks the Blue server instead of querying the root.

In previous releases of Network Registrar, the DNS server forwarded queries to the specified resolution exception server or servers only if there was no Name Server (NS) resource record (RR) for the requested domain in its cache. If there was an NS address, it would always try to forward to that nameserver. Network Registrar 7.0 removes this limitation and provides more flexibility in resolution exception forwarding. You can now set a DNS server attribute, *exception-forwarding*, with one of three values:

- **forward-always**—DNS server always forwards queries to the specified exception server or servers.
- **forward-first**—DNS server tries to forward queries to any exception server or servers, then to any NS nameservers.
- **forward-last** (preset value)—DNS server tries to forward queries to the NS nameservers, then to any resolution server or servers.



Tip With no resolution exception defined, when global forwarding is set (see the "Setting Subzone Forwarding" section on page 17-4), any query for the subzone delegation goes to the forwarder, and not to the server that is authoritative for that subzone. However, if you set the *subzone-forward* attribute to *no-forward* for a zone, the zone server is set as the implicit resolution exception server for all the subzones, and any forwarding and explicitly set resolution exceptions are ignored for that zone. Even though defining a resolution exception at a subzone name is a useful way to prevent queries to a specific subzone from being forwarded to the global forwarder, to do this for all subzones of a zone, use subzone forwarding.

The difference in server behavior with resolution exception and nameserver forwarding is:

- With resolution exception forwarding set to a list of nameservers:
  - Recursion applies.
  - The DNS server forwards only once per query and always in the list order.
  - The server tries to reach each exception server within the server *forward-retry-time*.
- With nameserver forwarding set to a list of nameservers:
  - Recursion does not apply (because we expect the nameserver to answer the query).
  - The DNS server retries each address in the list up to three times in a round-robin fashion.
  - The server uses previous round-trip times, or the server *request-retry-time* and *request-expiration-time*, to calculate how often to retry. The most likely situation is an exponential backoff from the *request-retry-time*. The server also uses an exponential backoff each time it tries the nameserver until the retry is more than half the *request-expiration-time*.

In both cases, the query times out if there is no answer within the *request-expiration-time*. The server drops the request (for a UDP request) or returns a SERVFAIL error (for a TCP request).

Local Basic or Advanced Web UI

On the Edit DNS Server page, in Group view under the Resolution Exceptions category:

- To specify the resolution exception servers, enter the domain name and IP address (or addresses) of each nameserver in the Name and IP Address(es) fields. Click **Add Exception** for each one.
- To guarantee that the server forwards queries to the resolution exception servers instead of (or before) forwarding to the nameserver specified by the NS RR, choose **forward-always** or **forward-first** for the *exception-forwarding* attribute. Otherwise, choose **forward-last** (the preset value).

Click **Modify Server**.

To delete an exception list, click the Delete icon () next to the domain name and IP address or addresses of the nameserver you want to remove, then click **Modify Server**.

CLI Commands

To:

- Add the exception domains and servers, separated by spaces, use **dns listExceptions** and **dns addException**. Use this command only if you do not want your DNS server to use the standard name resolution for names outside the local authoritative zone.
- Guarantee that the server forwards queries to the resolution exception server list instead of (or before) forwarding to the nameserver specified by the NS RR, use **dns set exception-forwarding=forward-always** or **dns set exception-forwarding=forward-first**. Otherwise, use **dns set exception-forwarding=forward-last** (the preset value).
- Remove a resolution exception, use **dns removeException**.
- Replace a resolution exception, use **dns addException** with the new values. You must also flush the cache so that the server does not refer to the old resolution values in cache.

Configuring Caching-Only DNS Servers

By definition, all servers are caching servers, because they save the data that they receive until it expires. However, you can create a caching-only server that is not authoritative for any zone. The only function of this type of server is to answer queries by storing in its memory data from authoritative servers. The caching-only server can then learn or cache the data to answer subsequent queries. Caching query responses considerably reduces the overhead (and latency) of subsequent queries to these cached records

When you first install Network Registrar, the DNS server automatically becomes a non-authoritative, caching-only server until you configure zones for it. If you keep the DNS server as a caching-only server, you must have another primary or secondary DNS server somewhere that is authoritative and to which the caching-only server can refer. The caching-only server should never be listed as an authoritative server for a zone.

You must set up a caching-only server to respond to recursive queries, where a server keeps trying to get to an authoritative server so that it can update its cache with the address resolution data. Network Registrar servers are recursive by default.

To tune the performance of a caching-only server:

- Increase the in-memory cache to the maximum possible value that is aligned to the operating system physical memory capacity by setting the *mem-cache-size* value. The preset value is 10000, but it is recommended to increase this value, for example, to 30000 or 100000.
- When using a large cache, you might want to disable persisting the cache, so that reload time is not impacting the need to save the large cache. Disable the *persist-mem-cache* attribute. (However, note that disabling the attribute discards the cache at each server reload.)

  In cases where you do not want to disable *persist-mem-cache* (such as when the DNS server requires frequent reloads or restarts), there can still be a performance benefit from not writing cached entries with small TTLs. These entries are likely to expire before the next time the server needs them. Therefore, it is recommended not to write cached entries with TTLs of less than 5 seconds. To do this, enter Expert mode in the web UI or use **session set visibility=3** in the CLI, then set the *cache-write-ttl-threshold* attribute to 5s.
- Restrict queries to certain clients only by setting the *restrict-query-acl* attribute.

Reload the DNS server after changing any of these values.

Local Basic or Advanced Web UI

On the Edit DNS Server page, click **Show A-Z View**, then find each previously described attribute to set it. Reload the server after the changes.

CLI Commands

Use commands similar to the following:

```
nrcmd> dns set me-cache-size=30000
nrcmd> dns disable persist-mem-cache
nrcmd> session set visibility=3
nrcmd> dns set cache-write-ttl-threshold=5s
nrcmd> session set visibility=5
nrcmd> dns set restrict-query-acl=myaccesslist
nrcmd> dns reload
```

Specifying Delegation-Only Zones

You can instruct the server to expect only referrals when querying the specified zone. In other words, you want the zone to contain only NS records, such as for subzone delegation, along with the apex SOA record of the zone. This can filter out “wildcard” or “synthesized” data from authoritative nameservers whose undelegated (in-zone) data is of no interest. Enable the DNS server *delegation-only-domains* attribute for this purpose.

Defining Root Nameservers

Root nameservers know the addresses of the authoritative nameservers for all the top-level domains. When you first start a newly installed Network Registrar DNS server, it uses a set of preconfigured root servers, called root hints, as authorities to ask for the current root nameservers.

When Network Registrar gets a response to a root server query, it caches it and refers to the root hint list. When the cache expires, the server repeats the process. Because Network Registrar has a persistent cache, it does not need to requery this data when it restarts. The time to live (TTL) on the official root server records is currently six days, so Network Registrar requeries every six days, unless you specify a lower maximum cache TTL value (see the "Setting Maximum Cache TTLs" section on page 17-15).

Because the configured servers are only hints, they do not need to be a complete set. You should periodically (every month to six months) look up the root servers to see if the information needs to be altered or augmented.

Local Basic or Advanced Web UI

On the Edit DNS Server page, in Group view under the Root Nameservers category, enter the domain name and IP address of each additional root nameserver, clicking **Add Root Namerserver** after each one, then click **Modify Server**.

CLI Commands

Use **dns addRootHint**.

Enabling Recursive Queries

There are two types of queries—*recursive* and *iterative* (nonrecursive). DNS clients typically generate recursive queries, where the nameserver asks other DNS servers for any non-authoritative data not in its own cache. With an iterative query, the nameserver answers the query if it is authoritative for the zone, has the answer in its cache, or tells the client which nameserver to ask next. You often want to make a root server iterative instead of recursive. Recursion is like saying, "Let me talk to Bob and get back to you." Iteration is like saying, "Let me direct you to Bob for the information."

You can also restrict the clients from which to honor recursive queries by setting the *restrict-recursion-acl* attribute to an access control list (ACL); see the "Restricting Zone Queries" section on page 17-10.

Here is a list of other attributes that you can set to optimize recursive query performance, although you would generally leave these values to their defaults:

- ***forward-retry-time***—Retry interval for forwarding queries to forwarders or resolution exception servers (preset value 8 seconds); see the "Defining Forwarders for DNS Servers" section on page 17-3.
- ***request-expiration-time***—Expiration time of general DNS queries (preset value 90 seconds).
- ***request-retry-time***—Retry time interval for general DNS queries (preset value 4 seconds).
- ***tcp-query-retry-time***—Retry time for DNS queries over TCP, in response to truncated UDP packets (preset value 10 seconds).

Enabling Round-Robin

A query might return multiple A records for a nameserver. To compensate for most DNS clients starting with, and limiting their use to, the first record in the list, you can enable *round-robin* to share the load. This method ensures that successive clients resolving the same name will connect to different addresses on a revolving basis. The DNS server then rearranges the order of the records each time it is queried. It is a method of load sharing, rather than load balancing, which is based on the actual load on the server.



Tip Adjust the switchover rate from one round-robin server to another using the TTL property of the server A record.

Local Basic or Advanced Web UI

On the Edit DNS Server page, in A-Z view, find the *round-robin* attribute, set it to enabled, then click **Modify Server**.

CLI Commands

Use **dns get round-robin** to see if round-robin is enabled (it is by default). If not, use **dns enable round-robin**.

Enabling Subnet Sorting

If you enable subnet sorting, as implemented in BIND 4.9.7, the Network Registrar DNS server confirms the client network address before responding to a query. If the client, server, and target of the query are on the same subnet, and the target has multiple A records, the server tries to reorder the A records in the response by putting the closest address of the target first in the response packet. DNS servers always return all the addresses of a target, but most clients use the first address and ignore the others.

If the client, DNS server, and target of the query are on the same subnet, Network Registrar first applies round-robin sorting and then applies subnet sorting. The result is that if you have a local response, it remains at the top of the list, and if you have multiple local A records, the server cycles through them.

Local Basic or Advanced Web UI

On the Edit DNS Server page, in A-Z view, find the *subnet-sorting* attribute, set it to enabled, then click **Modify Server**.

CLI Commands

Use **dns enable subnet-sorting** or **dns disable subnet-sorting** (the preset value).

Enabling Incremental Zone Transfers (IXFR)

Incremental Zone Transfer (IXFR, described in RFC 1995) allows only changed data to transfer between servers, which is especially useful in dynamic environments. IXFR works together with NOTIFY (see the "Enabling NOTIFY" section on page 17-11) to ensure more efficient zone updates.

Primary zone servers always provide IXFR. You should explicitly enable IXFR on the server (you cannot set it for the primary zone) only if the server has secondary zones. The DNS server setting applies to the secondary zone if there is no specific secondary zone setting.

Local Basic or Advanced Web UI

On the Edit DNS Server page, in A-Z view, find the *ixfr-enable* attribute, then set it to enabled. For a secondary zone, you can also fine-tune the incremental zone transfers by setting the *ixfr-expire-interval* attribute. This value is the longest interval the server uses to maintain a secondary zone solely from IXFRs before forcing a full zone transfer (AXFR). The preset value of one week is usually appropriate. Then, click **Modify Server**.

CLI Commands

Use **dns enable ixfr-enable**. By default, the *ixfr-enable* attribute is enabled.

Changesets and Checkpointing

Network Registrar maintains a changeset database that collects recent changes to RR data before they are committed to the authoritative database (auth.db). The DNS server checks the changeset database first before it checks the auth.db when responding to or performing incremental zone transfers or zone checkpointing (see later in this section). A single DNS changeset can consist of one or more RRs.

The changeset database is backed up during the usual **mcdshadow** backup. To keep it from growing without bounds, the server trims it periodically. The effect of this is, for example, that if a DNS client requests a zone IXFR based on a serial number for RRs that were trimmed, the DNS server cannot perform an IXFR, but must perform a full zone transfer (AXFR).

Zone checkpointing creates a snapshot of the zone data that may be necessary to recreate the auth.db in case it is unstable. The server automatically updates the checkpoint files periodically. In addition to occurring with each changeset that is over a certain size threshold, zone checkpointing happens by default every three hours. You can adjust this interval, from between one and 168 hours, using the *checkpoint-interval* DNS server or zone attribute.

Local Basic or Advanced Web UI

To manually force a zone checkpoint, click the Run icon ( ) on the List/Add Zones or List/Add Reverse Zones page. Then, on the Zone Commands for Zone page, click the Run icon ( ) next to Checkpoint zone.

CLI Commands

Use **zone** *name* **chkpt**, or dump the zone checkpoint file in a more humanly readable form by using **zone** *name* **dumpchkpt**.

Restricting Zone Queries

You can restrict clients to query only certain zones based on an access control list (ACL). An ACL can contain source IP addresses, network addresses, TSIG keys (see the "Transaction Security" section on page 28-11), or other ACLs. The ACL set by the DNS server *restrict-query-acl* attribute serves as a filter for non-authoritative queries. If a query targets an authoritative zone, the zone ACL applies.

You can also restrict the server to honor recursive requests from certain clients only by setting the *restrict-recursion-acl* attribute to an ACL that includes these clients (see "Access Control Lists" section on page 28-9). If you unset *restrict-recursion-acl*, anyone can request a recursive query, by default.

You can also set the corresponding *no-recurse* DNS attribute to enable or disable recursion altogether. Only when *no-recurse* is disabled (the preset value) does the *restrict-recursion-acl* setting apply.

Enabling NOTIFY

The NOTIFY protocol, described in RFC 1996, lets the Network Registrar DNS primary server inform its secondaries that zone changes occurred. The NOTIFY packet does not indicate the changes themselves, just that they occurred, and this triggers a zone transfer request. Use NOTIFY in environments where the namespace is relatively dynamic.

Because a zone master server cannot know specifically which secondary server transfers from it, Network Registrar notifies all nameservers listed in the zone NS records. The only exception is the server named in the SOA primary master field.

You can use IXFR and NOTIFY together, but this is not necessary. You can disable NOTIFY for a quickly changing zone for which immediate updates on all secondaries does not warrant the constant NOTIFY traffic. Such a zone might benefit from having a short refresh time and a disabled NOTIFY.

Local Basic or Advanced Web UI

Step 1 On the Edit DNS Server page, in A-Z view, find the *notify* attribute, then set it to enabled.

Step 2 Set any of the other NOTIFY attributes (*notify-defer-cnt*, *notify-min-inverval*, *notify-rcv-interval*, *notify-send-stagger*, *notify-source-address*, *notify-source-port*, and *notify-wait*).

Step 3 Click **Modify Server**.

Step 4 To add nameservers in addition to those specified in NS records, click **Forward Zones**.

Step 5 Click the zone name.

Step 6 Add a comma-separated list of servers using the *notify-set* attribute on the Edit Zone page.

Step 7 Set the *notify* attribute to true.

Step 8 Click **Modify Zone** on that page.

CLI Commands

Use **dns enable notify**. NOTIFY is enabled by default. You can also enable NOTIFY at the zone level, where you can use **zone** *name* **set notify-set** to specify an additional comma-separated list of servers to notify beyond those specified in NS records.

Setting Advanced DNS Server Properties

You can set these advanced server properties:

- **SOA time-to-live**—See the "Setting SOA Time to Live" section on page 17-12.
- **Secondary server attributes**—See the "Setting Secondary Refresh Times" section on page 17-12.
- **Glue records**—See the "Fetching Glue Records" section on page 17-13.
- **Report lame delegation**—See the "Reporting Lame Delegation" section on page 17-14.
- **Cache**—See the "Setting Maximum Negative Cache Times" section on page 17-14.

- **Prevent rogue A record queries**—See the "Handling Rogue Address Records and Other Cache Attributes" section on page 17-16.
- **Query sources**—See the "Setting Query Source Addresses and Port Numbers" section on page 17-17.
- **Port numbers**—See the "Setting Local and External Port Numbers" section on page 17-17.

Setting SOA Time to Live

The SOA record time to live (TTL) is usually determined by the zone default TTL. However, you can explicitly set the SOA TTL, which sets the maximum number of seconds a server can cache the SOA record data. For example, if the SOA TTL is set for 3600 seconds (one hour), an external server must remove the SOA record from its cache after an hour and then query your nameserver again.

Network Registrar responds to authoritative queries with an explicit TTL value. If there is no explicit TTL value, it uses the default TTL for the zone, as set by the value of the *defttl* zone attribute. Databases originating from versions of Network Registrar earlier than Release 3.5 do not have the *defttl* zone attribute, and use the minimum TTL in the zone SOA record for the default TTL.

Normally, Network Registrar assumes the default TTL when responding with a zone transfer with RRs that do not have explicit TTL values. If the default TTL value for the zone is administratively altered, Network Registrar automatically forces a full zone transfer to any secondary DNS server requesting a zone transfer.

Local Basic or Advanced and Regional Web UI

Step 1 On the Add Zone or Edit Zone page, set the Zone Default TTL, which defaults to 24 hours.

Step 2 If you want, set the SOA TTL, which is the TTL for the SOA records only. It defaults to the Zone Default TTL value.

Step 3 You can also set a TTL value specifically for the NS records of the zone. Set the NS TTL value under Nameservers. This value also defaults to the Zone Default TTL value.

Step 4 Click **Modify Zone**.

CLI Commands

Use **zone** *name* **set defttl**.

Setting Secondary Refresh Times

The secondary refresh time is how often a secondary server communicates with its primary about the potential need for a zone transfer. A good range is from an hour to a day, depending on how often you expect to change zone data.

If you use NOTIFY, you can set the refresh time to a larger value without causing long delays between transfers, because NOTIFY forces the secondary servers to notice when the primary data changes. For details about NOTIFY, see the "Enabling NOTIFY" section on page 17-11.

Local Basic or Advanced and Regional Web UI

On the Add Zone or Edit Zone page, set the Secondary Refresh field to the refresh time, which defaults to three hours. Make any other changes, then click **Modify Zone**.

CLI Commands

Use **zone** *name* **set refresh**. The preset value is 10800 seconds (three hours).

Setting Secondary Retry Times

The DNS server uses the secondary retry time between successive failures of a zone transfer. If the refresh interval expires and an attempt to poll for a zone transfer fails, the server continues to retry until it succeeds. A good value is between one-third and one-tenth of the refresh time. The preset value is one hour.

Local Basic or Advanced and Regional Web UI

On the Add Zone or Edit Zone page, set the Secondary Retry field to the retry time, which defaults to one hour. Make any other changes, then click **Modify Zone**.

CLI Commands

Use **zone** *name* **set retry**.

Setting Secondary Expiration Times

The secondary expiration time is the longest time a secondary server can claim authority for zone data when responding to queries after it cannot receive zone updates during a zone transfer. Set this to a large number that provides enough time to survive extended primary server failure. The preset value is seven days.

Local Basic or Advanced and Regional Web UI

On the Add Zone or Edit Zone page, set the Secondary Expire field to the expiration time, which defaults to seven days. Make any other changes, then click **Modify Zone**.

CLI Commands

Use **zone** *name* **set expire**.

Fetching Glue Records

A glue record is a DNS A record that specifies the address of the authoritative nameserver of a zone or subzone. It is an informational record in a query response. For example, most answers include NS records, which then cause the inclusion of A records to resolve the NS record name into an address. These A records are the glue records. Disabling the *no-fetch-glue* attribute tells the server to find records that it would not normally find, so that it can include them in answers to subsequent queries.

Local Basic or Advanced Web UI

On the Edit DNS Server page, in A-Z view, find the *no-fetch-glue* attribute, set it to disabled, then click **Modify Server**.

CLI Commands

Use **dns disable no-fetch-glue** (the preset value).

Reporting Lame Delegation

Lame delegation occurs when a DNS server listed in the parent delegation of a zone does not know that it is authoritative for the zone. The server can detect and report this when, in the process of tracking down an answer, it is referred to a server that in turn refers to another server for a domain closer to the root (actually farther from the answer). Lame delegation does not indicate a problem with the DNS configuration, but with the configuration at the DNS server you are querying. You cannot do anything to correct lame delegation at other domains, unless you happen to be authoritative for the domain as well.

Local Basic or Advanced Web UI

On the Edit DNS Server page, in A-Z view, under *log-settings*, click a check mark next to the **lame-delegation** value, then click **Modify Server**.

CLI Commands

Use **dns set log-settings=lame-delegation** (one of the preset values).

Setting Maximum Negative Cache Times

To ensure a quick response to repeated requests for the same information, the DNS server maintains a cache of data it learns from other servers on behalf of its clients. This includes negative information, such as “no such name” or “no such data,” as specified by RFC 2308. It is important to discard this information at some point to accommodate namespace changes at the authoritative source.

The maximum negative cache time establishes an upper bound on the time a negative cache entry can be valid. This value should be obtained from the SOA record in the negative response. If the originating zone has an abnormally large TTL value, the maximum negative cache time value reduces that value, limiting the time that the entry remains negatively cached. Choose this value carefully to balance the need to eliminate long negative cache entries with the desire to cache negative answers meaningfully.

Note that you can effectively reduce the maximum negative cache time through the maximum cache TTL value, because this value applies to all cache entries, positive and negative. See the “Setting Maximum Cache TTLs” section on page 17-15. The smaller of the TTL values of the *max-cache-ttl* and *max-negcache-ttl* wins when caching negative entries.

Local Basic or Advanced Web UI

On the Edit DNS Server page, in A-Z view, find the *max-negcache-ttl* attribute, set it to the desired value (the preset value is one hour), then click **Modify Server**.

CLI Commands

Use **dns set max-negcache-ttl** (the preset value is 60 minutes).

Setting Maximum Cache TTLs

The Maximum Cache TTL property specifies the maximum time that you want Network Registrar to retain cached information. TTL is the amount of time that any nameserver is allowed to cache data learned from other nameservers. Each record added to the cache arrives with some TTL value. When the TTL period expires, the server must discard the cached data and get new data from the authoritative nameservers the next time it sends a query. This parameter limits the lifetime of records in the cache whose TTL values are very large. The preset value is seven days (604800 seconds).

Local Basic or Advanced Web UI

On the Edit DNS Server page, in A-Z view, find the *max-cache-ttl* attribute, set it to the desired value (the preset value is one week), then click **Modify Server**.

CLI Commands

Use **dns set max-cache-ttl**.

Setting Maximum Memory Cache Sizes

The maximum memory cache size property specifies how much memory space you want to reserve for the DNS in-memory cache. The more memory that you allocate for the cache, the less frequently the server uses disk cache. One cache entry is about 100 bytes. The preset value prior to Network Registrar 6.0 was 200 KB; the value as of 6.0 is 10000 KB (10 MB). The recommended value is 30000 (30 MB).

Local Basic or Advanced Web UI

On the Edit DNS Server page, in A-Z view, find the *mem-cache-size* attribute, set it to the desired value, then click **Modify Server**. The recommended value is 30000 (30 MB).

CLI Commands

Use **dns set mem-cache-size**.

Flushing DNS Cache

The Network Registrar cache flushing function lets you stop the disk cache file from growing. However, the actual behavior depends on whether the DNS server is running or stopped. If you flush the cache:

- While the server is running, Network Registrar clears all expendable entries from the cache database file. Flushing the cache does not shrink the file because of the nature of the database, but does create free space in it. Because the memory cache is unaffected by this operation, recently used cache entries are not lost, and performance is not significantly affected.
- With the server stopped, Network Registrar interprets the request to flush all entries and removes the cache file. It then reinitializes the database when you restart the server.

To clear a cache that grew too large, or when changing a resolution exception, stop the server, enter the command, then restart the server. Stopping the server does not terminate the server process, but stops it from handling further requests. For details on resolution exception, see the "Using Resolution Exception" section on page 17-4.

Local Basic or Advanced Web UI

On the Manage DNS Server page, click the Run icon ( ) in the Commands column to open the DNS Server Commands page (see Figure 7-3 on page 7-3). On this page, click the Run icon ( ) next to Flush cache. If you want to flush cache older than a certain time, enter the time in the Time field next to Flush cache older than, then click the Run icon.

CLI Commands

Use **dns flushCache**.

Handling Rogue Address Records and Other Cache Attributes

You may become victim of a suspicious denial-of-service attack where a rogue host targets Address (A) RR queries to a caching DNS server. These A record queries contain names that resemble IP addresses. To avoid overloading the DNS server cache.db file with negative responses from the root, the server no longer tries to resolve these queries. The *fake-ip-name-response* DNS attribute (*Fake Responses for IP address-like names* in the web UI) is enabled by default to effect this. When the server receives a query for a non-authoritative name, it consults its cached data. If the server cannot answer from cached data, it examines the queried name to see if it resembles an IP address (four decimals each separated by a dot with no trailing or preceding characters). If so, it does not forward the query. Instead, the server sends a response with a NAME ERROR (NXDOMAIN) return code, and does not cache this synthesized negative response.

The server acts on the *save-negative-cache-entries* (*Save negative cache entries to disk* in the web UI) and *cache-write-ttl-threshold* attributes when it evicts entries from its memory cache to the cache.db file. It typically evicts positive and negative query responses from in-memory cache when the in-memory cache is full and the server needs to inserts a new entry. The server evicts the least-recently-used entry. If you disable *save-negative-cache-entries*, the server does not store evicted negative entries in the cache.db file, but simply discards them from the in-memory server cache. If the *cache-write-ttl-threshold* value is nonzero (it is zero by default), the server only persists entries from the in-memory cache to the cache.db file if the TTLs of an entry are greater than this value. Otherwise, the server discards the (soon to be, but not yet, expired) RR. A zero value causes the server to always persist unexpired RRs.

Local Basic or Advanced Web UI

On the Edit DNS Server page, in A-Z view, find the *fake-ip-name-response* and *save-negative-cache-entries* attributes, set them to enabled, then click **Modify Server**.

CLI Commands

Use **dns enable fake-ip-name-response** and **dns enable save-negative-cache-entries**.

Setting Query Source Addresses and Port Numbers

The query source address is the source IP address from which the DNS server should send queries to other servers when resolving names for clients. A value of 0.0.0.0 indicates that the best local address is used based on the destination.

The query source port is the UDP port number from which the DNS server should send queries to other servers when resolving names for clients. A value of zero indicates a random port, which is often used when the server is behind a firewall, and an unset value causes it to be sent from the local port (see the "Setting Local and External Port Numbers" section).

Local Basic or Advanced Web UI

On the Edit DNS Server page, in A-Z view, find the *query-source-address* and *query-source-port* attributes, set them to the desired values (there are none preset), then click **Modify Server**.

CLI Commands

Use **dns set query-source-address** and **dns set query-source-port**.

Setting Local and External Port Numbers

If you are experimenting with a new group of nameservers, you might want to use nonstandard ports for answering requests and asking for remote data. The local port and external port settings control the TCP and UDP ports on which the server listens for name resolution requests, and to which port it connects when making requests to other nameservers. The standard value for both is port 53. If you change these values during normal operation, the server will appear to be unavailable.

The full list of default ports is included in the "Default Ports for Network Registrar Services" section on page 1-11.

Local Basic or Advanced Web UI

On the Edit DNS Server page, in A-Z view, find the *local-port-num* and *remote-port-num* attributes, set them to the desired values (they are both preset to 53), then click **Modify Server**.

CLI Commands

Use **dns set local-port-num** and **dns set remote-port-num**.

Tuning DNS Properties

Here are some suggestions to tune some of the DNS server properties:

- ***Notify send min. interval* DNS server attribute (*notify-min-interval* in the CLI)**—Minimum interval required before sending notification of consecutive changes on the same zone to a server. The preset value is two seconds. For very large zones, you might want to increase this value to exceed the maximum time to send an outbound full zone transfer. This is recommended for secondary servers that receive inbound incremental zone transfers and send out full transfers to other secondaries. These include older BIND servers that do not support incremental zone transfers. Inbound incremental transfers may abort outbound full transfers.

- ***Notify delay between servers* DNS server attribute (*notify-send-stagger* in the CLI)**—Interval to stagger notification of multiple servers of a change. The preset value is one second, but you may want to raise it to up to five seconds if you need to support a large number of zone transfers distributed to multiple servers.
- ***Notify wait for more changes* DNS server attribute (*notify-wait* in the CLI)**—Time to delay, after an initial zone change, before sending change notification to other nameservers. The preset value is five seconds, but you may want to raise it to 15, for the same reason as given for the *notify-min-interval* attribute.
- ***Max. memory cache size* DNS server attribute (*mem-cache-size* in the CLI)**—Size of the in-memory record cache, in kilobytes. The preset value is 200 KB, but you may want to raise it to 1000 KB for larger networks.
- ***Report lame delegations* DNS server attribute (*lame-deleg-notify* in the CLI)**—Network Registrar should notice and log when a DNS server listed in a parent-zone delegation of subzones does not know that it is authoritative for the zone. This is normally disabled, but you may want to enable it. This attribute has the same effect as setting the log settings to *lame-delegation*.
- **IXFR check box in the Foreign Servers section of the Edit DNS Server page, or remote-dns *address/mask* create ixfr in the CLI**—Adding an entry for a server or group of servers allows controlling whether or not IXFR should occur when doing zone transfers from those servers.

Troubleshooting DNS Servers

Useful troubleshooting hints and tools to diagnose the DNS server and ways to increase performance include:

- **Restoring a loopback zone**—A loopback zone is a reverse zone that enables a host to resolve the loopback address (127.0.0.1) to the name *localhost*. The loopback address is used by the host to enable it to direct network traffic to itself. You can configure a loopback zone manually or you can import it from an existing BIND zone file.
- **Listing the values of the DNS server attributes**—Click **DNS**, then **DNS Server** to open the Edit DNS Server page in the web UI. In the CLI, use **dns show**.
- **Adjusting certain attribute values that could have inherited preset values from previous releases during an upgrade**—For deployments that were upgraded from Network Registrar 5.5.*x* or earlier, the DNS server might be operating with legacy preset values for critical settings. These preset values are probably not optimal for current systems and can cause performance issues. We strongly recommend that you update the legacy settings to use the new preset values. Table 17-1 lists the old and new preset values, along with a recommended setting for each attribute.

Table 17-1 DNS Attributes with Changed Preset Values

DNS Attribute	5.5.*x* Preset Value	7.0 Preset Value	Recommended Setting
auth-db-cache-kbytes	1000 (KB)	5120 (KB)	51200 (KB)
axfr-multirec-default	off	on	on
cache-db-cache-kbytes	1000 (KB)	5120	51200 (KB)
changeset-db-cache-size	1000 (KB)	10000 (KB)	10000 (KB)
changeset-db-logs-trimming-interval	10m	30m	30m
htrim-zone-seek-more-trim-interval	60m	5m	5m
mem-cache-size	200 (KB)	10000 (KB)	30000 (KB)

Table 17-1 DNS Attributes with Changed Preset Values (continued)

DNS Attribute	5.5.*x* Preset Value	7.0 Preset Value	Recommended Setting
zone-db-cache-kbytes	1000 (KB)	1024 (KB)	1024 (KB)
query-source-port	53	53	0 (let the operating system choose the outbound DNS port)

For many of these attributes, you must enter Expert mode in the web UI or use **set session visibility=3** in the CLI. To change the preset value to the current one, unset the attribute. To change to the recommended setting, change the attribute value.

Be sure to reload the DNS server after saving the settings.

- **Choosing from the DNS log settings to give you greater control over existing log messages**—Use the *Log settings* attribute on the Edit DNS Server page in the web UI, or **dns set log-settings** in the CLI, with one or more of these keyword or numeric values, separated by commas (see Table 17-2). Restart the server if you make any changes to the log settings.

Table 17-2 DNS Log Settings

Log Setting (Numeric Equivalent)	Description
config (1)	Server configuration and deinitialization.
ddns (2)	High level dynamic update messages.
xfr-in (3)	Inbound full and incremental zone transfers.
xfr-out (4)	Outbound full and incremental zone transfers.
notify (5)	NOTIFY transactions.
datastore (8)	Data store processing that provides insight into various events in the server embedded databases.
scavenge (9)	Scavenging of dynamic RRs (see the "Scavenging Dynamic Records" section on page 28-18).
scavenge-details (10)	More detailed scavenging output (disabled by default).
server-operations (11)	General high-level server events, such as those pertaining to sockets and interfaces.
lame-delegation (13)	Lame delegation events; although enabled by default, disabling this flag could prevent the log from getting filled with frequent lame delegation encounters.
root-query (14)	Queries and responses from root servers.
ddns-refreshes (15)	DNS update refreshes for Windows 2000 clients (disabled by default).
ddns-refreshes-details (16)	RRs refreshed during DNS updates for Windows 2000 clients (disabled by default).
ddns-details (17)	RRs added or deleted due to DNS updates.
tsig (18)	Logs events associated with Transaction Signature (TSIG) DNS updates (see the "Transaction Security" section on page 28-11).
tsig-details (19)	More detailed logging of TSIG DNS updates (disabled by default).

Table 17-2 DNS Log Settings (continued)

Log Setting (Numeric Equivalent)	Description
activity-summary (20)	Summary of activities in the server. You can adjust the interval at which these summaries are taken using the *activity-summary-interval* attribute, which defaults to five-minute intervals (you can adjust this interval using **dns set activity-summary-interval**).
query-errors (21)	Logs errors encountered while processing DNS queries.
config-details (22)	Generates detailed information during server configuration by displaying all configured and assumed server attributes (disabled by default).
incoming-packets (23)	Incoming data packets.
outgoing-packets (24)	Outgoing data packets.
xfer-in-packets (25)	Incoming full zone transfer (XFR) packets.
query-packets (26)	Incoming query packets.
notify-packets (27)	NOTIFY packets.
ddns-packets (28)	DNS Update packets.
xfer-out-packets (29)	Outgoing XFR packets.
ha-details (30)	Generates detailed logging of High-Availability (HA) DNS information.

- **Using the nslookup utility to test and confirm the DNS configuration**—This utility is a simple resolver that sends queries to Internet nameservers. To obtain help for the **nslookup** utility, enter **help** at the prompt after you invoke the command. Use only fully qualified names with a trailing dot to ensure that the lookup is the intended one. An **nslookup** begins with a reverse query for the nameserver itself, which may fail if the server cannot resolve this due to its configuration. Use the **server** command, or specify the server on the command line, to ensure that you query the proper server. Use the **–debug**, or better yet, the **–d2**, flag to dump the responses and (with **–d2**) the queries being sent.



CHAPTER 18

Configuring High-Availability DNS Servers

DNS was designed to have one primary server and multiple secondaries as authoritative for a zone. This works well for static addressing, because there is only one instance (the primary zone file) modified, while the secondaries periodically probe for updates from the primary, or the primary notifies the secondaries of them when the zone is loaded or reloaded.

This scenario has shortcomings with DNS updates under the RFC 2136 protocol where DHCP dynamically updates the DNS server, and only the primary DNS server can accept updates. This presents a single point of failure in that DNS updates cannot happen if the primary goes down.

To solve this problem, a second primary server can be made available as a hot standby that shadows the main primary server. This configuration is called High-Availability (HA) DNS. The Network Registrar web UI and CLI have features with which you can duplicate the primary setup required for HA DNS for the server pair. The server pair is responsible for detecting communication failures and the like.

See Also

HA DNS Processing
Configuring an HA DNS Server Pair, page 18-2
DNS Server Configuration for HA DNS, page 18-4
HA DNS Statistics, page 18-4

HA DNS Processing

In normal state, both the main and backup primary servers are up and running. The main server processes all DNS updates from clients and sends all accepted updates to the hot standby backup. The backup server refuses any DNS updates during normal times when the main is running and communicating with the backup. Both servers respond to nonupdate queries and zone transfers. The main and backup partners exchange heartbeat messages to detect if the other is not available.

If the hot standby backup goes down, the main waits a short time, then records the updates that the partner did not acknowledge. When the backup server comes back up, the main sends the recorded updates to the backup. If the backup has been down for an extended period, the main sends its entire zone data to the backup, essentially a full zone transfer.

If the main goes down, the backup waits a short time, then begins servicing the DNS updates from clients that the main would normally service and records the updates. When the main returns, the backup sends it the updates, and the main synchronizes with the backup any unsent updates it had before it went down. During the short synchronization period, neither server accepts DNS updates.

Both the main and backup can traverse the following states:

- **Startup**—The servers establish communication and agree on the HA version to use. In this state, the servers do not accept DNS updates or RR edits, and they defer scavenging, if enabled.
- **Normal**—Both servers are up and healthy, exchanging DNS updates and heartbeat messages. The main accepts DNS updates and RR edits, sends RR Update messages to the backup, and performs history trimming and scavenging, if enabled. The backup ignores DNS updates, refuses RR edits, but processes RR Update messages from the main server. The backup also performs history trimming, but defers scavenging, if enabled.
- **Communication-Interrupted**—The server goes into this state after not getting a response or request from the partner during the communication timeout (*ha-dns-comm-timeout*) period (preset to 30 seconds). The server continues listening for communication from the partner (they both send heartbeat messages every 12 seconds) and tries to connect, meanwhile accepting DNS updates and RR edits and disabling scavenging.
- **Partner-Down**—The server administrator notifies the partner that it will be down for an extended time. This manual intervention is possible only in Communication-Interrupted state. Either server continues listening for communication from the partner and tries to connect, accepts DNS updates and RR edits, and performs scavenging.
- **Synchronization**—Once the partners establish or reestablish communication, they synchronize RR changes that occurred during the interrupted period.
- **Synchronization-Pending**—Each server is waiting for the other to get ready to synchronize. In this state, DNS Updates and RR edits are not allowed.

When a DNS server starts up, it:

1. Tries to establish a connection with its partner.
2. Goes into Synchronization-Pending mode.
3. Goes into Synchronization mode once it receives a Synchronization-Pending response.
4. Goes into Normal mode.

Note HA DNS is fully integrated with DHCP servers updating the partners when hosts get added to the network (see Chapter 28, "Configuring DNS Update").

Configuring an HA DNS Server Pair

The attributes needed to set up an HA DNS server pair are:

- ***ha-dns***—Enabled or disabled. The preset value is disabled, so that this attribute must be set explicitly.
- ***main***—IP address of the main primary DNS server.
- ***backup***—IP address of the backup primary DNS server.

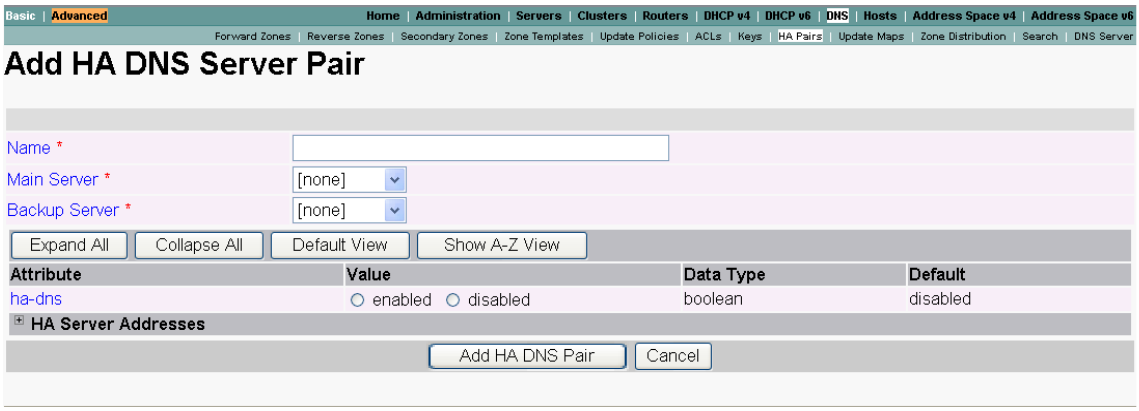
Local Basic or Advanced and Regional Web UI

Step 1 Create a cluster for the backup server.

Step 2 Click **DNS**, then **HA Pairs** to open the List HA DNS Server Pairs page.

Step 3 Click **Add HA DNS Server Pair** to open the Add HA DNS Server Pair page (see Figure 18-1 for the Advanced mode view of the page).

Figure 18-1 Add HA DNS Server Pair Page (Local Advanced)



Step 4 Enter the name of the server pair in the Name field. This can be any identifying text string.

Step 5 Click the cluster name of the main DNS server in the Main Server drop-down list.

Note If you change the IP address of your local host machine, you must modify the localhost cluster (on the Edit Cluster page) to change the address in the IP Address field. Do not set the value to 127.0.0.1.

Step 6 Click the cluster name of the backup DNS server in the Backup Server drop-down list. This cannot be the same as the main server cluster. Set the *ha-dns-main-server* and *ha-dns-backup-server* attributes only if the server is configured with different interfaces for configuration management and update requests. (Configure the HA DNS protocol only with the interface used to service updates.)

Step 7 Click the *ha-dns* enabled button to enable HA DNS for the server pair.

Step 8 Click **Add HA DNS Pair**.

Step 9 Once the server pair appears on the List HA DNS Server Pairs page, synchronize the servers:

a. Click the Report icon ( ) in the Synchronize column.

b. On the Report Sync HA DNS Pair page, choose the direction of synchronization (Main to Backup or Backup to Main).

c. Choose the operation type (Update, Complete, or Exact). See the table on the page for details on the operations for each operation type.

d. Click **Report** to display the prospective synchronization changes on the View HA DNS Sync Report page.

e. Click **Run** to complete the synchronization and view the actual changes. The configuration gets pushed to the remote cluster.

f. Click **Return** to return to the List HA DNS Server Pairs page.

Step 10 Reload both DNS servers to begin HA communication. The DNS servers synchronize things such as unprotected RRs themselves when they start communicating.

CLI Commands

Be sure that the main and backup DNS servers have the same zone configurations. Then explicitly enable HA DNS (**ha-dns-pair** *name* **enable ha-dns**). Create the HA DNS server pair (**ha-dns-pair** *name* **create** *mainaddr backupaddr*). Then synchronize the servers using **ha-dns-pair** *name* **sync**, specifying the synchronization operation (update, complete, or exact) and direction (main-to-backup or backup-to-main). For example:

```
nrcmd> ha-dns-pair enable ha-dns
nrcmd> ha-dns-pair examplehadnspair create localhost test-cluster
nrcmd> ha-dns-pair examplehadnspair sync exact main-to-backup
```

The CLI provides an additional command for the DNS server to set the HA DNS partner down, if necessary, which is possible only while in Communication-Interrupted state:

```
nrcmd> dns setPartnerDown
```

DNS Server Configuration for HA DNS

The only attribute on the main DNS server that addresses HA DNS is the *ha-dns-comm-timeout* attribute. This is the time required to determine if a partner is unreachable, after network communication is not acknowledged, which triggers the Communication-Interrupted state (see the description of this state in the "HA DNS Processing" section on page 18-1). The preset value is 30s. The server tries to communicate and then back off at multiples of the *ha-dns-comm-timeout* interval.

An additional log setting, *ha-details*, enables logging of HA DNS-related information.

Note that the HA DNS configuration is possible for Network Registrar 6.2 and later DNS servers only. Both the main and backup must have absolutely identical zone and RR configurations, and you must set the same HA DNS attributes for both servers.

HA DNS Statistics

You can view HA DNS statistics.

Local Basic or Advanced Web UI

Click the Statistics icon () on the Manage DNS Server page to open the DNS Server Statistics page. The statistics appear under the Max Counter Statistics subcategories of both the Total Statistics and Sample Statistics categories.

CLI Commands

Use **dns getStats ha** [**total**] to view the HA DNS Total counters statistics, and **dns getStats ha sample** to view the Sampled counters statistics.



PART 5

Dynamic Host Administration



CHAPTER 19

Introduction to Dynamic Host Configuration

All hosts seeking Internet access must have an IP address. As Internet administrator, you must do the following for every new user and for every user whose computer was moved to another subnet:

1. Choose a legal IP address.
2. Assign the address to the individual workstation.
3. Define workstation configuration parameters.
4. Update the DNS database, mapping the workstation name to the IP address.

These activities are time consuming and error prone, hence the Dynamic Host Configuration Protocol (DHCP). DHCP frees you from the burden of individually assigning IP addresses. It was designed by the Internet Engineering Task Force (IETF) to reduce the amount of configuration required when using TCP/IP. DHCP allocates IP addresses to hosts. It also provides all the parameters that hosts require to operate and exchange information on the Internet network to which they are attached.

DHCP localizes TCP/IP configuration information. It also manages allocating TCP/IP configuration data by automatically assigning IP addresses to systems configured to use DHCP. Thus, you can ensure that hosts have Internet access without having to configure each host individually.

See Also

How DHCP Works
Network Registrar DHCP Implementations, page 19-4
DNS Update, page 19-6
DHCP Failover, page 19-8
Client-Classes, page 19-11

How DHCP Works

DHCP makes dynamic address allocation possible by shifting workstation configuration to global address pools at the server level. DHCP is based on a client/server model. The client software runs on the workstation and the server software runs on the DHCP server.

See Also

Sample DHCP User, page 19-2
Typical DHCP Administration, page 19-2
Leases, page 19-3
Scopes and Policies, page 19-3

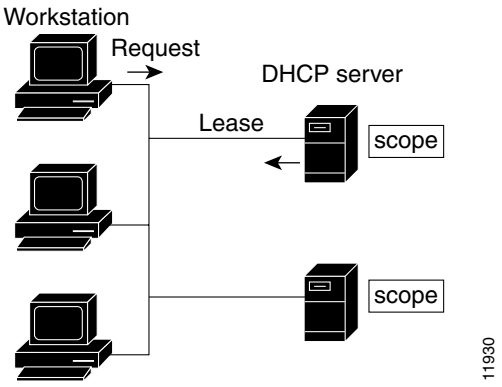
Sample DHCP User

After Beth's workstation (bethpc) is configured with DHCP, these actions occur when she first starts up:

1. Her workstation automatically requests an IP address from a DHCP server on the network.
2. The DHCP server offers her a lease that is an IP address with the configuration data necessary to use the Internet. Nobody else uses the leased address, and it is valid only for her workstation.
3. Before the address lease expires, bethpc renews it, thereby extending the expiration time. It continues to use the lease right up to its expiration or if it cannot reach the server.
4. If Beth relocates to another department and her workstation moves to a different subnet, her current address expires and becomes available for others. When Beth starts her workstation at its new location, it leases an address from an appropriate DHCP server on the subnet (see Figure 19-1).

As long as the DHCP server has the correct configuration data, none of the workstations or servers using DHCP will ever be configured incorrectly. Therefore, there is less chance of incurring network problems from incorrectly configured workstations and servers that are difficult to trace.

Figure 19-1 Hosts Request an IP Address



The example shows the DHCP protocol with a set of DHCP servers that provide addresses on different subnets. To further simplify the administration of address pools, network routers are often configured as DHCP relay agents to forward client messages to a central DHCP server. This server is configured with address pools for a group of subnets.

Typical DHCP Administration

To use DHCP, you must have at least one DHCP server on the network. After you install the server:

- Define a scope of IP addresses that the DHCP server can offer to DHCP clients. You no longer need to keep track of which addresses are in use and which are available.
- Configure a secondary server to share the distribution or handle leases if the first DHCP server goes down. This is known as DHCP failover, and is described further in the "DHCP Failover" section on page 19-8.

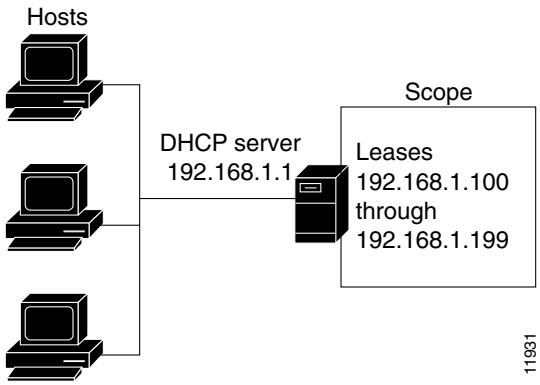
Leases

One of the most significant benefits of DHCP is that it can dynamically configure workstations with IP addresses and associate leases with the assigned addresses. DHCP uses a lease mechanism that offers an automated, reliable, and safe method for distributing and reusing addresses in networks, with little need for administrative intervention. As system administrator, you can tailor the lease policy to meet the specific needs of your network.

Leases are grouped together in an address pool, called a scope, which defines the set of IP addresses available for requesting hosts. A lease can be reserved (the host always receives the same IP address) or dynamic (the host receives the next available, unassigned lease in the scope). The DHCP server of the site is configured to lease addresses 192.168.1.100 through 192.168.1.199 (see Figure 19-2).

If you plan not to have more network devices than configured addresses for the scope, you can define long lease times, such as one to two weeks, to reduce network traffic and DHCP server load.

Figure 19-2 DHCP Hosts Requesting Leases from a DHCP Server



Scopes and Policies

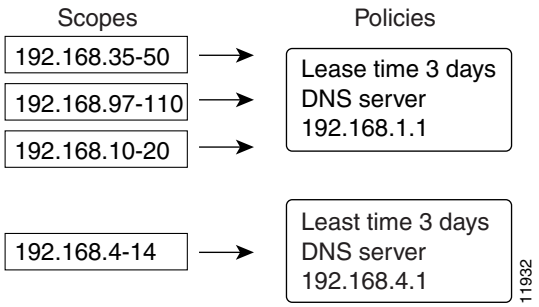
A scope contains a set of addresses for a subnet, along with the necessary configuration parameters. You must define at least one scope for each subnet for which you want dynamic addressing.

A policy includes lease times and other configuration parameters that a DHCP server communicates to clients. Use policies to configure DHCP options that the DHCP server supplies to a client upon request. Policies ensure that the DHCP server supplies all the correct options for scopes without having to do so separately for each scope (see Figure 19-3 on page 19-4).

The difference between scopes and policies is that scopes contain server information about addresses, such as which address is leasable and whether to ping clients before offering a lease. Policies contain client configuration data, such as the lease duration and address of the local DNS server.

Policies are especially useful if you have multiple scopes on a server. You can create policies that apply to all or selected scopes. The Network Registrar policy hierarchy is a way to define policies from least to most specific. For example, you usually specify a router option for each policy, which means that you would need a policy for each scope. Scope-specific policies like this can be defined in a scope-embedded policy. More general policies, such as those referring to lease times, can be applied in a system-wide policy (see the "Configuring DHCP Policies" section on page 21-1). You can also write extensions to handle policy assignments (see the "Using Extensions to Affect DHCP Server Behavior" section on page 23-9).

Figure 19-3 Scopes and Policies



Network Registrar DHCP Implementations

The Network Registrar DHCP server provides a reliable method for automatically assigning IP addresses to hosts on your network. You can define DHCP client configurations, and use the Network Registrar database to manage assigning client IP addresses and other optional TCP/IP and system configuration parameters. The TCP/IP assignable parameters include:

- IP addresses for each network adapter card in a host.
- Subnet masks for the part of an IP address that is the physical (subnet) network identifier.
- Default gateway (router) that connects the subnet to other network segments.
- Additional configuration parameters you can assign to DHCP clients, such as a domain name.

Network Registrar automatically creates the databases when you install the DHCP server software. You add data through the web UI or CLI as you define DHCP scopes and policies.

The Network Registrar DHCP server also supports allocating addresses in virtual private networks (VPNs) and subnets to pool manager devices for on-demand address pools. These features are described in the following sections.

See Also

DHCP and IPv6, page 19-4
Virtual Private Networks, page 19-4
Subnet Allocation and DHCP Address Blocks, page 19-5

DHCP and IPv6

For details on the Network Registrar implementation of DHCPv6, see Chapter 26, “Managing DHCPv6 Addresses.”

Virtual Private Networks

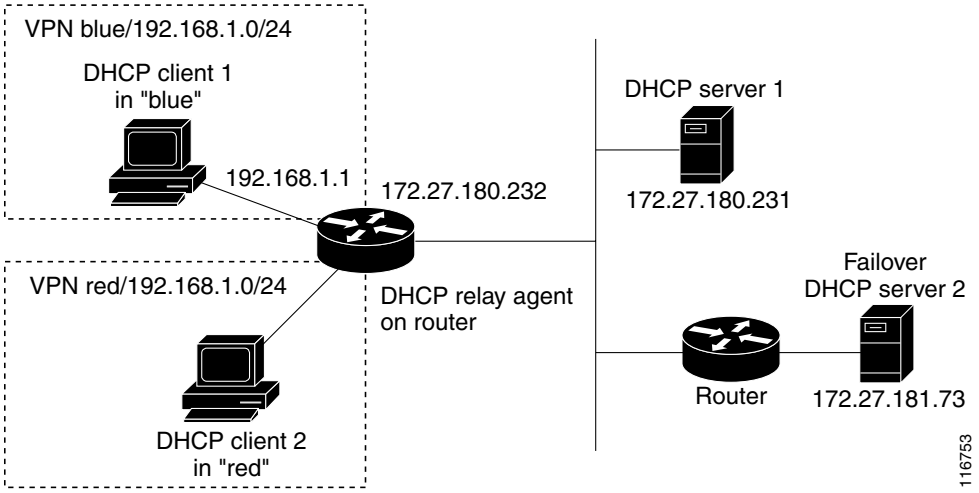
Virtual private networks (VPNs) allow the possibility that two pools in separate networks can have the same address space, with these two pools having overlapping private network addresses. This can save address resources without having to use valuable public addresses. These VPN addresses, however,

require a special designator to distinguish them from other overlapping IP addresses. Network Registrar DHCP servers that are not on the same VPN as their clients can now allocate leases and addresses to these clients, and can distinguish the addresses from one VPN to another.

Through changes made to the Network Registrar DHCP server and Cisco IOS DHCP Relay Agent, the DHCP server can service clients on multiple VPNs. A VPN distinguishes a set of DHCP server objects, making them independent of otherwise identical objects in other address spaces. You can define multiple VPNs containing the same addresses. You create a VPN based on the VPN identifier configured in the Cisco IOS Relay Agent.

Figure 19-4 shows a typical VPN-aware DHCP environment. The DHCP Relay Agent services two distinct VPNs, blue and red, with overlapping address spaces. The Relay Agent has the interface address 192.168.1.1 on VPN blue and is known to DHCP Server 1 as 172.27.180.232. The server, which services address requests from DHCP Client 1 in VPN blue, can be on a different network or network segment than the client, and can be in a failover configuration with DHCP Server 2 (see the "DHCP Failover" section on page 19-8). The Relay Agent can identify the special, distinguished route of the client address request to the DHCP server, as coordinated between the Relay Agent and Network Registrar administrators. The DHCP servers can now issue leases based on overlapping IP addresses to the clients on both VPNs.

Figure 19-4 Virtual Private Network DHCP Configuration



Subnet Allocation and DHCP Address Blocks

Network Registrar supports creating on-demand address pools as a network infrastructure for address provisioning and VPNs. Traditionally, the DHCP server is limited to interact with individual host devices. Through subnet allocation, the server can interact with VPN routers and other provisioning devices to provision entire IP subnets. This Network Registrar feature enhances the on-demand address pool capability currently supported by the Cisco IOS Relay Agent.

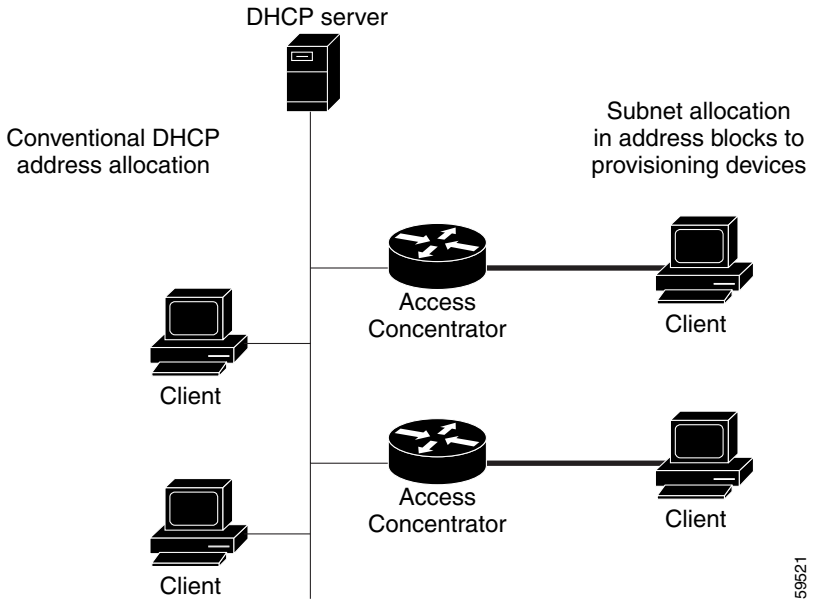
Network Registrar supports explicitly provisioned subnets. You must explicitly configure the DHCP server address space and subnet allocation policies before the server can allocate pools or leases. You can thereby configure a server as a pool manager to manage subnets and delegate them to client devices.

You manage DHCP subnet allocation using DHCP server address block objects in Network Registrar. A DHCP address block is a range of contiguous IP addresses delegated to the DHCP server for assignment. The server expects to subdivide these addresses into pools so that it or other servers or devices can

allocate them. DHCP address blocks are parents to subnets. These DHCP address blocks are distinct from the address blocks you can create using the Network Registrar web UI, which are static. DHCP address blocks cannot include static address ranges or lease reservations.

Figure 19-5 shows a sample environment where a DHCP server allocates entire subnets to access concentrators or other provisioning devices, in addition to servicing individual clients. The traditional client/server relationship is shown on the left of the diagram, while the subnet allocation to access concentrators is shown on the right of the diagram. Dialup customers, for example, connect to the service provider network at two ISP gateways (routers), which connect to the management network segment where the DHCP server resides. The gateways provision addresses to their connected clients based on the subnet requested from the DHCP server.

Figure 19-5 Sample DHCP Subnet Allocation Configuration



DNS Update

Although DHCP frees you from the burden of distributing IP addresses, it still requires updating the DNS server with DHCP client names and addresses. DNS update automates the task of keeping the names and addresses current. With the Network Registrar DNS update feature, the DHCP server can tell the corresponding DNS server when a name-to-address association occurs or changes. When a client gets a lease, Network Registrar tells the DNS server to add the host data. When the lease expires or when the host gives it up, Network Registrar tells the DNS server to remove the association.

In normal operation, you do not have to manually reconfigure DNS, no matter how frequently clients' addresses change through DHCP. Network Registrar uses the hostname that the client workstation provides. You also can have Network Registrar synthesize names for clients who do not provide them, or use the client lookup feature to use a preconfigured hostname for the client.

See Also

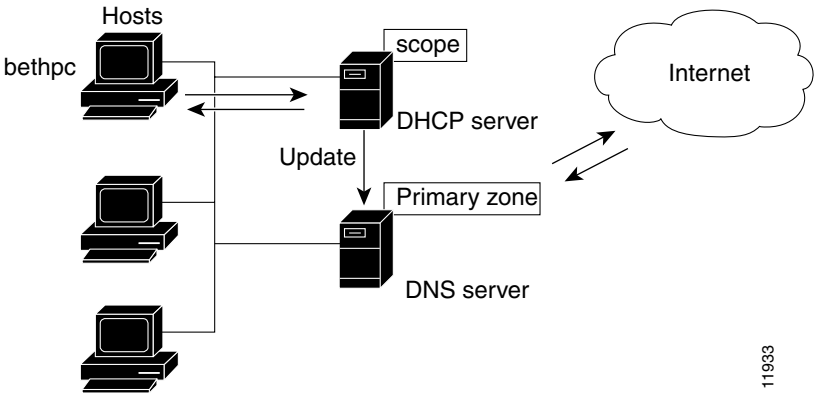
Effect on DNS of Obtaining Leases
Effect on DNS of Releasing Leases, page 19-7
Effect on DNS of Reacquiring Leases, page 19-8

Effect on DNS of Obtaining Leases

For ExampleCo, the administrator creates a scope on the DHCP server and allocates 100 leases (192.168.1.100 through 192.168.1.199). Each workstation gets its owner name. The administrator also configures the DHCP server to use DNS update and associates it with the correspondingly configured DNS server. The administrator does not need to enter the names in the DNS server database.

Monday morning, Beth (user of bethpc) tries to log in to a website without having an address. When her host starts up, it broadcasts an address request (see Figure 19-6).

Figure 19-6 DNS Update at ExampleCo Company



The DHCP server then:

1. Gives bethpc the next available (unassigned) IP address (192.168.1.125).
2. Updates her DNS server with the hostname and address (bethpc 192.168.1.125).

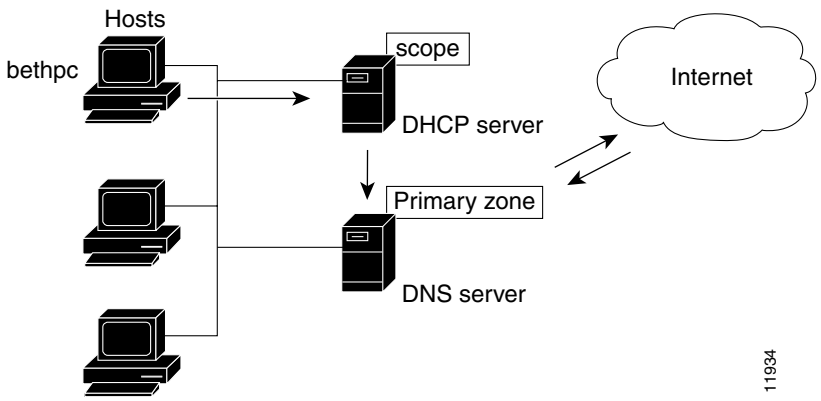
Beth can now access the website. In addition, programs that need to translate the name of Beth's machine to her IP address, or the other way around, can query the DNS server.

Effect on DNS of Releasing Leases

Later that day, Beth learns that she needs to travel out of town. She turns off her host, which still has a leased address that is supposed to expire after three days. When the lease is released, the DHCP server:

1. Acknowledges that the IP address is now available for other users (see Figure 19-7 on page 19-8).
2. Updates the DNS server by removing the hostname and address. The DNS server no longer stores data about bethpc or its address.

Figure 19-7 Relinquishing a Lease



Effect on DNS of Reacquiring Leases

When Beth returns from her trip to start up her host again:

1. Her workstation broadcasts for an IP address.
2. The DHCP server checks if the host is on the correct network. If so, the server issues an address. If not, the server on the correct network issues the address.
3. The DHCP server updates the DNS server again with the host and address data.

DHCP Failover

Because DHCP provides for multiple servers (see RFC 2131), you can configure these servers so that if one cannot provide leases to requesting clients, another one can take over. Network Registrar provides the DHCP failover feature, where two servers operate as redundant partners. Existing DHCP clients can keep and renew their leases without needing to know or care which server is responding to their requests.

See Also

How Failover Works
Failover States and Transitions
Allocating Addresses Through Failover, page 19-11

How Failover Works

Failover is based on a partner server relationship. The partners must have identical scopes, leases, policies, and client-classes. After the servers start up, each contacts the other. The main server provides its partner with a private pool of addresses and updates its partner with every client operation. If the main server fails, then the partner takes over offering and renewing leases, using its private pool. When the main server becomes operational again, it reintegrates with its partner without administrative intervention. These servers are in a relationship known as a failover pair.

The failover protocol keeps DHCP operational if:

- **The main server fails**—The partner takes over services during the time the main server is down. The servers cannot generate duplicate addresses, even if the main server fails before updating its partner.
- **Communication fails**—A partner can operate correctly even though it cannot tell whether it was the other server or the communication with it that failed. The servers cannot issue duplicate addresses, even if they are both running and each can communicate with only a subset of clients.

Failover configurations are usually in a simple, back office, or symmetrical fashion. Once configured:

1. The partners connect.
2. The main server supplies data about all existing leases to its partner.
3. The backup server requests a pool of backup addresses from the main server.
4. The main server replies with a percentage of available addresses from each scope to its partner.
5. The backup server ignores all DHCPDISCOVER requests, unless it senses that the main server is down. In normal operations, it handles only DHCPRENEW and DHCPREBINDING requests. A DHCPDISCOVER request is a broadcast to locate available servers.
6. The main server updates its partner with the results of all client operations.

As of Network Registrar 6.2, you can automatically synchronize the servers in a failover pair. The two servers dynamically rebalance the available leases; if the main server hands out a large portion of its available leases, it can reclaim leases from its partner.

Note Always configure failover on the same interface that the server uses to serve client traffic.

Failover States and Transitions

During normal operation, the failover partners transition between states. They stay in their current state until all the actions for the state transition are completed and, if communication fails, until the conditions for the next state are fulfilled. The states and their transitions are described in Table 19-1.

Table 19-1 Failover States and Transitions

State	Server Action
STARTUP	Tries to contact its partner to learn its state, then transitions to another state after a short time, typically seconds.
NORMAL	Can communicate with its partner. The main and backup servers act differently in this state: • The main server responds to all client requests using its pool. If its partner requests a backup pool, the main server provides it. • The backup server only responds to renewal and rebinding requests. It requests a backup pool from the main server.

Table 19-1 Failover States and Transitions (continued)

State	Server Action
COMMUNICATIONS-INTERRUPTED	Cannot communicate with its partner, whether it or the communication with it is down. The servers cycle between this state and NORMAL state as the connection fails and recovers, or as they cycle between operational and nonoperational. During this time, the servers cannot give duplicate addresses. During this state, you usually do not need to intervene and move a server into the PARTNER-DOWN state. However, this is not practical in some cases. A server running in this state is not using the available pool efficiently. This can restrict the time a server can effectively service clients. A server is restricted in COMMUNICATIONS-INTERRUPTED state: • It cannot reallocate an expired address to another client. • It cannot offer a lease or renewal beyond the maximum client lead time (MCLT) longer than the current lease time. The MCLT is a small additional time added that controls how much the client lease expiration is ahead of what the backup server thinks it is. • A backup server can run out of addresses to give new clients, because it normally has only a small pool, while the main server has most of them. The server is limited only by the number of addresses allocated to it and the arrival rate of DHCPDISCOVER or INIT-REBOOT packets for new clients. With a high new client arrival or turnover rate, you may need to move the server into PARTNER-DOWN state more quickly.
PARTNER-DOWN	Acts as if it were the only operating server, based on one of these facts: • The partner notified it during its shutdown. • The administrator put the server into PARTNER-DOWN state. • The safe period expired and the partner automatically went into this state.
PARTNER-DOWN (continued)	In this state, the server ignores that the other server might still operate and could service a different set of clients. It can control all its addresses, offer leases and extensions, and reallocate addresses. The same restrictions to servers in COMMUNICATIONS-INTERRUPTED state do not apply. Either server can be in this state, but only one should be in it at a time so that the servers do not issue duplicate addresses and can properly resynchronize later on. Until then, an address is in a pending-available state.
POTENTIAL-CONFLICT	Might be in a situation that does not guarantee automatic reintegration, and is trying to reintegrate with its partner. The server might determine that two clients (who might not be operating) were offered and accepted the same address, and tries to resolve this conflict.
RECOVER	Has no data in its stable storage, or is trying to reintegrate after recovering from PARTNER-DOWN state, from which it tries to refresh its stable storage. A main server in this state does not immediately start serving leases again. Because of this, do not reload a server in this state.
RECOVER-DONE	Can transition from RECOVER or PARTNER-DOWN state, or from COMMUNICATIONS-INTERRUPTED into NORMAL state.

Table 19-1 Failover States and Transitions (continued)

State	Server Action
PAUSED	Can inform its partner that it will be out of service for a short time. The partner then transitions to COMMUNICATIONS-INTERRUPTED state and begins servicing clients.
SHUTDOWN	Can inform its partner that it will be out of service for a long time. The partner then transitions to PARTNER-DOWN state to take over completely.

Allocating Addresses Through Failover

To keep your failover pair operating in spite of a network partition, in which both can communicate with clients but not with each other, you must allocate more addresses than are needed to run a single server. Configure the main server to allocate a percentage of the currently available (unassigned) addresses in each scope address pool to its partner. These addresses become unavailable to the main server. The partner uses them when it cannot talk to the main server and does not know if it is down.

How many additional addresses are needed? There is no single percentage for all environments. It depends on the arrival rate of new DHCP clients and the reaction time of your network administration staff. The backup server needs enough addresses from each scope to satisfy the requests of all new DHCP clients that arrive during the period in which the backup does not know if the main server is down.

Even during PARTNER-DOWN state, the backup server waits for the lease expiration and the maximum client lead time (MCLT), a small additional time buffer, before reallocating any leases. When these times expire, the backup server offers:

- Leases from its private pool of addresses.
- Leases from the main server pool of addresses.
- Expired leases to new clients.

During the day, if the administrative staff can respond within two hours to a COMMUNICATIONS INTERRUPTED state to determine if the main server is working, the backup server needs enough addresses to support a reasonable upper bound on the number of new DHCP clients that might arrive during those two hours.

During off hours, if the administrative staff can respond within 12 hours to the same situation, and considering that the arrival rate of previously unheard from DHCP clients is also less, the backup server then needs enough addresses to support a reasonable upper bound on the number of DHCP clients that might arrive during those 12 hours.

Consequently, the number of addresses over which the backup server requires sole control would be the greater of the numbers of addresses given out during peak and nonpeak times, expressed as a percentage of the currently available (unassigned) addresses in each scope.

Client-Classes

Assigning classes to clients is an important adjunct to DHCP addressing and addresses quality of service issues. You can use the Network Registrar client and client-class facility to provide differentiated services to users that are connected to a common network. You can group your user community based on administrative criteria, and then ensure that each user receives the appropriate class of service.

Although you can use the Network Registrar client-class facility to control any configuration parameter, the most common uses are for:

- **Lease periods**—How long a set of clients should keep their addresses.
- **IP address ranges**—From which lease pool to assign clients addresses.
- **DNS server addresses**—Where clients should direct their DNS queries.
- **DNS hostnames**—What name to assign clients.
- **Denial of service**—Whether unauthorized clients should be offered leases.

One way to use the client-class facility is to allow visitors access to some, but not all, of your network. For example, when Joe, a visitor to ExampleCo, tries to attach his laptop to the example.com network, Network Registrar recognizes the laptop as being foreign. ExampleCo creates one class of clients known as having access to the entire network, and creates another visitor class with access to a subnet only. If Joe needs more than the standard visitor access, he can register his laptop with the Network Registrar system administrator, who adds him to a different class with the appropriate service.

The following sections describe how DHCP normally processes an address assignment, and then how it would handle it with the client-class facility in effect.

See Also

DHCP Processing Without Client-Classes
DHCP Processing with Client-Classes, page 19-13
Defining Scopes for Client-Classes, page 19-14
Choosing Networks and Scopes, page 19-14

DHCP Processing Without Client-Classes

To understand how you can apply client-class processing, it is helpful to know how the DHCP server handles client requests. The server can perform three tasks:

- Assign an IP address.
- Assign the appropriate DHCP options (configuration parameters).
- Optionally assign a fully qualified domain name (FQDN) and update the DNS server with that name.

The DHCP server:

1. Assigns an address to the client from a defined scope—To choose an address for the client, the DHCP server determines the client subnet, based on the request packet contents, and finds an appropriate scope for that subnet.

   If you have multiple scopes on one subnet or several network segments, which is known as multinetting, the DHCP server may choose among these scopes in a round-robin fashion, or you can change the priority of the scope choice by using the DHCP server address allocation priority feature (see the "Configuring Multiple Scopes Using Allocation Priority" section on page 20-12). After the server chooses a scope, it chooses an available (unassigned) address from that scope:

   a. It assigns DHCP option values from a defined policy. Network Registrar uses policies to group options. There are two types of policies: scope-specific and system default. For each DHCP option the client requests, the DHCP server searches for its value in a defined sequence.

   b. If the scope-specific policy contains the option, the server returns its value to the client and stops searching.

   c. If not found, the server looks in the system default policy, returns its value, and stops searching.

   d. If neither policy contains the option, the server returns no value to the client and logs an error.
   e. The server repeats this process for each requested option.
2. With DNS update in effect, the server assigns an FQDN to the client. If you enabled DNS update, Network Registrar enters the client name and address in the DNS host table. See the "DNS Update" section on page 19-6. The client name can be:
   - Its name as specified in the client lease request (the default value).
   - Its MAC address (hardware address; for example, 00:d0:ba:d3:bd:3b).
   - A unique name using the default prefix *dhcp* or a specified prefix.

DHCP Processing with Client-Classes

When you enable the client-class facility for your DHCP server, the request processing performs the same three tasks of assigning IP addresses, options, and domain names as described in the "DHCP Processing Without Client-Classes" section on page 19-12, but with added capability. The DHCP server:

1. **Considers the client properties and client-class inclusion before assigning an address**—As in regular DHCP processing, the DHCP server determines the client subnet. The server then checks if there is a client-class defined or a MAC address for this client in its database. If there is:
   a. A client-class defined by a client-class lookup ID expression, the client is made a member of this client-class.
   b. No MAC address, it uses the default client. For example, the default client could have its client-class name set to Guest, and that client-class could limit (using options and address selection) what network operations such clients are permitted.
   c. No MAC address and no default client, the server handles the client through regular DHCP processing.
   d. No client-specifier, but a MAC address, the MAC address is converted into a client-specifier. An unknown client is mapped to the default client, if the default client is defined.

   The scopes must have addresses on client-accessible subnets—they must have a scope-selection tag that associates them with a client-class. To assign the same clients to different address pools, you must use separate scopes. For example, a scope would either have a scope-selection tag of Employee or Guest, but not both. In this case, there are two scopes for each subnet—one with the scope-selection tag Employee, and the other with Guest. Each scope has a different associated policy and address range that provides the appropriate access rights for the user group.
2. **Checks for client-class DHCP options**—In regular DHCP processing, the server checks the scope-specific and system default DHCP options. With client-class, it also first checks the client-specific and client-class-specific options.
3. **Provides additional FQDN assignment options**—Beyond the usual name assignment process of using the hostname the client requests, the server can:
   - Provide an explicit hostname that overrides it.
   - Drop the client-requested hostname and not replace it.
   - Synthesize a hostname from the client MAC address.

Defining Scopes for Client-Classes

The motivating factor for using client-classes is often to offer an address from one or another address pool to a client. Another motivating factor might be to provide clients with different option values or lease times. Offering clients addresses from separate pools requires defining more than one scope.

To get more than one scope on a subnet, they must come from the same network segment. Networks are not configured directly in Network Registrar, but are inferred from scope configurations. Scopes become related (end up in the same network):

- **Implicitly**—Two scopes have the same network number and subnet mask. These scopes naturally end up on the same network without explicit configuration.
- **Explicitly**—One scope is marked as a secondary to another. This is required when the scope marked as a secondary has a network and subnet mask unrelated to the primary. An example is putting a set of 10.0.0.0 network addresses on a normal, routable network segment.

When the Network Registrar DHCP server reads the scope configuration from its database, it places every scope in a network, and logs this information. Scopes with the same network number and subnet mask end up on the same network, while a secondary scope ends up on the primary scope network.

Choosing Networks and Scopes

When a DHCP packet arrives, the server determines the address from which it came by:

- Gateway address (*giaddr*), if there was one, for packets sent through a BOOTP relay.
- Interface address of the interface on which the broadcast packet arrived, if the DHCP client is on a network segment to which the DHCP server is also directly connected.

In all cases, the DHCP server determines a network from the gateway or interface address. Then, if the network has multiple scopes, the server determines from which scope to allocate an address to the DHCP client. It always looks for a scope that can allocate addresses to this type of client. For example, a DHCP client needs a scope that supports DHCP, and a BOOTP client needs one that supports BOOTP. If the client is a DHCP client and there are multiple scopes that support DHCP, each with available (unassigned) addresses, the DHCP server allocates an IP address from any of those scopes, in a round-robin manner, or by allocation priority.

Scope-selection tags and client-classes let you configure the DHCP server to allocate IP addresses from:

- One or more scopes on a network to one class of clients.
- A different set of scopes to a different class of clients.

In the latter case, the gateway or interface address determines the network. The client-class capability, through the mechanism of the scope-selection tags, determines the scope on the network to use.



CHAPTER 20

Configuring Scopes and Networks

The Dynamic Host Configuration Protocol (DHCP) is an industry-standard protocol for automatically assigning IP configuration to workstations. DHCP uses a client/server model for address allocation. As administrator, you can configure one or more DHCP servers to provide IP address assignment and other TCP/IP-oriented configuration information to your workstations. DHCP frees you from having to manually assign an IP address to each client. The DHCP protocol is described in RFC 2131. For an introduction to the protocol, see Chapter 19, "Introduction to Dynamic Host Configuration."

This chapter describes how to set up DHCP policies and options. Before clients can use DHCP for address assignment, you must add at least one scope (dynamic address pool) to the server.

See Also

Configuring DHCP Servers
Defining and Configuring Scopes, page 20-3
Managing DHCP Networks, page 20-24

Configuring DHCP Servers

When configuring a DHCP server, you must configure the server properties, policies, and associated DHCP options. Network Registrar needs:

- The DHCP server IP address.
- One or more scopes (see the "Defining and Configuring Scopes" section on page 20-3).

See Also

General Configuration Guidelines, page 20-2
Configuring DHCP Server Interfaces, page 20-2

General Configuration Guidelines

Here are some guidelines to consider before configuring a DHCP server:

- **Separate the DHCP server from secondary DNS servers used for DNS updating**—To ensure that the DHCP server is not adversely affected during large zone transfers, it should run on a different cluster than your secondary DNS servers.
- **Configure a separate DHCP server to run in remote segments of the wide area network (WAN)**—Ensure that the DHCP client can consistently send a packet to the server in under a second. The DHCP protocol dictates that the client receive a response to a DHCPDISCOVER or DHCPREQUEST packet within four seconds of transmission. Many clients, notably early releases of the Microsoft DHCP stack, actually implement a two-second timeout.
- **Lease times**—See the "Guidelines for Lease Times" section on page 22-4.

Configuring DHCP Server Interfaces

To configure the DHCP server, accept the Network Registrar defaults or supply the data explicitly:

- **Network interface**—Ethernet card IP address, which must be static and not assigned by DHCP.
- **Subnet mask**—Identifies the interface network membership. The subnet mask is usually based on the network class of the interface address, in most cases 255.255.255.0.

By default, the DHCP server uses the operating system support to automatically enumerate the active interfaces on the machine and listens on all of them. You can also manually configure the server interface. You should statically configure all the IP addresses assigned to NIC cards on the machine where the DHCP server resides. The machine should not be a BOOTP or DHCP client.

Local Basic or Advanced Web UI

Step 1 Click **Servers** to open the Manage Servers page.

Step 2 Click the Interfaces icon () for the DHCP server to open the Manage DHCP Server Network Interfaces page. This page shows the available network interfaces that you can configure for the server. By default, the server uses all of them.

Step 3 To configure an interface, click the Edit icon () in the Configure column for the interface. This adds the interface to the Configured Interfaces table, where you can edit or delete it.

Step 4 Clicking the name of the configured interface opens the Edit DHCP Server Network Interface page, where you can change the address and ports (in Expert mode) of the interface.

Step 5 Click **Modify Interface** when you are done editing.

Step 6 Click **Return** to return to the Manage Servers page.

CLI Commands

Use **dhcp-interface** to manually control which network interface cards' IP addresses the DHCP server will listen on for DHCP clients. By default, the DHCP server automatically uses all your server network interfaces, so use this command to be more specific about which ones to use.

Defining and Configuring Scopes

This section describes how to define and configure scopes for the DHCP server. A scope consists of one or more ranges of dynamic addresses in a subnet that a DHCP server manages. You must define one or more scopes before the DHCP server can provide leases to clients. (For more on listing leases and defining lease reservations for a scope, see Chapter 22, "Managing Leases.")

See Also

Creating and Applying Scope Templates
Creating Scopes, page 20-9
Getting Scope Counts on the Server, page 20-11
Configuring Multiple Scopes, page 20-11
Editing Scopes, page 20-17
Staged and Synchronous Mode, page 20-18
Configuring Embedded Policies for Scopes, page 20-19
Configuring Multiple Subnets on a Network, page 20-19
Enabling and Disabling BOOTP for Scopes, page 20-20
Disabling DHCP for Scopes, page 20-21
Deactivating Scopes, page 20-21
Setting Scopes to Renew-Only, page 20-21
Setting Free Address SNMP Traps on Scopes, page 20-22
Removing Scopes, page 20-23

Creating and Applying Scope Templates

Scope templates apply certain common attributes to multiple scopes. These common attributes include a scope name based on an expression, policies, address ranges, and an embedded policy option based on an expression (see the "Using Expressions in Scope Templates" section on page 20-4).

Local Advanced and Regional Web UI

Scope templates you add or pull from the local clusters are visible on the List DHCP Scope Templates page. To get there, click **DHCP**, then **Scope Templates**. This functionality is available only to administrators assigned the dhcp-management subrole of the regional central-cfg-admin or local ccm-admin role.

To explicitly create a scope template, click **Add Scope Template** on this page. This opens the Add DHCP Scope Template page, which includes a number of fields and settings. You must give the template at least a name. You can also choose an existing policy for the scope template. The other fields require expression values (see the "Create a Scope Template" section on page 5-33 that describes these fields).

See Also

Using Expressions in Scope Templates, page 20-4
Additional Scope Template Attributes, page 20-7
Editing Scope Templates, page 20-8
Applying Scope Templates to Scopes, page 20-8

CLI Commands

Create a scope template using **scope-template** *name* **create**. For example:

```
nrcmd> scope-template example-scope-template create
```

You can also associate a policy with the scope template:

```
nrcmd> scope-template example-scope-template set policy=examplepolicy
```

Using Expressions in Scope Templates

You can specify expressions in a scope template to dynamically create scope names, IP address ranges, and embedded options when creating a scope. Expressions can include context variables and operations.

Note Expressions are not the same as DHCP extensions. Expressions are commonly used to create client identities or look up clients. Extensions (see Chapter 29, "Using Extension Points") are used to modify request or response packets.

If you apply the template to a scope that already has ranges defined, the address range expression of the scope template is not evaluated for that scope.

Table 20-1 lists the scope expression functions. Note that these functions are not case sensitive.

Table 20-1 Expression Functions

Expression Function	Description
Context Variables	
bcast-addr	Derived from the broadcast address in the subnet, such as 192.168.50.255. Use in any expression field. first-addr Derived from the first address in the subnet, such as the first address in 192.168.50.64/26 is 192.168.50.65. Use in any expression field.
last-addr	Derived from the last address in the subnet, such as the last address in 192.168.50.64/26 is 192.168.50.127. Use in any expression field.
mask-addr	Derived from the network mask address in the subnet, such as 255.255.255.0. Use in any expression field.
mask-count	Derived from the number of bits in the network address of the subnet, such as 24. Use in the Scope Name Expression or Embedded Policy Option Expression field.
naddrs	Derived from the number of IP addresses in the subnet, such as 255. Use in the Scope Name Expression field.
nhosts	Derived number of usable hosts in the subnet, such as 254. Use in any expression field.
subnet	Derived from the IP address and mask of the subnet, such as 192.168.50.0/24. Use in the Scope Name Expression or Embedded Policy Option Expression field.
subnet-addr	Derived from the subnet address, such as 192.168.50.0. Use in any expression field.

Table 20-1 Expression Functions (continued)

Expression Function	Description
template.*attribute*	Attribute of the scope template, such as template.ping-timeout. Use in the Embedded Policy Option Expression field.
Arithmetic Operations (unsigned integer arguments only)	
(**+** *arg1 arg2*)	Adds the two argument values, such as (+ 2 3).
(**–** *arg1 arg2*)	Subtracts the second argument value from the first one, such as with *ping-timeout* defined as 100, (– template.ping-timeout 10) yields 90.
(***** *arg1 arg2*)	Multiplies the values of two arguments.
(**/** *arg1 arg2*)	Divides the value of the first argument by that of the second one (which cannot be zero).
Concatenation Operation	
(**concat** *arg1 ... argn*)	Concatenates the arguments into a string, to be used in the Scope Name Expression field. Examples: With subnet=192.168.50.0/24 and template.ping-timeout=100: `(concat "ISP-" subnet)` --> ISP-192.168.50.0/24 `(concat subnet "-" (+ template.ping-timeout 10))` --> 192.168.50.0/24-110 (concat "ISP-" subnet "-" (+ template.ping-timeout 10)) --> ISP-192.168.50.0/24-110 See also the "Scope Name Expression Example" section on page 20-6.
Create Option Operation	
(**create-option** *opt val*)	Use this operation in the Embedded Policy Option Expression field. It creates a new DHCP option for the scope. The first argument can be an integer or string to represent the option number or name. The second argument can be a string or blob to give the option a value. Examples: `(create-option "domain-name" "example.com")` `(create-option 3 "10.10.10.1")` `(create-option "routers" "10.10.10.1,10.10.10.2,10.10.10.3")` `(create-option "routers" (create-ipaddr subnet 10))` See also the "Embedded Policy Option Expression Example" section on page 20-7.
Create Range Operation	
(**create-range** *start end*)	Use this operation in the Range Expression field. It creates an IP address range for the scope. The first argument is the start of the address range and can be an integer or IP address string. The second argument is the end of the range and can be an integer or IP address string. Do not include the local host or broadcast address determined by the mask (such as 0 and 255 for /24 subnets) in the range. Validation ensures that the range must be in the subnet defined by the template and that the first argument value must be lower than the second. An integer value determines the position of the address in the given subnet. Examples (with subnet=192.168.50.0/26): `(create-range "192.168.50.65" "192.168.50.74")` --> 192.168.50.65 - 192.168.50.74 `(create-range 1 10)` --> 192.168.50.65 - 192.168.50.74 See also the "Range Expression Example" section on page 20-7.

Table 20-1 Expression Functions (continued)

Expression Function	Description
Create IP Operation	
(**create-ipaddr** *net host*)	Use this operation in the Embedded Policy Option Expression or Range Expression fields. It creates an IP address string. The net argument is a string or variable. The host argument is an integer. Example: `(create-ipaddr subnet 4)`
List Operation	
(**list** *oper1 ... opern*)	Arguments must all be create-option or create-range operations. Nesting is not supported. Examples: `(list (create-option "routers" "10.10.10.1")` `(create-option "domain-name" "example.com"))` `(list (create-range 1 5) (create-range 10 20))`

Local Advanced and Regional Web UI

There are three fields on the Add DHCP Scope Template page for which you must specify an expression:

- **Scope Name Expression**—Must return a string
- **Range Expression**—Must return IP addresses
- **Embedded Policy Option Expression**—No requirements

CLI Commands

Use the following **scope-template** command attributes:

- ***scope-name***
- ***ranges-exp***
- ***options-exp***

Scope Name Expression Example

You might want to set an expression so that the template constructs scope names starting with "ISP–" and followed by the subnet of the scope and a derivative of its ping timeout value. You would use the following expression in the Scope Name Expression field:

```
(concat "ISP-" subnet "-" (+ template.ping-timeout 10))
```

The elements of the example expression are:

- **(concat** ...**)**—Concatenation operation, which concatenates all the following values into one value.
- **"ISP–"**—String with which to start the scope name.
- **subnet**—Keyword variable that indicates to use the existing subnet defined for the scope.
- **"–"**—Indicates to include this hyphen to construct the value.
- **(+ template.ping-timeout 10)**—Indicates to add the *ping-timeout* property value for the scope to the number 10.

If the scope subnet happens to be 192.168.50.0/24 and its *ping-timeout* value 100, the resulting constructed scope name would be:

```
ISP-192.168.50.0/24-110
```

Range Expression Example

You might want to set an expression so that the template constructs only certain address ranges for scopes. You can either be explicit about the actual starting and ending addresses, or you can make them relative to the subnet. Here are two ways of requesting relative ranges in the Range Expression field:

```
(create-range first-addr last-addr)
(create-range 1 10)
```

The first **create-range** operation creates the address range based on the first through last usable address in the subnet. For the 192.168.50.0/24 subnet, for example, the address range would be 192.168.50.1 through 192.16850.254. Because the second operation specifies integers instead of full IP addresses, it makes the range relative to the subnet based on its mask. If the template discovers the subnet to be 192.168.50.0/26, it takes the first through tenth address in this subnet, which would be 192.168.50.65 through 192.168.50.74.

In the CLI, you would set the range expression with the second operation as follows:

```
nrcmd> scope-template example-scope-template set ranges-expr=(create-range 1 10)
```

Embedded Policy Option Expression Example

An embedded policy is important because the DHCP server looks at it before it looks at the assigned, named policy of the scope. This is usually where you would set the DHCP options on a scope. You might want to set an expression so that the template constructs DHCP options for the scope embedded policy. Here are some examples:

```
(create-option "domain-name" "example.com")
(create-option 3 "10.10.10.1")
(create-option "routers" (create-ipaddr subnet 10))
```

The first **create-option** operation associates the value example.com with the *domain-name* option for the scope. The second operation associates the address 10.10.10.1 with the *routers* option (number 3). The third operation creates an IP address for the *routers* option based on the tenth address in a subnet.

In the CLI, you would set the policy option expression with the first operation as follows:

```
nrcmd> scope-template example-scope-template set options-expr=(create-option domain-name
example.com)
```

Additional Scope Template Attributes

The optional additional attributes appear in functional categories. For a description of each attribute, click the attribute name to open a help window. For example, you might want to enable dynamic DNS updates for the scope, or set the main and backup DHCP failover servers.

After you complete these fields, click **Add Scope Template**.

Editing Scope Templates

To edit a scope template, click its name on the List DHCP Scope Templates page. The Edit DHCP Scope Template page is essentially the same as the Add DHCP Scope Template page (see the "Creating and Applying Scope Templates" section on page 20-3) except for an additional attribute unset function. Make your changes, then click **Modify Scope Template**.

In the CLI, edit a scope template attribute by using **scope-template** *name* **set** *attribute*. For example:

```
nrcmd> scope-template example-scope-template set policy=default
```

Applying Scope Templates to Scopes

You can apply a scope template to a scope in a few ways.

Caution Be careful applying a scope template to an existing scope. The template overwrites all the scope attributes with its own, which can have a detrimental effect if the scope is active.

Local Advanced Web UI

- **While creating a named scope**—On the List/Add DHCP Scopes page, include the name of the scope, add its subnet and mask, then choose the scope template from the drop-down list. Clicking **Add Scope** creates a scope with the name specified and with the attributes set for the scope template, including the expressions you might have set (see the "Using Expressions in Scope Templates" section on page 20-4). (Note that Basic mode lets you specify a Class of Service, but not apply a scope template.)
- **While creating a scope, derive its name from the template**—If you set a Scope Name Expression for the scope template (see the "Using Expressions in Scope Templates" section on page 20-4) on the Add DHCP Scope Template page, when you add a scope on the List/Add DHCP Scopes page, omit the name of the scope, but add its subnet and mask, then choose the scope template from the Template drop-down list. Clicking **Add Scope** creates a scope with a name synthesized from the scope name expression. If you do not set a scope name expression in the template and apply it to the scope without specifying a name for the scope, you get an error. (Note that Basic mode does not provide this functionality.)
- **After creating a named scope**—On the Edit DHCP Scopes page, scroll to the bottom to find the **Apply Template** button. Choose a preconfigured template from the drop-down list, then click the button. Then click **Modify Scope**. (Be aware of the previous warning that the template attributes overwrite the existing ones of the scope.)

CLI Commands

To apply a template to the scope while creating the scope, use **scope** *name* **create** *address mask* **template=***template-name*. For example:

```
nrcmd> scope example-scope create 192.168.50.0 24 template=example-scope-template
```

To derive the scope name from the template during scope creation, use **scope-template** *name* **apply-to** {**all** | *scope1*,*scope2*,...}. For example:

```
nrcmd> scope-template example-scope-template apply-to examplescope-1,examplescope-2
```

Cloning a Scope Template

In the CLI, you can also clone a scope template from an existing one by using **scope-template** *clone-name* **create clone=***template*, and then make adjustments to the clone. For example:

```
nrcmd> scope-template cloned-template create clone=example-scope-template-1
ping-timeout=200
```

Creating Scopes

Creating scopes is a local cluster function. Each scope needs to have the following:

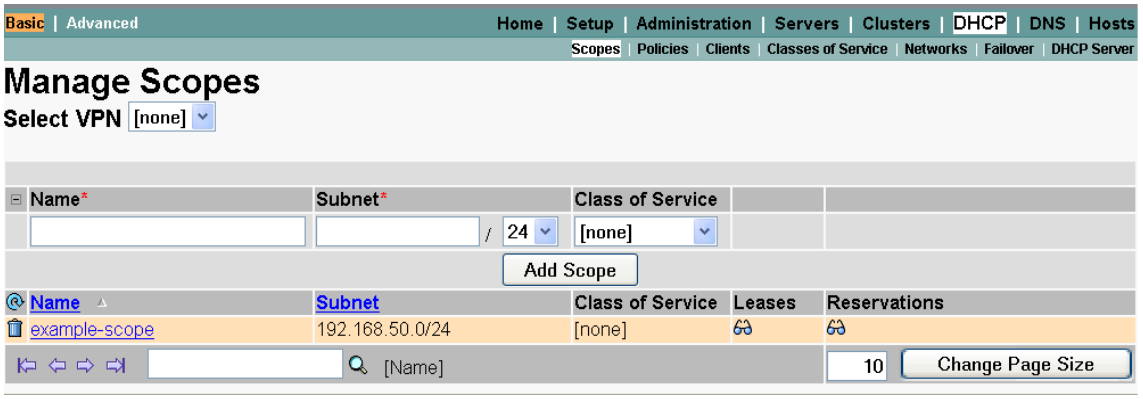
- Name
- Policy that defines the lease times, grace period, and options
- Network address and subnet mask
- Range or ranges of addresses

You can configure scopes at the local cluster only. The web UI pages are different for local basic and advanced modes.

Local Basic Web UI

Step 1 Click **DHCP**, then **Scopes** to open the Manage Scopes page (see Figure 20-1).

Figure 20-1 Manage Scopes Page (Local Basic)



Step 2 Choose a VPN for the scope, if necessary.

Step 3 Enter a scope name, enter the subnet IP address and choose a mask value from the drop-down list.

Step 4 If desired, choose a preconfigured class of service (client-class) for the scope from the drop-down list.

Step 5 Click **Add Scope**.

Step 6 Reload the DHCP server.

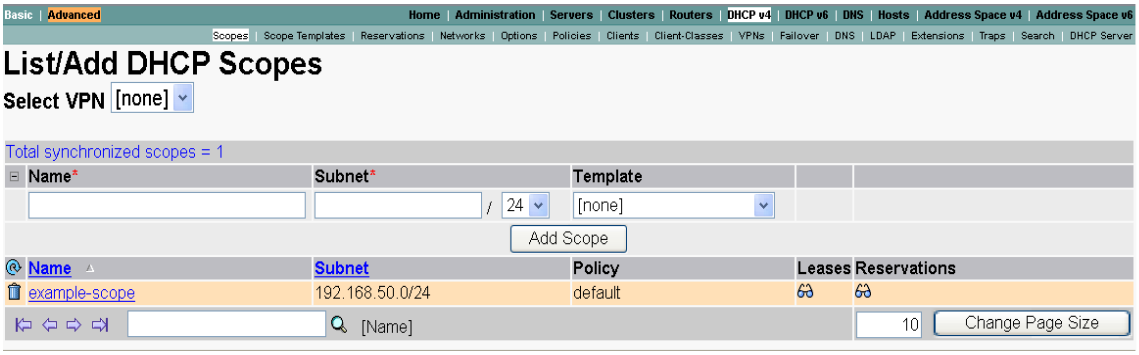
Local Advanced Web UI

Step 1 Click **DHCP**, then **Scopes** to open the List/Add DHCP Scopes page (see Figure 20-2 on page 20-10).

Step 2 Choose a VPN for the scope, if necessary.

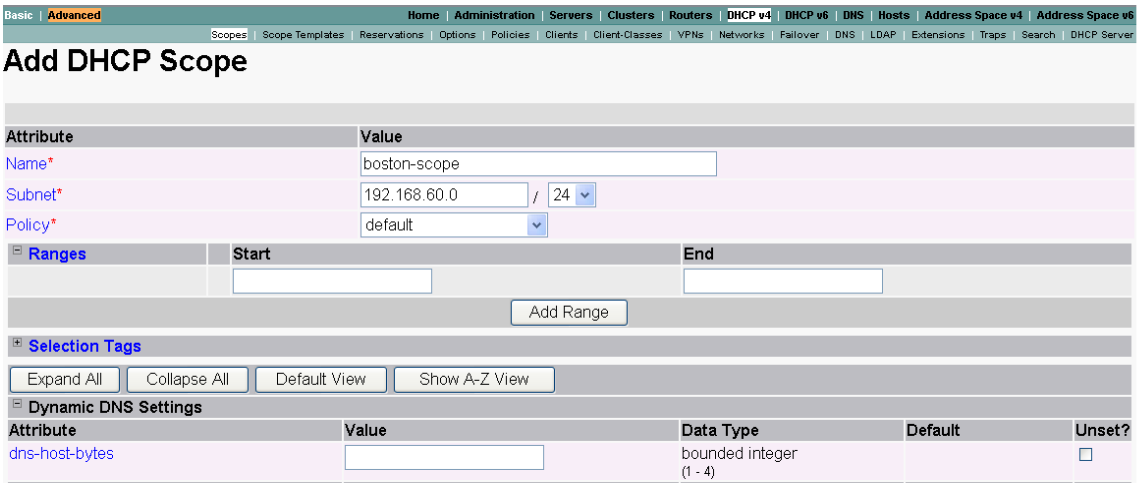
Step 3 Enter a scope name, or leave it blank to use the one defined in the scope name expression of a scope template, if any (see the "Using Expressions in Scope Templates" section on page 20-4). In the latter case, choose the scope template. You must always enter a subnet and mask for the scope.

Figure 20-2 List/Add DHCP Scopes Page (Local Advanced)



Step 4 Click **Add Scope**. This opens the Add DHCP Scope page (see Figure 20-3 for a partial view).

Figure 20-3 Add DHCP Scope Page (Local Advanced)



Step 5 Choose a policy for the scope from the drop-down list. The policy defaults to the *default* policy.

Step 6 Add ranges for addresses in the scope. The ranges can be any subset of the defined scope, but cannot overlap. If you enter just the host number, the range is relative to the netmask. Do not enter ranges that include the local host or broadcast addresses (usually 0 and 255). Add the range, then click **Add Range**.

Step 7 Click **Add Scope**.

Step 8 Reload the DHCP server.

Tip To view any leases and reservations associated with the scope, see Chapter 22, "Managing Leases." To search for leases, see the "Searching Server-Wide for Leases" section on page 22-9.

See Also

Getting Scope Counts on the Server, page 20-11
Configuring Multiple Scopes, page 20-11
Editing Scopes, page 20-17
Staged and Synchronous Mode, page 20-18

Getting Scope Counts on the Server

You can view the created scopes associated with the DHCP server, hence obtain a count, in the web UI.

CLI Commands

Using the CLI, you can get an exact count of the total scopes for the DHCP server by using **dhcp getScopeCount** [**FailoverPair** *name* | **vpn** *name* | **all**]. You can specify a VPN or all VPNs. Omitting the **vpn** *name* returns a count for the current VPN. Specifying a failover pair name returns the total scopes and networks for the failover pair. Because a failover pair definition includes explicit VPN settings in its matchlist, these counts are not limited to the current VPN only.

To create a scope, use **scope** *name* **create**. Each scope must identify its network address and mask. When you create the scope, Network Registrar places it in its current virtual private network (VPN), as defined by **session set current-vpn**.

To explicitly set the scope VPN, use **scope** *name* **set vpn-id**. The VPN must already exist before you can set it for the scope.

To set a policy for the scope, use **scope** *name* **set policy**.

To add a range of IP addresses to the scope, use **scope** *name* **addRange**.

Configuring Multiple Scopes

You can configure multiple scopes (with disjointed address ranges) with the same network number and subnet mask. By default, the DHCP server pools the available leases from all scopes on the same subnet and offers them, in a round-robin fashion, to any client that requests a lease. However, you can also bypass this round-robin allocation by setting an allocation priority for each scope (see the "Configuring Multiple Scopes Using Allocation Priority" section on page 20-12).

Configuring the addresses of a a single subnet into multiple scopes helps to organize the addresses in a more natural way for administration. Even though you can configure a virtually unlimited number of leases per scope, if you have a scope with several thousand leases, it can take a while to sort them. This can be a motivation to divide the leases among multiple scopes.

You can divide the leases among the scopes according to the types of leases. Because each scope can have a separate reservations list, you can put the dynamic leases in one scope that has a policy with one set of options and lease times, and all the reservations in another scope with different options and times. Note that in cases where some of the multiple scopes are not connected locally, you should configure the router (having BOOTP relay support) with the appropriate helper address.

See Also

Configuring Multiple Scopes for Round-Robin Address Allocation
Configuring Multiple Scopes Using Allocation Priority

Configuring Multiple Scopes for Round-Robin Address Allocation

By default, the DHCP server searches through the multiple scopes in a round-robin fashion. Because of this, you would want to segment the scopes by the kind of DHCP client requests made. When multiple scopes are available on a subnet through the use of secondary scopes, the DHCP server searches through all of them for one that satisfies an incoming DHCP client request. For example, if a subnet has three scopes, only one of which supports dynamic BOOTP, a BOOTP request for which there is no reservation is automatically served by the one supporting dynamic BOOTP.

You can also configure a scope to disallow DHCP requests (the default is to allow them). By using these capabilities together, you can easily configure the addresses on a subnet so that all the DHCP requests are satisfied from one scope (and address range), all reserved BOOTP requests come from a second one, and all dynamic BOOTP requests come from a third. In this way, you can support dynamic BOOTP while minimizing the impact on the address pools that support DHCP clients.

Configuring Multiple Scopes Using Allocation Priority

As of Network Registrar Release 6.1, you can set an allocation priority among scopes instead of the default round-robin behavior described in the previous section. In this way, you can have more control over the allocation process. You can also configure the DHCP server to allocate addresses contiguously from within a subnet and control the blocks of addresses allocated to the backup server when using DHCP server failover (see Chapter 27, “Configuring DHCP Failover”).

A typical installation would set the allocation priority of every scope by using the *allocation-priority* attribute on the scope. Some installations might also want to enable the *allocate-first-available* attribute on their scopes, although many would not. There is a small performance loss when using *allocate-first-available*, so you should only use it when absolutely required.

You can control:

- A hierarchy among scopes of which should allocate addresses first.
- Whether to have a scope allocate the first available address rather than the default behavior of the least recently used one.
- Allocating contiguous and targeted addresses in a failover configuration for a scope.
- Priority address allocation server-wide.
- In cases where the scopes have equal allocation priorities set, whether the server should allocate addresses from those with the most or the least number of available addresses.

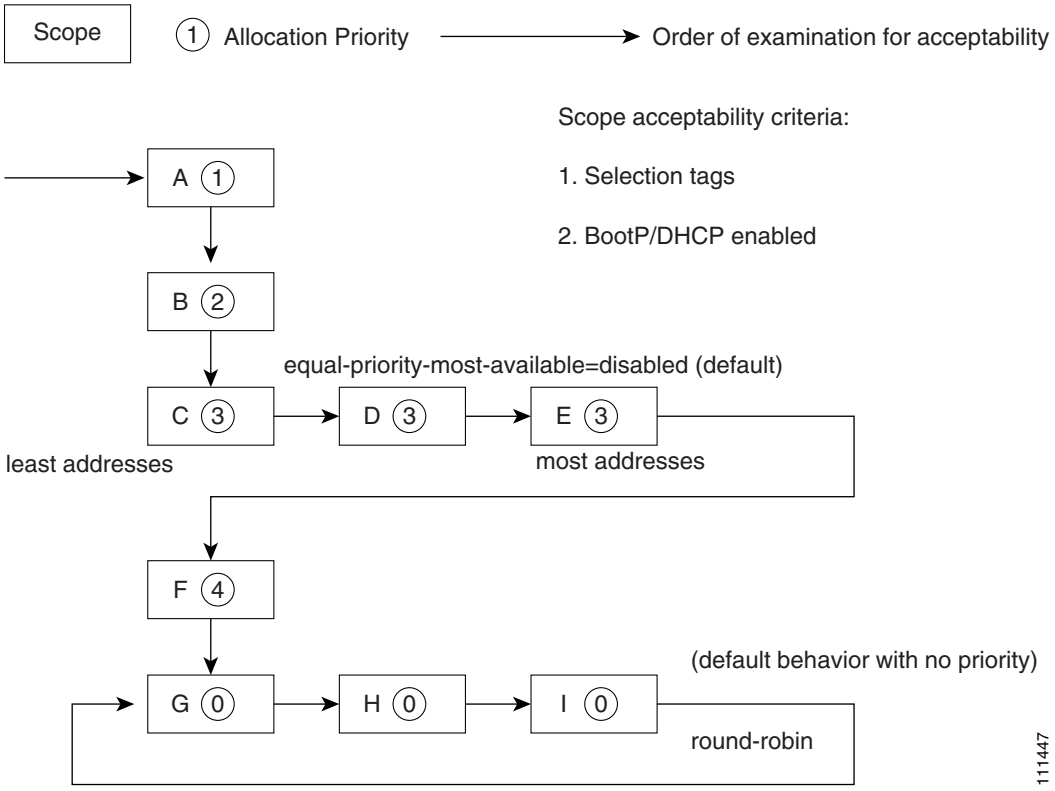
When there is more than one scope in a network, then the DHCP must decide which scope to allocate an IP address from when it processes a DHCPDISCOVER request from a DHCP client that is not already associated with an existing address. The algorithm that the DHCP server uses to perform this allocation is described in the following section.

Allocation Priority Algorithm

The DHCP server examines the scopes in a network one at a time to determine if they are acceptable. When it finds an acceptable scope, it tries to allocate an IP address from it to fulfill the DHCPDISCOVER request. The *allocation-priority* scope attribute is used to direct the DHCP server to examine the scopes in a network in a particular order, because in the absence of any allocation priority, the DHCP server examines the scopes in a round-robin order.

Figure 20-4 shows an example of a network with nine scopes (which is unusual, but serves to illustrate several possibilities of using allocation priority).

Figure 20-4 Scope Allocation Priority



Six of these scopes were configured with an allocation priority, and three of them were not. The server examines the six that were configured with an allocation priority first, in lowest to highest priority order. As the server finds an acceptable scope, it tries to allocate an IP address from it. If the server succeeds, it then finishes processing the DHCPDISCOVER request using this address. If it cannot allocate an address from that scope, it continues examining scopes looking for another acceptable one, and tries to allocate an address from it.

This process is straightforward if no scopes have the same allocation priority configured, but in the case where (as in the example in Figure 20-4) more than one scope has the same nonzero allocation priority, then the server has to have a way to choose between the scopes of equal priority. The default behavior is to examine the scopes with equal priority starting with the one with the fewest available addresses. This uses up all of the addresses in one scope before using any others from another scope. This is the situation shown in Figure 20-4. If you enable the *equal-priority-most-available* DHCP server attribute,

then the situation is reversed and the scope with the most available addresses is examined first when two scopes have equal priority. This spreads out the utilization of the scopes, and more or less evenly distributes the use of addresses across all of the scopes with equal allocation priority set.

You can use this *equal-priority-most-available* approach because of another feature in the processing of equal priority scopes. In the situation where there are two scopes of equal priority, if the DHCPDISCOVER request, for which the server is trying to allocate an address, also has a *limitation-id* (that is, it is using the option 82 limitation capability; see the "Subscriber Limitation Using Option 82" section on page 24-14), then the DHCP server tries to allocate its IP address from the same scope as that used by some existing client with the same *limitation-id* (if any). Thus, all clients with the same *limitation-id* tend to get their addresses allocated from the same scope, regardless of the number of available addresses in the scopes of equal priority or the setting of the *equal-priority-most-available* server attribute.

To bring this back to the *equal-priority-most-available* situation, you might configure *equal-priority-most-available* (and have several equal priority scopes), and then the first DHCP client with a particular *limitation-id* would get an address from the scope with the most available addresses (since there are no other clients with that same *limitation-id*). Then all of the subsequent clients with the same *limitation-id* would go into that same scope. The result of this configuration is that the first clients are spread out evenly among the acceptable, equal priority scopes, and the subsequent clients would cluster with the existing ones with the same *limitation-id*.

If there are scopes with and without allocation priority configured in the same network, all of the scopes with a nonzero allocation priority are examined for acceptability first. Then, if none of the scopes were found to be acceptable and also had an available IP address, the remaining scopes without any allocation priority are processed in a round-robin manner. This round-robin examination is started at the next scope beyond the one last examined in this network, except when there is an existing DHCP client with the same *limitation-id* as the current one sending the DHCPDISCOVER. In this case, the round-robin scan starts with the scope from which the existing client IP address was drawn. This causes subsequent clients with the same *limitation-id* to draw their addresses from the same scope as the first client with that *limitation-id*, if that scope is acceptable and has available IP addresses to allocate.

Address Allocation Attributes

The attributes that correspond to address allocation are described in Table 20-2.

Table 20-2 Address Allocation Priority Settings

Attribute	Type	Description
allocation-priority	Scope (set or unset)	If defined, assigns an ordering to scopes such that address allocation takes place from acceptable scopes with a higher priority until the addresses in all those scopes are exhausted. An allocation priority of 0 (the preset value) means that the scope has no allocation priority. A priority of 1 is the highest priority, with each increasing number having a lower priority. You can mix scopes with an allocation priority along with those without one. In this case, the scopes with a priority are examined for acceptability before those without a priority. If set, this attribute overrides the DHCP server *priority-address-allocation* attribute setting. However, if *allocation-priority* is unset and *priority-address-allocation* is enabled, then the allocation priority for the scope is its subnet address. With *allocation-priority* unset and *priority-address-allocation* disabled, the scope is examined in the default round-robin fashion.

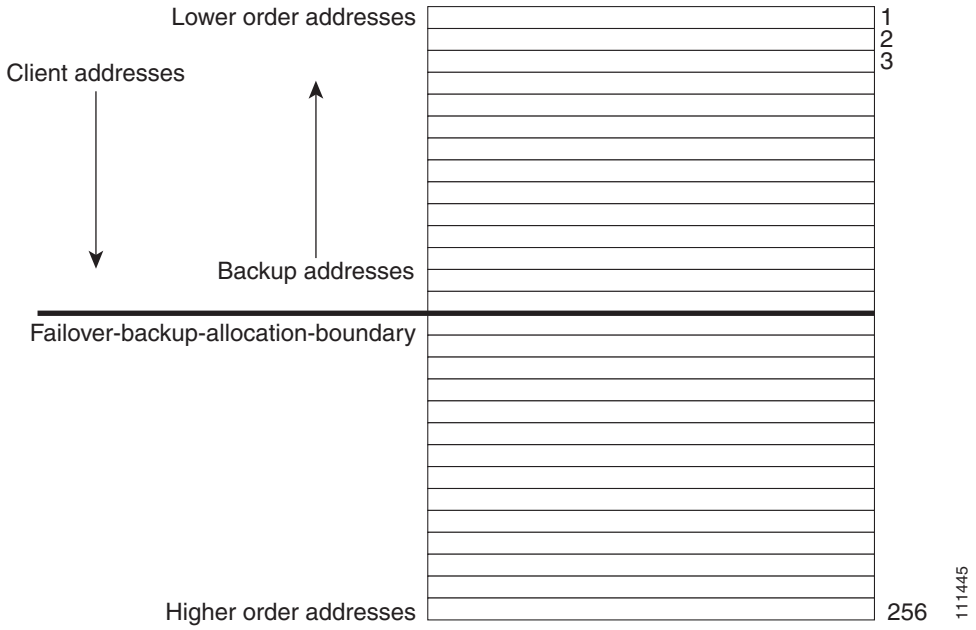
Table 20-2 Address Allocation Priority Settings (continued)

Attribute	Type	Description
allocate-first-available	Scope (enable or disable)	If enabled, forces all allocations for new addresses from this scope to be from the first available address. If disabled (the preset value), uses the least recently used address. If set, this attribute overrides the DHCP server *priority-address-allocation* attribute setting. However, if unset and *priority-address-allocation* is enabled, then the server still allocates the first available address. With *allocate-first-available* unset and *priority-address-allocation* disabled, the scope is examined in the default round-robin fashion.
failover-backup-allocation-boundary	Scope (set or unset)	If *allocate-first-available* is enabled and the scope is in a failover configuration, this value is the IP address to use as the point from which to allocate addresses to a backup server. Only addresses below this boundary are allocated to the backup server. If there are no available addresses below this boundary, then the addresses above it are allocated to the backup server. The actual allocation works down from this address, while the normal allocation for DHCP clients works up from the lowest address in the scope. If this attribute is unset or set to zero, then the boundary used is halfway between the first and last addresses in the scope ranges. If there are no available addresses below this boundary, then the first available address is used. See Figure 20-5 on page 20-16 for an illustration of how addresses are allocated in a scope using this setting.
priority-address-allocation	DHCP (enable or disable)	Provides a way to enable priority address allocation for the entire DHCP server without having to configure it on every scope. (However, the scope *allocation-priority* setting overrides this one.) If *priority-address-allocation* is enabled and the scope *allocation-priority* attribute is unset, then the scope subnet address is used for the allocation priority. If the scope *allocate-first-available* is unset, then priority address allocation is considered enabled. Of course, when exercising this overall control of the address allocation, the actual priority of each scope depends only on its subnet address, which may or may not be desired.
equal-priority-most-available	DHCP (enable or disable)	By default, when two or more scopes with the same nonzero *allocation-priority* are encountered, the scope with the least available IP addresses is used to allocate an address for a new client (if that client is not in a limitation list). If *equal-priority-most-available* is enabled and two or more scopes have the same nonzero allocation priority, then the scope with the most available addresses is used to allocate an address for a new client (if that client is not in a limitation list). In either case, if a client is in a limitation-list, then among those scopes of the same priority, the one that contains other clients in the same list is always used.

Allocating Addresses In Scopes

When trying to allocate an IP address from within a scope, the default action of the DHCP server is to try to allocate the least recently used address first, although there are a variety of events that can cause an IP address to be used. Thus, in general, there is no way to predict which IP address within a scope is allocated at a given time. Usually this poses no difficulty, but there are times when a more deterministic allocation strategy is desired. To configure a completely deterministic address allocation strategy, you can enable the *allocate-first-available* attribute on a scope. This causes the available address with the lowest numeric value to be allocated for a DHCP client. Thus, the first client gets the first address in the lowest range, and the second client the second one in that range, and so on. This is shown in Figure 20-5.

Figure 20-5 Address Allocation with allocate-first-available Set



Note that there is some minor performance cost to this deterministic allocation strategy, not so much that you should not use it, but possibly enough so that you should not use it if you do not need it. When using this deterministic allocation strategy approach in a situation where the scope is in a failover relationship, the question of how to allocate the available IP addresses for the backup server comes up on the main server. By default, the address halfway between the lowest and highest ones in the scope becomes the *failover-backup-allocation-boundary*. The available addresses for the backup server are allocated working down from this boundary (if any addresses are available in that direction). If no address is available below this boundary, then the first available one above the boundary is used for the backup server. You can configure the *failover-backup-allocation-boundary* for the scope if you want to have a different address boundary than the halfway point.

You would use a deterministic allocation strategy and configure *allocate-first-available* in situations where you might allocate a scope with a larger number of IP addresses than you were sure you needed. you can later shrink back the ranges in the scope so as to allow moving address space to another network or server. In the nondeterministic approach, the allocated addresses are scattered all over the ranges, and it can be very hard to reconfigure the DHCP clients to free up, say, half of the scope addresses. However, if you configure *allocate-first-available*, then the allocated addresses tend to cluster low in the scope ranges. It is then probably simpler to remove ranges from a scope that does not need them, so that those addresses can be used elsewhere.

Editing Scopes

Tip The web UI indicates the current synchronization status of the scope on the Edit DHCP Scope page. The CLI equivalent is **scope list**, which shows the current synchronization status.

Local Advanced Web UI

Step 1 Create a scope, as described in the "Creating Scopes" section on page 20-9.

Step 2 Reload the DHCP server.

Step 3 Click the name of the scope on the List/Add DHCP Scopes page to open the Edit DHCP Scope page. (If a server reload is required, a status message indicates it and you must reload first before proceeding.)

Step 4 Modify the fields or attributes as necessary.

Step 5 To edit the scope embedded policy, see the "Configuring Embedded Policies for Scopes" section on page 20-19. To list leases for the scope, see the "Viewing Leases" section on page 22-1.

Step 6 Click **Modify Scope**.

Step 7 Reload the DHCP server.

CLI Commands

After you create a scope, look at the properties for all the scopes on the server, use **scope list** (or **scope listnames**, **scope** *name* **show**, or **scope** *name* **get** *attribute*). Then:

- To reset an attribute, use **scope** *name* **set**.
- To enable or disable an attribute, use **scope** *name* **enable** or **scope** *name* **disable**.
- To change the subnet and mask of the scope, use **scope** *name* **change-subnet**.
- To change just the mask, use **scope** *name* **changeMask**. This changes the *primary-mask* attribute on any secondary scopes, iterates over all reservations and ranges, and displays reservations and ranges that now fall outside the scope.

Note Changing a scope subnet and mask may result in a warning that certain address ranges have values outside of the new scope definition.

Changing a mask:

- Changes it on the specified scope.
- Changes the *primary-mask* attribute on any secondary scopes to the specified scope.
- Iterates over all reservations in the scope and displays any that now fall outside the scope. If reservations fall outside the scope, then the command returns `101, Ok with warnings` instead of `100 Ok`.

  - Iterates over all ranges in the scope and displays any that have endpoints that now fall outside the scope. If range endpoints fall outside the scope, then the command returns `101, Ok with warnings` instead of `100 Ok`.
  - While enabling the *delete-orphaned-leases* attribute, at the next DHCP server reload, deletes existing leases that fall outside of the acceptable ranges for this scope and are not in the acceptable ranges of any other scope.

See Also

Staged and Synchronous Mode
Configuring Embedded Policies for Scopes, page 20-19
Configuring Multiple Subnets on a Network, page 20-19
Enabling and Disabling BOOTP for Scopes, page 20-20
Disabling DHCP for Scopes, page 20-21
Deactivating Scopes, page 20-21
Setting Scopes to Renew-Only, page 20-21
Setting Free Address SNMP Traps on Scopes, page 20-22
Removing Scopes, page 20-23

Staged and Synchronous Mode

New scopes or modifications to scopes can be in one of two modes—staged or synchronous:

- **Staged**—New scopes or modifications to existing scopes are written to the database, but not propagated to the DHCP server until the DHCP server is reloaded.
- **Synchronous**—Most new scopes and scope modifications (including deletions) are immediately propagated to the DHCP server (without the need for a reload). Not all scope changes are possible. For example, changing the primary subnet on a scope is not allowed (a reload is required to effect the change). Furthermore, only scope attribute changes can be propagated without a reload. For example, changes to named policies require a DHCP server reload.

If you add or modify a scope while in staged mode and then change the scope edit mode to synchronous, the first change in synchronous mode applies all pending changes for that scope (not just the ones made while in synchronous mode).

Local Basic or Advanced Web UI

To view the current scope edit mode or change the scope edit mode, go to the Home page under Session Settings. If the scope is up to date in the DHCP server, the "Total synchronized scopes" message appears on the List/Add DHCP Scopes page (in Advanced mode) and the "Scope status: synchronized" message appears on the Edit DHCP Scope page (in both modes). If the scope is not up to date, the "Scope *name* status: reload required" message is displayed.

CLI Commands

View the scope edit mode by using **session get scope-edit-mode**, or set the scope edit mode using **session set scope-edit-mode=**{**sync** | **staged**}. To view the scopes that are not synchronized with the DHCP server, use **scope report-staged-edits**. For example:

```
nrcmd> scope report-staged-edits
100 Ok
example-scope: [reload-required]
```

Configuring Embedded Policies for Scopes

When you create a scope, Network Registrar automatically creates an embedded policy for it. However, the embedded policy has no associated properties or DHCP options until you enable or add them. An embedded policy can be useful, for example, in defining the router for the scope. As the "Types of Policies" section on page 21-1 describes, the DHCP server looks at the embedded policy of a scope before it looks at its assigned, named policy.

Note If you delete a scope policy, you remove all of its properties and attributes.

Local Advanced Web UI

Step 1 Create a scope, as described in the "Creating Scopes" section on page 20-9.

Step 2 Click the name of the scope on the List/Add DHCP Scopes page to open the Edit DHCP Scope page.

Step 3 Click **Create New Embedded Policy** to create a new embedded policy, or **Edit Existing Embedded Policy** if there is already an existing one, to open the Edit DHCP Embedded Policy for Scope page.

Step 4 Modify the fields, options, and attributes on this page. If necessary, unset attributes.

Step 5 Click **Modify Embedded Policy**.

CLI Commands

Create a scope first. In the CLI, **scope-policy** uses the same syntax as **policy**, except that it takes the scope name as an argument. Then, to:

- Determine if there are any embedded property values already set for a scope, use **scope-policy** *scope-name* **show**.
- Enable or disable an attribute, use **scope-policy** *name* **enable** or **scope-policy** *name* **disable**.
- Set and unset attributes, use **scope-policy** *name* **set** and **unset**.
- List, set, and unset vendor options (see the "Using Standard Option Definition Sets" section on page 21-10).

Configuring Multiple Subnets on a Network

Network Registrar supports multiple logical subnets on the same network segment, which are also called secondary subnets. With several logical subnets on the same physical network, for example, 192.168.1.0/24 and 192.168.2.0/24, you might want to configure DHCP so that it offers addresses from both pools. By pooling addresses this way, you can increase the available number of leases.

To join two logical subnets, create two scopes, and elect one to be primary and the other to be a secondary. After you configure the secondary subnet, a new client on this physical network gets a lease from one or the other scope on a round-robin basis.

Local Advanced Web UI

Step 1 Create a scope (see the "Creating Scopes" section on page 20-9) that you will make a secondary scope.

Step 2 Click the name of the scope on the List/Add DHCP Scopes page to open the Edit DHCP Scope page.

Step 3 The first attribute under the Leases area of the page is the *Primary Subnet* attribute. Enter the network address of the subnet of the primary scope, thereby making this a secondary scope.

It is common practice for the *primary-subnet* to correspond directly to the network address of the primary scope or scopes. For example, with examplescope1 created in the 192.168.1.0/24 network, associate examplescope2 with it using *primary-subnet*=192.168.1.0/24. (Note that if Network Registrar finds that the defined subnet has an associated scope, it ignores the mask bit definition and uses the one from the primary scope, just in case they do not match.) However, the *primary-subnet* can be a subnet address that does not have a scope associated with it.

Three other properties used in previous versions of Network Registrar denote primary subnet affiliation: *primary-addr*, *primary-mask*, and *primary-scope*. These properties are present to provide backward compatibility, but should not be used in the current release. The *primary-subnet* attribute (both in the web UI and CLI) now sets these properties.

Step 4 Click **Modify Scope**.

Step 5 Restart or reload the server, if in staged scope edit mode.

CLI Commands

To assign the secondary scope to a primary one, use **scope** *name* **set primary-subnet**, then reload the server (if in staged scope edit mode).

To remove the secondary scope, use **scope** *name* **unset primary-subnet**. When setting the *primary-subnet* attribute, include the number bits for the network mask, using slash notation. For example, represent the network 192.168.1.0 with mask 255.255.255.0 as 192.168.1.0/24. The mask bits are important. If you omit them, a /32 mask (single IP address) is assumed.

Enabling and Disabling BOOTP for Scopes

The BOOTstrap Protocol (BOOTP) was originally created for loading diskless computers. It was later used to allow a host to obtain all the required TCP/IP information so that it could use the Internet. Using BOOTP, a host can broadcast a request on the network and get the data required from a BOOTP server. The BOOTP server listens for incoming requests and generates responses from a configuration database for the BOOTP clients on that network. BOOTP differs from DHCP in that it has no concept of lease or lease expiration. All addresses that a BOOTP server allocates are permanent.

You can configure the Network Registrar DHCP server to act like a BOOTP server. In addition, although BOOTP normally requires static address assignments, you can choose either to reserve addresses (and use static assignments) or have addresses dynamically allocated (known as *dynamic BOOTP*).

When you need to move or decommission a BOOTP client, you can reuse its lease simply by forcing lease availability. See the "Forcing Lease Availability" section on page 22-18.

Local Advanced Web UI

On the Edit DHCP Scope page, under BootP Settings, enable the *bootp* attribute for BOOTP, or the *dynamic-bootp* attribute for dynamic BOOTP. They are disabled by default. Then click **Modify Scope**.

CLI Commands

Use **scope** *name* **enable bootp** to enable BOOTP, and **scope** *name* **enable dynamic-bootp** to enable dynamic BOOTP. Reload the DHCP server (if in staged scope edit mode).

Disabling DHCP for Scopes

You can disable DHCP for a scope if you want to use it solely for BOOTP. See the "Enabling and Disabling BOOTP for Scopes" section on page 20-20. You can also temporarily deactivate a scope by disabling DHCP, but deactivation is more often used if you are enabling BOOTP. See the "Deactivating Scopes" section.

Local Advanced Web UI

On the Edit DHCP Scope page, under BootP Settings, disable the *dhcp* attribute and enable the *bootp* attribute. Then click **Modify Scope**.

CLI Commands

Use **scope** *name* **disable dhcp** to disable DHCP. You should also enable BOOTP and reload the server (if in staged scope edit mode).

Deactivating Scopes

You might want to temporarily deactivate all the leases in a scope. To do this, you must disable both BOOTP and DHCP for the scope.

Local Advanced Web UI

On the Edit DHCP Scope page, under Miscellaneous Settings, explicitly enable the *deactivated* attribute. Then click **Modify Scope**.

CLI Commands

Use **scope** *name* **enable deactivated** to disable BOOTP and DHCP for the scope. Reload the DHCP server (if in staged scope edit mode).

Setting Scopes to Renew-Only

You can control whether to allow existing clients to re-acquire their leases, but not to offer any leases to new clients. A renew-only scope does not change the client associated with any of its leases, other than to allow a client currently using an available IP address to continue to use it.

Local Advanced Web UI

On the Edit DHCP Scope page, under Miscellaneous Settings, explicitly enable the *renew-only* attribute. Then click **Modify Scope**.

CLI Commands

Use **scope** *name* **enable renew-only** to set a scope to renew-only.

Setting Free Address SNMP Traps on Scopes

You can set SNMP traps to capture unexpected free address events by enabling the traps and setting the low and high thresholds for a scope. You can also set traps based on networks and selection tags instead of scopes.

When setting the threshold values, it is advisable to maintain a small offset between the low and high values, as described in the "Simple Network Management" section on page 1-3). The offset can be as little as 5%, for example, a low value of 20% and a high value of 25%, which are the preset values.

Here are some variations on how you can set the server and scope values for these attributes:

- Get each scope to trap and reset the free address values based on the server settings, as long as at least one recipient is configured.
- Disable the traps at the scope level or specify different percentages for each scope.
- Disable the traps globally on the server, but turn them on for different scopes.
- Set the traps at the network level or selection tags level.

Local Advanced Web UI

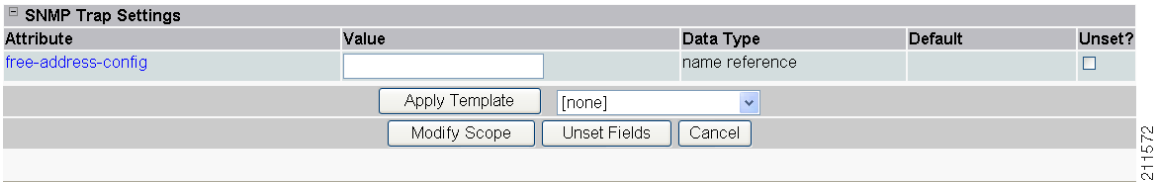
Step 1 Create a trap configuration by clicking **DHCP**, then **Traps** to open the List Trap Configurations page.

Step 2 Click **Add Trap Configuration** to open the Add Trap Configuration page (see Figure 1-3 on page 1-9).

Step 3 Enter a name for the trap configuration, choose **scope** from the mode drop-down list, and enter the low and high threshold values (they are 20% and 25%, respectively, by default). Click **Add Trap Configuration**. (You can go back to edit these values if you need to.)

Step 4 Edit the created scope to which you want to apply the threshold settings. Under SNMP Trap Settings, enter the name of the trap in the *free-address-config* attribute field (see Figure 20-6). Click **Modify Scope**.

Figure 20-6 SNMP Trap Settings on the Edit DHCP Scope Page (Local Advanced)



CLI Commands

Use **addr-trap** *name* **create** to add a trap configuration. To set the thresholds, use the **addr-trap** *name* **set** method (or include the threshold settings while creating the trap). For example:

```
nrcmd> addr-trap trap-1 create
nrcmd> addr-trap trap-1 set low-threshold
nrcmd> addr-trap trap-1 set high-threshold
```

To set the free-address trap, use **scope** *name* **set free-address-config=***trap-name*. For example:

```
nrcmd> scope scope-1 set free-address-config=trap-1
```

Removing Scopes

Caution Although removing a scope from a DHCP server is easy to do, be careful. Doing so compromises the integrity of your network. There are several ways to remove a scope from a server, either by re-using or not re-using addresses, as described in the following sections.

DHCP, as defined by the IETF, provides an address lease to a client for a specific time (defined by the server administrator). Until that time elapses, the client is free to use its leased address. A server cannot revoke a lease and stop a client from using an address. Thus, while you can easily remove a scope from a DHCP server, the clients that obtained leases from it can continue to do so until it expires. This is true even if the server does not respond to their renewal attempts, which happens if the scope was removed.

This does not present a problem if the addresses you remove are not reused in some way. However, if the addresses are configured for another server before the last lease expires, the same address might be used by two clients, which can destabilize the network.

Network Registrar moves the leases on the removed scope to an orphaned leases pool. When creating a scope, orphaned leases are associated with appropriate scopes.

See Also

Removing Scopes if Not Reusing Addresses
Removing Scopes if Reusing Addresses, page 20-24

Removing Scopes if Not Reusing Addresses

To remove scopes if not reusing addresses:

Local Basic or Advanced Web UI

If you are sure you do not plan to reuse the scope, on the Manage Scopes or List/Add DHCP Scopes page, click the Delete icon ( ) next to its name, and confirm or cancel the deletion.

CLI Commands

Be sure that you are not immediately planning to reuse the addresses in the scope, then use **scope** *name* **delete** to delete it.

Removing Scopes if Reusing Addresses

If you want to reuse the addresses for a scope you want to remove, you have two alternatives:

- **If you can afford to wait until all the leases in the scope expire**—Remove the scope from the server, then wait for the longest lease time set in the policy for the scope to expire. This ensures that no clients are using any addresses from that scope. You can then safely reuse the addresses.
- **If you cannot afford to wait until all the leases in the scope expire**—Do not remove the scope. Instead, deactivate it. See the "Deactivating Scopes" section on page 20-21. Unlike a removed scope, the server refuses all clients' renewal requests, which forces many of them to request a new lease. This moves these clients more quickly off the deactivated lease than for a removed scope.

  You can use the **ipconfig** utility in Windows to cause a client to release (**/release**) and re-acquire (**/renew**) its leases, thereby moving it off a deactivated lease immediately. You can only issue this utility from the client machine, which makes it impractical for a scope with thousands of leases in use. However, it can be useful in moving the last few clients in a Windows environment off deactivated leases in a scope.

Managing DHCP Networks

When you create a scope, you also create a network based on its subnet and mask. Scopes can share the same subnet, so that it is often convenient to show their associated networks and the scopes. Managing these networks is a local cluster function only. You can also edit the name of any created network.

See Also

Listing Networks
Editing Networks, page 20-24

Listing Networks

The List Networks page lets you list the networks created by scopes and determine to which scopes the networks relate. The networks are listed by name, which the web UI creates from the subnet and mask. On this page, you can expand and collapse the networks to show or hide their associated scopes.

In Basic mode, click **DHCP** (or in Advanced or Expert mode, click **DHCP**, then **Networks** to open the View Network Tree page (for DHCPv6 networks, this opens the View DHCPv6 Networks page). On this page, you can:

- **List the networks**—The networks appear alphabetically by name and identify their subnet and any assigned scope-selection tags. Click the plus (**+**) sign next to a network to expand the view to show the associated scopes. To expand all the network views, click **Expand All**; to collapse all the network views to show just the network names, click **Collapse All**.
- **Edit a network name**—Click the network name. See the "Editing Networks" section.

Editing Networks

You can edit a network name. The original name is based on the subnet and mask as specified in the scope. You can change this name to an arbitrary but descriptive string.

Local Basic or Advanced Web UI

Step 1 Click **DHCP**, then **Networks** to open the View Network Tree page (DHCPv4) or the View DHCPv6 Networks (DHCPv6).

Step 2 Click the name of the network you want to edit. This opens the Edit Network page.

For DHCPv6, the View DHCPv6 Networks page is for creating networks. Enter a name for the network, choose a template, if desired, and enter the template root prefix name (see the "Viewing DHCPv6 Networks" section on page 26-24.) The DHCPv6 network view is not available in Basic mode.

Step 3 Edit the network data.

Step 4 Click **Modify Network**. For DHCPv6, click **Add Link**.



CHAPTER 21

Configuring Policies and Options

This chapter describes how to set up DHCP policies and options. Before clients can use DHCP for address assignment, you must add at least one DHCPv4 scope (dynamic address pool) or DHCPv6 prefix to the server. The policy attributes and options are assigned to the scope or prefix.

See Also

Configuring DHCP Policies
Creating DHCP Option Definition Sets and Option Definitions, page 21-9

Configuring DHCP Policies

Every DHCPv4 scope or DHCPv6 prefix must have one or more policies defined for it. Policies define lease duration, gateway routers, and other configuration parameters, in what are called DHCP options. Policies are especially useful if you have multiple scopes or prefixes, because you need only define a policy once.

This section describes how you can define named policies with specific attributes and option definitions, or use system default or embedded policies.

See Also

Types of Policies
Policy Hierarchy, page 21-3
Creating and Applying DHCP Policies, page 21-3
Cloning a Policy, page 21-6
Setting DHCP Options and Attributes for Policies, page 21-6
Creating and Editing Embedded Policies, page 21-8

Types of Policies

There are three types of policies—system default, named, and embedded:

- **System default** (**system_default_policy**)—Provides a single location for setting default values on certain options for all scopes or prefixes. Use the system default policy to define attributes and standard DHCP options that have common values for all clients on all the networks that the DHCP server supports. You can modify the system default options and their values. If you delete a system default policy, it reappears using its original list of DHCP options and their system-defined values (see Table 21-1 on page 21-2).

Table 21-1 System Default Policy Option Values

System Default Option	Predefined Value
all-subnets-local	False
arp-cache-timeout	60 seconds
broadcast-address	255.255.255.255
default-ip-ttl	64
default-tcp-ttl	64
dhcp-lease-time	604800 seconds (7d)
ieee802.3-encapsulation	False
interface-mtu	576 bytes
mask-supplier	False
max-dgram-reassembly	576 bytes
non-local-source-routing	False
path-mtu-aging-timeout	6000 seconds
path-mtu-plateau-tables	68, 296, 508, 1006, 1492, 2002, 4352, 8166, 17914, 32000
perform-mask-discovery	False
router-discovery	True
router-solicitation-address	224.0.0.2
tcp-keepalive-garbage	False
tcp-keepalive-interval	0 seconds
trailer-encapsulation	False

- **Named**—Policies you explicitly define by name. Named policies are usually named after their associated scope, prefix, or client grouping. For example, the policy might be assigned attributes and options that are unique to a subnet, such as for its routers, and then be assigned to the appropriate scope or prefix.

  Network Registrar includes a policy named **default** when you install the DHCP server. The server assigns this policy to newly created scopes and prefixes. You cannot delete this default policy.

- **Embedded**—A policy embedded in (and limited to) a named scope, scope template, prefix, prefix template, client, or client-class. An embedded policy is implicitly created (or removed) when you add (or remove) the corresponding object. Embedded policy options have no default values and are initially undefined.



Tip Be sure to save the object (scope, prefix, client, or client-class) for which you are creating or modifying an embedded policy. Not doing so is a common error when using the web UI. Click **Modify** for both the embedded policy and the parent object.

Policy Hierarchy

To eliminate any conflicting attribute and option values that are set at various levels, the Network Registrar DHCP server uses a local priority method. It adopts the more locally defined attribute and option values first while ignoring the ones defined on a more global level, and includes any default ones not otherwise defined. When the DHCP server makes processing decisions for a DHCPv4 client, it prioritizes the attributes and options in this order:

1. Client embedded policy.
2. Client named policy.
3. Client-class embedded policy.
4. Client-class named policy.
5. Scope embedded policy for clients, or address block embedded policy for subnets.
6. Scope named policy for clients (or default policy if a named policy is not applied to the scope), or address block named policy for subnets.
7. Any remaining unfulfilled attributes and options in the system_default_policy. For attributes, the default value for the most local policy applies.

Note For DHCPv6 policy prioritization, see the "DHCPv6 Policy Hierarchy" section on page 26-7.

Creating and Applying DHCP Policies

This section describes how to create a policy at the DHCP server level and then allow specific scopes or prefixes to reference it. A policy can consist of a:

- **Name**—Not case sensitive and must be unique.
- ***permanent-leases* attribute**—A permanent lease never expires.
- **Lease time**—How long a client can use an assigned lease before having to renew the lease with the DHCP server (the lease time attributes are not available for an embedded policy, only the option). The default lease time for both system default and default policies is seven days (604800 seconds). A policy contains two lease times—the client lease time and the server lease time:
  - **Client lease time**—Determines how long the client believes its lease is valid. (Set the client lease time using a DHCP option, not a policy attribute.)
  - **Server lease time**—Determines how long the server considers the lease valid. Note that the server lease time is independent of the lease grace period. The server does not allocate the lease to another client until after the lease time and grace period expire.

Caution Although Network Registrar supports the use of two lease times for special situations, Cisco Systems generally recommends that you not use the *server-lease-time* attribute.

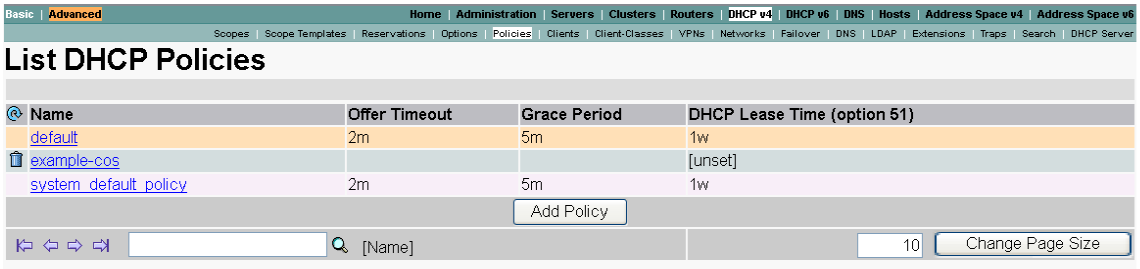
You can establish these two different lease times if you want to retain information about client DNS names and yet have them renew their leases frequently. When you use a single lease time and it expires, the server no longer keeps that client DNS name. However, if you use a short client lease time and a longer server lease time, the server retains the client information even after the client lease expires. (For details on leases, see Chapter 22, "Managing Leases.")

- **Lease grace period**—Time period after the lease expires that it is unavailable for reassignment (not available for an embedded policy).
- **DNS update configuration**—A DNS update configuration specifies the type of DNS updates to perform, the zones involved, the DNS server to be updated, and the related security. The policy determines the forward and reverse DNS update configuration objects, and can also specify the forward zone to use if a DNS server hosts multiple zones. (For details on DNS update configurations, see the "Creating DNS Update Configurations" section on page 28-5.)
- **DHCP options**—To add option values, see the "Setting DHCP Options and Attributes for Policies" section on page 21-6. See also Appendix B, "DHCP Options," for the supported option types.

Local Basic or Advanced and Regional Web UI

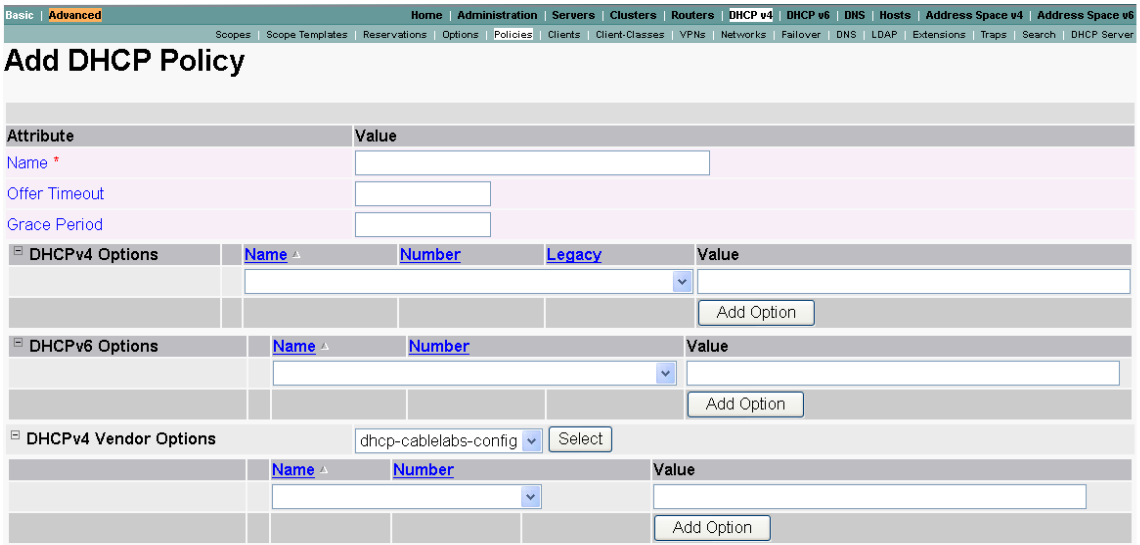
Step 1 Click **DHCP**, then **Policies** to open the List DHCP Policies page (see Figure 21-1 for a DHCPv4 example in local Advanced mode). (DHCPv6 is not available in Basic mode.)

Figure 21-1 List DHCP Policies Page (Local Advanced)



Step 2 The default policy and system_default_policy are already provided for you. To add a named policy, click **Add Policy** to open the Add DHCP Policy page (see Figure 21-2 for a partial view).

Figure 21-2 Add DHCP Policy Page (Local Advanced)



Step 3 Give the policy a unique name (required).

Step 4 Set the offer timeout and grace period values or leave them blank.

Step 5 Add the necessary DHCP options (see the "Setting DHCP Options and Attributes for Policies" section on page 21-6):

- **Lease time**—Set the *dhcp-lease-time (51)* option.
- **Subnet mask**—Set the *subnet-mask (1)* option, but also enable the *get-subnet-mask-from-policy* attribute for the DHCP server. To remove the subnet mask from the policy, either unset the attribute or disable it.

To set vendor-specific options, see the "Using Standard Option Definition Sets" section on page 21-10.

Step 6 Set the policy attributes, which include:

- **Unavailable timeout**—See the "Setting Timeouts for Unavailable Leases" section on page 22-20.
- **Inhibit all renews**—See the "Inhibiting Lease Renewals" section on page 22-18.
- **Limitation count**—See the "Using Expressions" section on page 25-1.
- **Use client IDs for reservations**—See the "Overriding Client IDs" section on page 22-16.
- **Permanent leases** (not recommended).
- **DNS update settings**—To set the DNS update configuration that determines which forward or reverse zones you want to include in a DNS update, set the following attributes:
  - ***forward-dnsupdate***—Name of the update configuration for the forward zone. Note that you can thereby set different update configurations for forward and reverse zones.
  - ***forward-zone-name***—If necessary, overrides the forward zone in the update configuration. Use this in case a DNS server is hosting multiple zones.
  - ***reverse-dnsupdate***—Name of the update configuration for the reverse zone. If not set on any policy in the policy hierarchy applicable to the client request (see the "Policy Hierarchy" section on page 21-3), the DHCP server uses the *forward-dnsupdate* configuration.

Step 7 Click **Add Policy**.

Step 8 Reload the DHCP server.

Step 9 In the regional web UI, you can also pull replica policies and push policies to local clusters. (See the "Managing DHCP Policies" section on page 6-13 for regional policy management.)

CLI Commands

Use **policy** *name* **create** to create the policy. Then use **policy** *name* **set offer-timeout=***value* and **policy** *name* **set grace-period=***value* to set these two values.

To set policy options, use **policy** *name* **setOption**:

- **Lease time**—Use **policy** *name* **setLeaseTime**.
- **Subnet mask**—Use a combination of **policy** *name* **setOption subnet-mask** *value* and **dhcp enable get-subnet-mask-from-policy**.

To confirm the option settings, use **policy** *name* **listOptions** or **policy** *name* **getOption**.

To enable permanent leases (not recommended), use **policy** *name* **enable permanent-leases**. Note that enabling permanent leases forces the *dhcp-lease-time* option (51) to be set to infinite.

See Also

Types of Policies, page 21-1
Policy Hierarchy, page 21-3
Cloning a Policy, page 21-6
Setting DHCP Options and Attributes for Policies, page 21-6
Creating and Editing Embedded Policies, page 21-8
Creating DHCP Option Definition Sets and Option Definitions, page 21-9

Cloning a Policy

In the CLI, you can clone a policy from an existing one by using **policy** *clone-name* **create clone=***policy*, and then make adjustments to the clone. For example:

```
nrcmd> policy cloned-policy create clone=example-policy-1 offer-timeout=4m
```

Setting DHCP Options and Attributes for Policies

DHCP options automatically supply DHCP clients with configuration parameters, such as domain, nameserver, and subnet router addresses (see the "Creating DHCP Option Definition Sets and Option Definitions" section on page 21-9). Note that the Network Registrar user interfaces allow you to set some option values on a policy that actually have no effect on the packet returned to the client (such as *hostname* and *dhcp-server-identifier*).

The server searches the policies, in order, for these BOOTP and DHCP attribute values and returns the first occurrence of these values in its reply packet:

- *packet-siaddr* returned in the *siaddr* packet field
- *packet-file-name* returned in the *file* field
- *packet-server-name* returned in the *sname* field

See Also

Adding Option Values

Adding Complex Values for Suboptions, page 21-7

Adding Option Values

You can view, set, unset, and edit DHCP option values. When you set an option value, the DHCP server replaces any existing value or creates a new one, as needed for the given option name. Network Registrar DHCP options are grouped into categories to aid you in identifying options that you must set in various usage contexts (Table B-10 on page B-14 describes the categories). You can create custom option definitions to simplify entering custom option values (see the "Creating Custom Option Definitions" section on page 21-12).

Local Basic or Advanced and Regional Web UI

Step 1 Create a policy, as described in the "Creating and Applying DHCP Policies" section on page 21-3.

Step 2 On the Add DHCP Policy or Edit DHCP Policy page, add each DHCP option to the policy by clicking its number and name in the drop-down list. The choices indicate the data type of the option value (see the "Option Definition Data Types and Repeat Counts" section on page 21-14).

Tip You can sort the options by Name, Number, or (in the case of DHCPv4) Legacy (grouping).

Step 3 Add the appropriate option value in the Value field. The web UI does error checking based on the value entered. For example, to add the lease time for the policy, click the *[51] dhcp-lease-time (unsigned time)* option in the Number drop-down list, then add a lease time value in the Value field. (Options do not have preset values.)

Tip If you are configuring an option on a policy while another user is editing the option definition, log out of the session and log back in to get the new option definition.

Step 4 Click **Add Option** for each option. You must supply a value or you cannot add the option.

Step 5 Click **Add Policy**.

Tip If you add new option values or edit existing ones, be sure to save the policy object by clicking **Modify Policy**.

CLI Commands

To view option values, use **policy** *name* **getOption** and **policy** *name* **listOptions**. To set option values, use **policy** *name* **setOption** *option*. When you set an option value, the DHCP server replaces any existing value or creates a new one, as needed, for the given option name. To unset option values, use **policy** *name* **unsetOption**.

Adding Complex Values for Suboptions

If you are adding more complex option values such as for suboptions, use a parenthesized string format. The format requires that you:

- Enclose each option level (option, suboption, subsuboption) in parentheses.
- Separate multiple values with commas.
- Separate data fields for packed data (missing the suboption code or length) with semicolons.

For example, the *cablelabs-client-configuration* option (122) normally has 10 suboptions as well as some subsuboptions. This example shows the syntax to set the suboption 1, 2, 3, and 4 data values, and includes the two subsuboptions for suboption 3 and the three subsuboptions for suboption 4 (which are packed data and have no code numbers):

```
(primary-dhcp-server 1 10.1.1.10)
(secondary-dhcp-server 2 10.2.2.10)
(provisioning-server 3 (flag 59; provisioning-server 10:10:10:10))
(as-backoff-retry 4 (as-backoff-retry-initial-time-ms 10;
as-backoff-retry-max-time 10s; as-backoff-retry-count 100))
```

The suboption name (such as primary-dhcp-server) is optional. Hence, it is often safer to use just the code number and data value (or just the data value for packed data) to minimize typographical errors and parsing failures. The compacted (and preferred) version of the previous example that strips out the suboption names is:

```
(1 10.1.1.10) (2 10.2.2.10) (3 (59;10:10:10:10)) (4 (10;10s;100))
```

Even if you use numerical code values, Network Registrar always includes the equivalent names when it displays the suboptions (see the "Creating DHCP Option Definition Sets and Option Definitions" section on page 21-9).

To include suboptions that include enterprise IDs (such as for option 125), use the following format, for example, when entering in the policy option value:

```
(enterprise-id 1((1 10.1.1.1) (2 10.2.2.2) (3 www.cisco.com)))
```

The parentheses surround the enterprise ID itself, the suboptions as a group, and each suboption.

Creating and Editing Embedded Policies

An embedded policy is embedded for a DHCPv4 scope or scope template, DHCPv6 prefix or prefix template, client, or client-class (see Chapter 26, "Managing DHCPv6 Addresses," for embedded policies in DHCPv6). You can create or edit an embedded policy.

Local Advanced Web and Regional UI

Step 1 Click **DHCP**, then one of the following that appear for DHCPv4 or DHCPv6 in the local web UI: **Scopes**, **Scope Templates**, **Clients**, **Client-Classes**, **Prefixes**, or **Links**. (The regional web UI can have the selections **Scope Templates**, **Client-Classes**, **Prefixes**, and **Links**.)

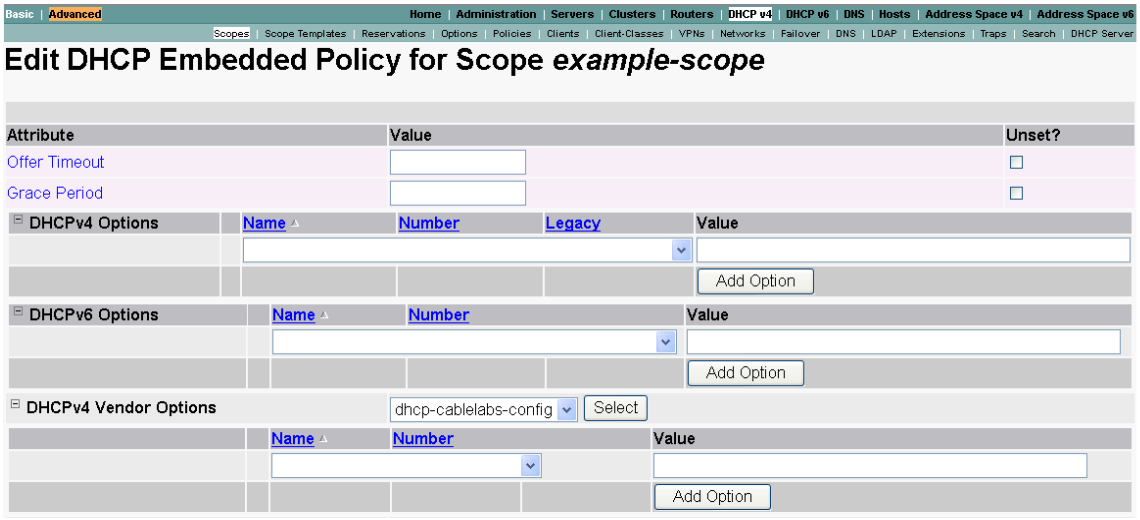
Step 2 Click the name of the object to open its Edit page.

Step 3 Click **Create Embedded Policy** or **Edit Existing Embedded Policy** under the Embedded Policy section of the page. This opens the Edit DHCP Embedded Policy page for the object (see Figure 21-3 on page 21-9 for a partial view of a scope embedded policy page).

Step 4 Make changes to the values as needed, then click **Modify Embedded Policy**.

Step 5 On the Edit page for the object, be sure to save the changes by clicking **Modify**.

Figure 21-3 Edit DHCP Embedded Policy Page (Local Advanced)



CLI Commands

Use the embedded commands, such as **client-class-policy** *client-class-name* **set** *attribute*=*value*, where the command starts with the object name followed by -policy.

Creating DHCP Option Definition Sets and Option Definitions

In Network Registrar, you configure option values on policies for such things as lease times and router addresses. Numerous RFCs describe the formatting of DHCP option values, beginning with RFC 2132. Option definitions are used in the web UI and CLI to control formatting of option values in policies. You can define option definitions separately for the DHCPv4 and DHCPv6 address spaces, as:

- **Standard (built-in) options**—Defined by the RFCs. In the web UI, these are in the **dhcp-config** and **dhcp6-config** definition sets. The CLI includes additional **dhcp-default** and **dhcp6-default** definition sets that are hidden, but accessible if you call for them specifically. (See the "Using Standard Option Definition Sets" section.)
- **Custom options**—New or modified definitions in the supplied **dhcp-config** or **dhcp6-config** definition sets. Once you add or modify definitions in the web UI, they are added to the **dhcp-custom** or **dhcp6-custom** definition sets in the CLI. (See the "Creating Custom Option Definitions" section on page 21-12.)
- **Vendor-specific options**—Defined in their own definition sets. The CableLabs definition sets (**dhcp-cablelabs-config** and **dhcp6-cablelabs-config**) are preconfigured in Network Registrar. The CLI also includes **dhcp-cablelabs-default**, **dhcp6-cablelabs-default**, **dhcp-cablelabs-custom**, and **dhcp6-cablelabs-custom** definition sets. (See the "Using Standard Option Definition Sets" section on page 21-10.)

See Also

Using Standard Option Definition Sets
Creating Custom Option Definitions, page 21-12
Creating Vendor-Specific Option Definitions, page 21-12
Option Definition Data Types and Repeat Counts, page 21-14
Adding Suboption Definitions, page 21-15
Importing and Exporting Option Definition Sets, page 21-15
Pushing Option Definition Sets to Local Clusters, page 21-16
Pulling Option Definition Sets from Replica Data, page 21-16
Setting Option Values for Policies, page 21-17

Using Standard Option Definition Sets

Network Registrar provides two standard, built-in option definition sets, **dhcp-config** and **dhcp6-config**, for DHCPv4 and DHCPv6 option definitions, respectively. You can create new options definitions in these sets or you can overwrite existing ones. New option definitions or ones that were overwritten are identified by an asterisk (*). You can delete these definitions and there is no deletion confirmation given. However, saving the set after deleting an overwritten definition causes the original definition to reappear in the set.

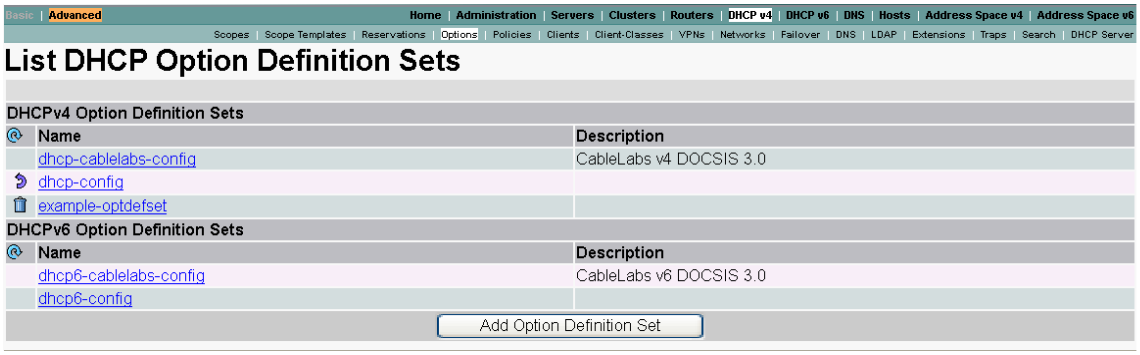
The full list of predefined DHCP option definitions is included in Appendix B, "DHCP Options."

Caution Arbitrarily modifying the standard definitions (or adding suboption definitions) can adversely affect configurations.

Local Advanced and Regional Web UI

Step 1 Click **DHCP**, then **Options** to open the List DHCP Option Definition Sets page (see Figure 21-4). (DHCP option definition is not available in Basic mode.)

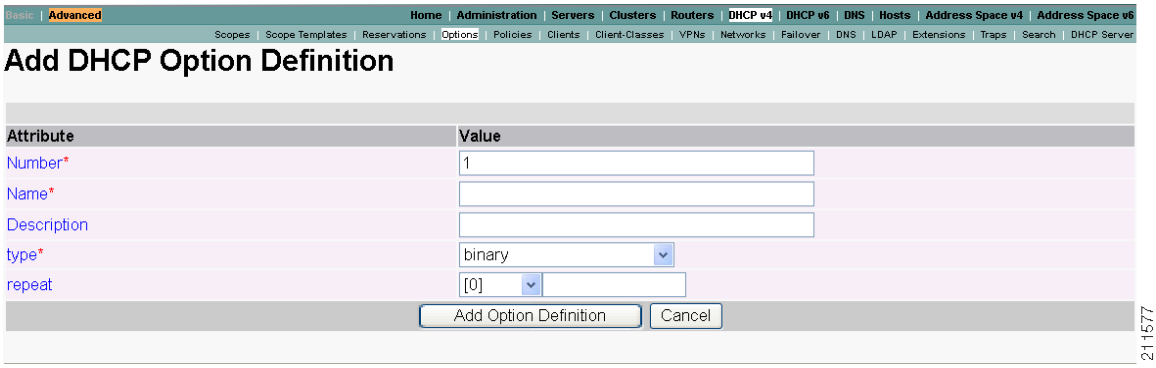
Figure 21-4 List DHCP Option Definition Sets Page (Local Advanced)



Step 2 Click the **dhcp-config** or **dhcp6-config** link to open the Edit DHCP Option Definition Set page, then click **Add/Edit Option Definitions**. View the predefined definitions on the List DHCP Option Definitions page. These are the definitions that control the formatting of the option values you add to policies. If there are suboption definitions, you can expand to show them.

Step 3 To add a definition, click **Add Option Definition**. On the Add DHCP Option Definition page (see Figure 21-5), give the option an ID, name, description, type, and repeat count (whether more than one instance of the option is allowed or required). (For details on the data types and repeat count values, see the "Option Definition Data Types and Repeat Counts" section on page 21-14.)

Figure 21-5 Add DHCP Option Definition Page (Local Advanced)



Note You cannot add an option definition for an option number or name that already exists. However, you can modify any option definition that appears as a hyperlink on the page.

Step 4 Click **Add Option Definition**. Then, on the List DHCP Option Definitions page, click **Modify Option Definition Set**.

Step 5 If you modify a standard definition in a set, a Revert icon ( ) appears next to it on the List DHCP Option Definitions page. Click this icon if you want to revert to the original definitions in that standard set.

Step 6 In the regional web UI, you can also pull replica definition sets and push definition sets to local clusters. (See the "Pulling Option Definition Sets from Replica Data" section on page 21-16 and the "Pushing Option Definition Sets to Local Clusters" section on page 21-16.)

CLI Commands

To view the entire list of standard DHCP option definitions, use **option-set dhcp-config** [**show**] or **option-set dhcp6-config** [**show**], or **option** {*id* | *name*} *option-set* **show** to view a specific definition. For example:

```
nrcmd> option-set dhcp-config
nrcmd> option subnet-mask dhcp-config show
```

To add a definition to a set, use **option** *id option-set* **create** *name type*. You cannot add a definition for an option ID (number) or name that already exists. For example, to add option number 222 with the name example-option in the dhcp-config option set, with a string type, use:

```
nrcmd> option 222 dhcp-config create example-option AT_STRING
```

To get a particular option attribute value, use **option** (*id* | *name*} *optionset* **get** *attribute*. To modify an option attribute, use **option** (*id* | *name*} *optionset* **set**. You can also unset an option attribute.

Creating Custom Option Definitions

You can create custom option definitions in the standard sets. Click the **dhcp-config** or **dhcp6-config** set on the List DHCP Option Definition Sets page (see Figure 21-4 on page 21-10). Then proceed with Step 3 in the "Using Standard Option Definition Sets" section on page 21-10.

Creating Vendor-Specific Option Definitions

You can send vendor-specific option data to DHCP clients that request them.

Note In prior Network Registrar releases, setting vendor-specific options was available in the CLI by using **vendor-option** *name* **create**, together with setting option data types by using **option-datatype** *name* **create** and **option-datatype** *name* **defineField**. There was only one vendor option code defined, for option 43, and the **vendor-option** command implicitly operated on this option. As of Network Registrar 6.2, there are several other option codes set aside for vendor-specific options, so that you must explicitly specify the option code number for which you are creating a vendor-specific option definition.

In Network Registrar, you can create vendor-specific option definitions in the web UI, or in the CLI by using **option** *id option-set-name* **create**. (For details on the option data types, see the "Option Definition Data Types and Repeat Counts" section on page 21-14.)

Vendor-specific options are sent in the following DHCP options:

- ***vendor-encapsulated-options* (43)**—Set this to a binary data type, then add the vendor-specific suboption definitions. (The data type of the parent option definition is a placeholder only. The suboption definitions define the valid option value formatting.)
- ***v-i-vendor-info* (125) or *vendor-options* (17) for DHCPv6**—Set this to a vendor-opts data type, then add the vendor-specific suboption definitions.

You can create vendor-specific option definitions for DHCPv4 options 43 and 125, and DHCPv6 option 17. You add the vendor-specific option definitions into a vendor option definition set that you create.

Caution Changing option definition properties, or deleting the option definition altogether, can have unexpected side effects on policies. If you delete a custom option definition, also check for the policies that include an option value. Changing an option definition changes the way that they are displayed, not what is stored, so that you do not need to modify the policy value unless you want the policy to return a differently formatted option value. Some option types are very similar, and changing between them can have side effects. For example, strings and DNS names are both entered as string values in the user interfaces, but the formatted option values are quite different.

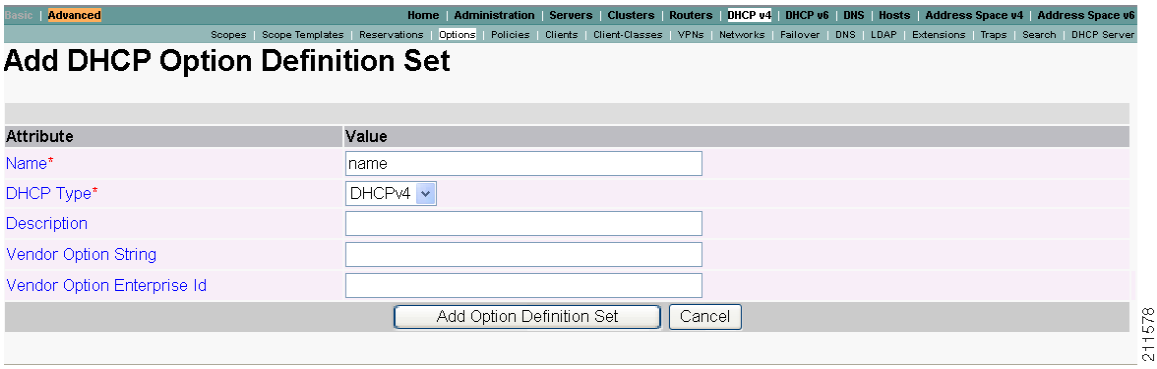
Note Network Registrar 7.0 preconfigures separate CableLabs (enterprise ID 4491) option definitions in the **dhcp-cablelabs-config** and **dhcp6-cablelabs-config** vendor-specific option definition sets.

Local Advanced and Regional Web UI

Step 1 Click **DHCP**, then **Options** to open the List DHCP Option Definition Sets page (see Figure 21-4 on page 21-10). View the existing DHCPv4 or DHCPv6 options.

Step 2 Click **Add Option Definition Set** to open the Add DHCP Option Definition Set page (see Figure 21-6).

Figure 21-6 Add DHCP Option Definition Set Page (Local Advanced)



Step 3 Enter a name for the option definition set, then choose DHCPv4 or DHCPv6 from the DHCP Type drop-down list.

If you are creating vendor-specific option definitions using:

- Option 43, enter a value in the Vendor Option String field.
- Option 125 for DHCPv4 or option 17 for DHCPv6, enter a valid Enterprise Option Enterprise ID value.

Step 4 Click **Add Option Definition Set**.

Step 5 Click the option definition set name.

Step 6 On the Edit DHCP Option Definition Set page, click **Add/Edit Option Definitions**. This opens the List DHCP Option Definitions page. Any existing option definitions will appear on this page (new or modified standard definitions are marked with an asterisk).

Step 7 Click **Add Option Definition**. This opens the Add DHCP Option Definition page (see Figure 21-5 on page 21-11).

Step 8 Enter the ID number of the option definition, along with its name and a description. The ID must be 43, 125, or 17 (for DHCPv6) for the client to recognize a vendor-specific option definition. The option name does not need to match the one specified in the RFC and can be of your own creation.

Step 9 Choose or enter the data type and repeat count (or enter an absolute repeat count in the next field). The data type must be:

- Binary (AT_BLOB) for option 43.
- Vendor-opts (AT_VENDOR_OPTS) for option 125 (for DHCPv4) and option 17 (for DHCPv6).

(For details on the data type and repeat count values, see the "Option Definition Data Types and Repeat Counts" section on page 21-14.)

Step 10 Click **Add Option Definition**. Then, on the List DHCP Option Definitions page, click **Modify Option Definition Set**.

CLI Commands

Use **option-set dhcp-custom list** (or **listnames**) to list just the vendor-specific option definitions. Use **option-set** *name* **create** *dhcp-type* to create a DHCP option set. The *dhcp-type* is either **8-bit** (for DHCPv4) or **16-bit** (for DHCPv6). For example, to create an option definition set with the name myoptionset for DHCPv4, with an option 43 vendor option string equal to ve-string, enter:

```
nrcmd> option-set myoptionset create 8-bit vendor-option-string=ve-string
nrcmd> save
```

Use **option** *optionset* **list** to show all the option definitions in a set. To show all the values for the option definitions in the set, use **option** *optionset* [**show**]. Use **option** *optionset* **get** to show individual properties of the option definition. You can also unset an option.

Use **option** *id option-set* **create** *name type* to create the definition. For example, to create option 43 for the v-s-optionset option set, naming it vendor-opt-43 with a binary data type, use:

```
nrcmd> option 43 v-s-optionset create vendor-opt-43
nrcmd> save
```

Option Definition Data Types and Repeat Counts

The data type values that you can use appear in Table 21-2 (for a full description of these data types, see the "Option Validation Types" section on page B-24).

Table 21-2 Option Definition Data Types

AT_INT8 unsigned 8-bit	AT_SHORT unsigned 16-bit	AT_INT unsigned 32-bit	AT_STRING string
AT_SINT8 signed 8-bit	AT_SSHORT signed 16-bit	AT_SINT signed 32-bit	AT_NSTRING string (no termination)
	AT_SHRTI unsigned 16-bit (Intel)	AT_INTI unsigned 32-bit (Intel)	AT_BLOB binary
	AT_SSHRTI signed 16-bit (Intel)	AT_SINTI signed 16-bit (Intel)	
AT_DNSNAME DNS name	AT_IPADDR IP address	AT_BOOL boolean	AT_DATE date
AT_RDNSNAME relative DNS name	AT_IP6ADDR IPv6 address	AT_MACADDR MAC address	AT_TIME unsigned time
AT_VENDOR-CLASS vendor-class	AT_VENDOR-OPTS vendor-opts	AT_TYPECNT counted-type	AT_STIME signed time
AT_VENDOR_NOLEN vendor-nolen			AT_ZEROSIZE zero size

You can view these types in the CLI by using **option listtypes**.

To set the repeat count, set the *repeat-count* attribute to one of the following, or enter an absolute number:

- **ZERO_OR_MORE—0+** in the web UI
- **ONE_OR_MORE—1+** in the web UI
- **EVEN_NUMBER—2n** in the web UI

In the CLI, for example, use:

```
nrcmd> option 200 ex-opt-def-set set repeat-count=ZERO_OR_MORE
nrcmd> save
```

Adding Suboption Definitions

You can set a suboption definition for the option definition by clicking **Add Suboption Definition** on the Edit DHCP Option Definition page. This opens the Add DHCP Option Definition page (see Figure 21-5 on page 21-11), where you can add the same values as for an option definition. The suboption definition you create is associated with its parent option (or parent suboption) definition. You can define up to six option and suboption levels.

Note You can add suboption definitions by using the web UI only. You currently cannot do so by using the CLI.

Suboption definition formats can be packed or type/length/value (TLV):

- **Packed**—A suboption with a zero ID value and an implicit data type. The option value is the only data in the packet. DHCPv6 options are virtually all defined with packed data. There are no markers for type or length and the layout of the data is inherent in the option definition. You cannot have further suboption definitions for packed suboptions.
- **TLV**—A suboption with a value of 1 through 255 (or 65535) that includes a type, length, and value. The data in the packet has the type and length preceding the value.

In most cases, you will not be mixing packed with TLV suboptions for the same option.

In addition to adding the AT_NOLEN datatype, you can enter PAD (0) and END (255) options anywhere in a list of suboptions for vendor option definitions (it is not necessary for the vendor option definition itself). For example:

```
(0 )(0 )(suboption-1 1 64)(255 )
```

To enter suboption values when editing policies, see the "Adding Complex Values for Suboptions" section on page 21-7.

Importing and Exporting Option Definition Sets

Importing and exporting option definition sets is a way to copy them between servers. In the CLI, you can import and export option sets by using **import option-set** *file* and **export option-set** *name file*. For example, to import an option set for Preboot Execution Environment (PXE) clients, modify and import a sample file located in the /examples/dhcp directory:

```
nrcmd> import option-set /examples/dhcp/OptionSetPXE.txt
```

Some of the guidelines for the file format include:

- The version string in the file must match the version for the import utility.
- The utility imports just the first option definition set found in the file.
- Delimit objects using curly brackets ({ }), attributes using parentheses (( )), and lists of objects in attributes using square brackets ([ ]). Delimit string value attributes using quotes (" ").

Using some care, you can also edit the text file to make minor modifications to an option definition set. Network Registrar provides two sample option definition set text files in the examples/dhcp directory, OptionSetJumpStart.txt and OptionSetPXE.txt:

- **OptionSetJumpStart.txt**—Edit the vendor-option-string to match the dhcp-class-identifier (option 60) that your JumpStart clients are sending.
- **OptionSetPXE.txt**—Edit the vendor-option-string to match the dhcp-class-identifier (option 60) that your Pre-boot Execution Environment (PXE) clients are sending.

Pushing Option Definition Sets to Local Clusters

You can push option definition sets you create from the regional cluster to any of the local clusters. If you want to push a specific option definition set to a cluster, click **Push Option Definition** sets on the List DHCP Option Definition Sets page, which opens the Push DHCP Option Definition Set to Local Clusters page.

This page identifies the data to push, how to synchronize it with the local cluster, and the cluster or clusters to which to push it. The data synchronization modes are:

- **Ensure** (preset value)—Ensures that the local cluster has new data without affecting any existing data.
- **Replace**—Replaces data without affecting other objects unique to the local cluster.
- **Exact**—Available for "push all" operations only. Use this with caution, because it overwrites the data and deletes any other objects unique to the local cluster.

Choose the destination cluster or clusters in the Available field and move it or them to the Selected field.

Tip The synchronization mode and cluster choice settings are persistent for the duration of the current login session, so that they are in effect each time you access this page, unless you change them.

After making these choices, click **Push Data to Clusters**. This opens the View Push DHCP Option Definition Set Data Report page.

Pulling Option Definition Sets from Replica Data

You may choose to pull option definition sets from the replica data of the local clusters instead of explicitly creating them. (You may first want to update the option definition set replica data by clicking the Replicate icon [] next to the cluster name.) To pull the option definition sets in the web UI, click **Pull Replica Option Definition Sets** to open the Select Replica DHCP Option Definition Set Data to Pull page.

This page shows a tree view of the regional server replica data for the local clusters' option definition sets. The tree has two levels, one for the local clusters and one for the scope templates in each cluster. You can pull individual option definition sets from the clusters, or you can pull all of their option

definition sets. To pull individual ones, expand the tree for the cluster, then click **Pull Option Definition Set** next to its name. To pull all the ones from a cluster, click **Pull All Option Definition Sets from Cluster**. To pull the option definition sets, you must also choose a synchronization mode:

- **Ensure**—Ensures that the regional cluster has new data without affecting any existing data.
- **Replace** (preset value)—Replaces data without affecting other objects unique to the regional cluster.
- **Exact**—Available for "pull all" operations only. Use this with caution, because it overwrites the data and deletes any other objects unique to the regional cluster.

Setting Option Values for Policies

You enter option values on a policy. The option definitions in your server configuration control the format and values that you enter.

Local Advanced and Regional Web UI

On the List DHCP Policies page, click a policy to edit it. (Note that you cannot set options for policies in Basic mode.) On the Edit DHCP Policy page:

- To enter a standard DHCPv4 or DHCPv6 option value for a policy, choose it from the DHCPv4 Options or DHCPv6 Options drop-down list, then set a value for the option. Click **Add Option**.
- To enter a vendor-specific DHCPv4 or DHCPv6 option value for a policy, choose an option definition set in the DHCPv4 Vendor Options or DHCPv6 Vendor Options drop-down list, then click **Select**. The page changes to show the drop-down list that includes the option; choose it, then click **Add Option**.

Note that you can also edit policy attributes on this page. Be sure to click **Modify Policy**.

To edit a configured policy option, click the name of the configured option on the Edit DHCP Policy page to open the Edit DHCP Policy Option page. Enter a new value, then click **Modify Option**.

CLI Commands

Use one of these commands:

```
nrcmd> policy name setOption {name | id} value
nrcmd> policy name setV6Option {name | id} value
nrcmd> policy name setVendorOption {name | id} option-set-name value
nrcmd> policy name setV6VendorOption {name | id} option-set-name value
```

To list the options in the policy, use one of these commands:

```
nrcmd> policy name listOptions
nrcmd> policy name listV6Options
nrcmd> policy name listVendorOptions
nrcmd> policy name lsitV6VendorOptions
```

To add suboption values, see the "Adding Complex Values for Suboptions" section on page 21-7.



CHAPTER 22

Managing Leases

Leases are at the center of the Dynamic Host Configuration Protocol (DHCP). They are the IP addresses allocated to individual clients for a certain time period. The DHCP server automatically allocates these leases with properly configured scopes that include valid IP address ranges. No two clients can have the same leased address. Reservations are leases that always get the same IP address.

This chapter describes how to manage leases and reservations in a network.

See Also

Configuring Leases in Scopes
Searching Server-Wide for Leases, page 22-9
Creating Lease Reservations, page 22-11
Setting Advanced Lease and Reservation Properties, page 22-14
Running Address and Lease Reports, page 22-21
Querying Leases, page 22-29

Configuring Leases in Scopes

After setting the IP address ranges for a scope, you can monitor and adjust the leases that result from DHCP assignments.

See Also

Viewing Leases, page 22-1
Lease States, page 22-4
Guidelines for Lease Times, page 22-4
Importing and Exporting Lease Data, page 22-5
Pinging Hosts Before Offering Addresses, page 22-7
Deactivating Leases, page 22-8
Excluding Leases from Ranges, page 22-8

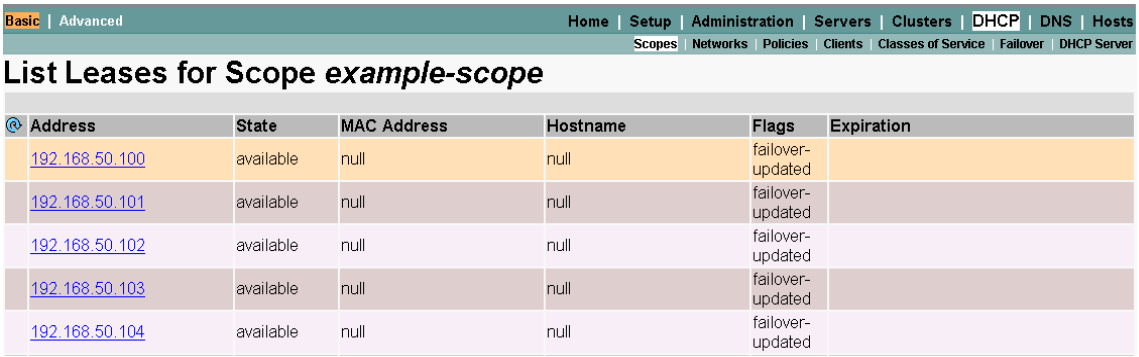
Viewing Leases

To view leases, you must first create a range of IP addresses for them in a scope, as described in the "Set Up DHCP" chapter of the *Quick Start Guide to Cisco Network Registrar* or the "Defining and Configuring Scopes" section on page 20-3, then wait for the DHCP server to generate leases based on these addresses.

Local Basic Web UI

To view leases, click **DHCP**, then **Scopes** to open the Manage Scopes page, then click the View icon (🔭) in the Leases column for the scope. This opens the List Leases for Scope page, where you can click each lease to manage it (see Figure 22-1).

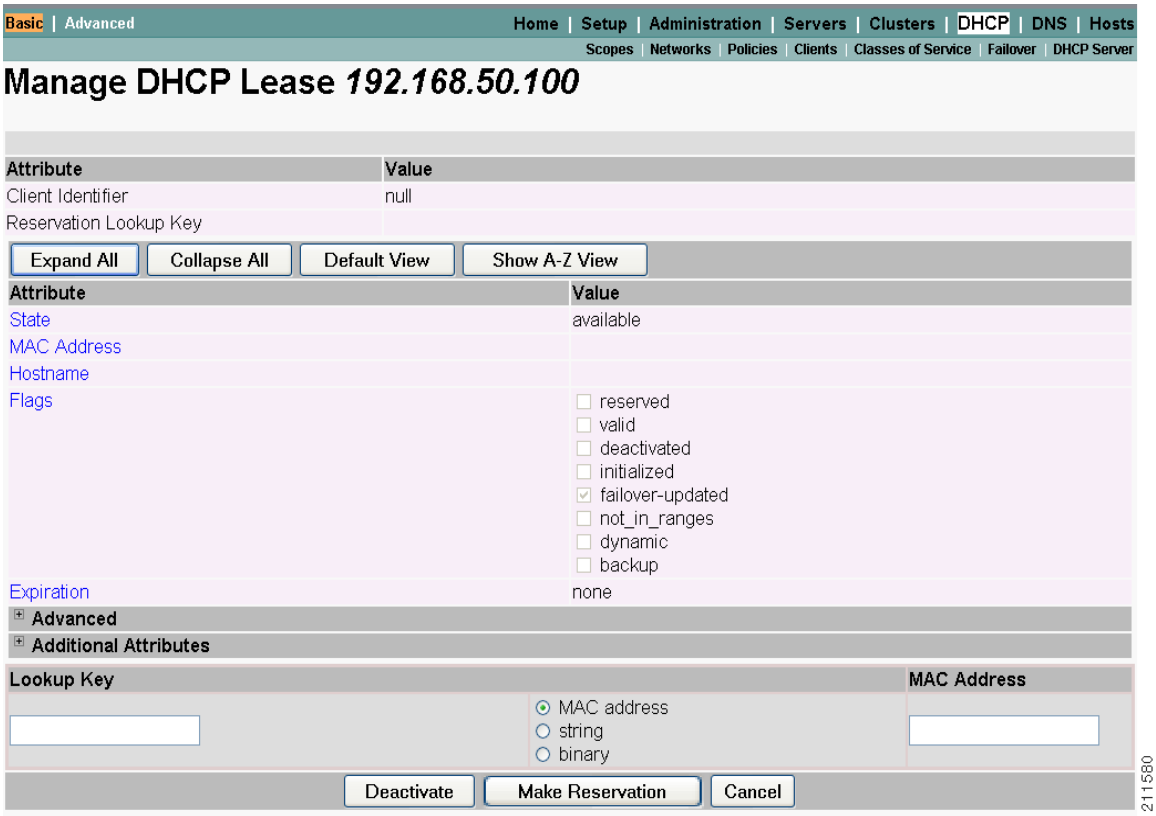
Figure 22-1 List Leases for Scope Page (Local Basic)



See the "Lease States" section on page 22-4 for a description of the values in the State column. For guidelines as to the lease expiration time, see the "Guidelines for Lease Times" section on page 22-4.

When you click the lease IP address, you open the Manage DHCP Lease page (see Figure 22-2 on page 22-3).

Figure 22-2 Manage DHCP Lease Page (Local Basic)



Local Advanced Web UI

Click **DHCP**, then **Scopes** to open the List/Add DHCP Scopes page. You can then click the View icon (🔭) in the Leases column for the scope; or you can click the name of the scope to open the Edit DHCP Scope page, then click **List Leases** in the Leases area of the page. Both actions open the List DHCP Leases for Scope page, where you can manage the leases as in Basic mode.

CLI Commands

Use **lease** *ipaddr* **show** to show the properties of a particular lease based on its IP address. Use **scope** *name* **listLeases** to show all the leases for a named scope. The output is nearly identical for both commands. Note that you cannot list leases in a particular virtual private network (VPN); all the leases in all the VPNs appear in the list.

You can show the most recent MAC address associated with a lease or what lease is associated with a MAC address. The **lease** *addr* **macaddr** command shows the MAC address of the lease, whether or not that lease is reserved or active. The **lease** *addr* **list –macaddr** command lists the lease data only if the IP address for that MAC address was actively leased (and not reserved). You can also list leases by LAN segment and subnet by using **lease** *addr* **list –subnet network** *netaddr netmask*.

Lease States

A lease can be in one of the states described in Table 22-1.

Table 22-1 Lease States

State	Description
Available	IP address available to be leased.
Unavailable	Not leasable. See the "Handling Leases Marked as Unavailable" section on page 22-19 for ways the DHCP server might set a lease to unavailable.
Leased	Held by a client.
Offered	Offered to the client.
Expired	Available when the lease grace period expires.
Deactivated	Not renewable or leasable after the lease expires. See the "Deactivating Leases" section on page 22-8.
Pending available	Failover-related. See Chapter 27, "Configuring DHCP Failover."

Note The DHCP server logs the lease state only if you enable the *ip-history-detail* attribute for the server.

When the state of an existing lease changes (such as when it becomes reserved or deactivated), the DHCP server does not report the change as an IP lease history change to the regional servers. The DHCP server reports a lease history change only when the lease expires or the DHCP server leases the IP address to another client. (See the "Running IP Lease Histories" section on page 22-21.)

Guidelines for Lease Times

To define appropriate values for lease times, consider these events on your network:

- Frequency of changes to DHCP options and default values.
- Number of available IP addresses compared to clients requesting them.
- Number of network interface failures.
- Frequency at which computers are added to and removed from the network.
- Frequency of subnet changes by users.

All these events can cause clients to release IP addresses or the leases to expire at the DHCP server. Consequently, the addresses may return to the free-address pool for reuse. If many changes occur on your network, Cisco recommends a lease time between one and three days for active networks, and between four and ten days for inactive networks. Assigning such a lease time reassigns IP addresses more quickly as clients leave the subnet.

Another important factor is the ratio of available addresses to connected computers. For example, the demand for reusing addresses is low in a class C network having 254 available addresses, of which only 40 are used. A long lease time, such as two months, might be appropriate in such a situation. The demand would be much higher if there were 240 to 260 clients trying to connect at one time. In this situation, you should try to configure more address space. Until you do, keep the DHCP lease time to under a hour.

Tip Short lease periods increase the demand that the DHCP server be continuously available, because clients will be renewing their leases more frequently. The DHCP failover functionality can help guarantee such levels of availability.

Be careful when creating policies that have permanent leases. A certain amount of turnover among clients occurs, even in a stable environment. Portable hosts might be added and removed, desktop hosts moved, and network adapter cards replaced. If you remove a client with a permanent lease, it requires manual intervention in the server configuration to reclaim the IP address. It would be better to create a long lease, such as six months, to ensure that addresses are ultimately recovered without administrator intervention.

Recommendations for lease durations include:

- Set cable modem lease times to seven days (604800 seconds). The leases should come from private address space, and the cable modems should seldom move around.
- Leases for customer premises equipment (CPE) or laptops should come from public address space and should match the habits of the user population, with as long a lease as possible to reduce load on the server.
- Shorter lease times require more DHCP request and response buffers. Set the request and response buffers for optimal throughput (see the "Setting DHCP Request and Response Packet Buffers" section on page 27-18).
- Allow the server to determine the lease period, by ensuring that the *allow-lease-time-override* policy attribute is disabled, which is its normal default. Even if enabled, clients can only request lease times that are shorter than you configure for the server. Some clients always request a fixed lease time (such as an hour) or the same one they had previously. These kinds of requests can cause problems in that the client never gets the full lease time, thereby generating more traffic for the server.
- Defer any lease extensions for clients trying to renew leases before the halfway mark in the lease. For details, see the "Deferring Lease Extensions" section on page 23-6.

Importing and Exporting Lease Data

You can use the CLI to import lease data to, and export from, text files.

Import Prerequisites

Before you can import leases, you must perform several configuration steps:

1. Configure a scope or scopes in the DHCP server for the leases that you plan to import.
2. If you want the hostnames for the leases dynamically entered into DNS as part of the import, configure zones in the DNS server to allow dynamic updates from the DHCP server.
3. Set the DHCP server to import mode so that it does not respond to other lease requests during the lease importing.
4. For all the time fields, use either the number of seconds since midnight GMT January 1, 1970, or a day, month, date, time, year format (Mon Apr 15 16:35:48 2002).
5. After you import the leases, take the DHCP server out of import mode so that it can respond to other lease requests.



Note Importing permanent leases will fail if you disable the permanent leases option. Enable this option using **policy** *name* **enable permanent-leases**, as necessary.

Import and Export Commands

The **import leases** and **export leases** commands use a special file format. Each record, or line, in the file represents one DHCP client:

field-1*|*field-2*|*field-3*|...|*field-13

Do not use spaces between the vertical line (|) delimiter and the field values. You must include at least the first four required fields. If you include more, you must delimit all the remaining null fields with the vertical line (|) so that there are 13 fields. The fields are, in order:

1. MAC address in *aa*:*bb*:*cc*:*dd*:*ee*:*ff* format (required)
2. MAC address type (required)
3. MAC address length (required)
4. IP address in dotted decimal format, *a.b.c.d* (required)
5. Start of lease time (Greenwich Mean Time, GMT) (optional)
6. Lease expiration time (GMT) (optional)
7. Allowable extension time (GMT) (optional)
8. Last transaction time (GMT) (optional)
9. IP address of the DHCP server (optional)
10. Hostname (without domain) (optional)
11. Domain name (optional)
12. Client ID (optional)
13. VPN name (optional; if omitted, the global VPN is used)

For all the time fields, use either the number of seconds since 1970, or the *day-month-date-time-year* format (such as Mon Apr 9 16:35:48 2007).

When importing leases, the DHCP server might not accept a lease, or a communication failure might drop the lease packet. In the latter case, the server retries the import several times, and after about a minute, reports a failure. If the import fails, check the DHCP server log file to find the lease that caused the error. Then go back to the import file, delete all lease entries up to and including the offending one, and repeat the lease import.

When you use **export leases**, you can choose between writing the state of all current and expired leases, or just the current leases, to the output file. Example 22-1 shows part of a lease data export from a Network Registrar DHCP server. The blank lines between records appear in the example for clarity; they are not in the actual output.

Example 22-1 Lease Data Export

```
00:60:97:40:c1:96|1|6|204.253.96.103|Wed Aug 30 08:36:57 2000|Fri Sep 01 13:34:05 2000|
Wed Aug 30 08:36:57 2000|Fri Sep 01 09:34:05 2000|204.253.96.57|nomad|cisco.com|
00:d0:ba:d3:bd:3b|blue-vpn

00:d0:ba:d3:bd:3b|1|6|204.253.96.77|Thu Aug 17 13:10:11 2000|Fri Sep 01 14:24:46 2000|
Thu Aug 17 13:10:11 2000|Fri Sep 01 10:09:46 2000|
204.253.96.57|NPI9F6AF8|cisco.com|blue-vpn
```

```
00:d0:ba:d3:bd:3b|1|6|204.253.96.78|Fri Jun 23 15:02:18 2000|Fri Sep 01 14:11:40 2000|
Fri Jun 23 15:02:18 2000|Fri Sep 01 09:56:40 2000|
204.253.96.57|JTB-LOCAL|cisco.com|blue-vpn
```

Lease Times in Import Files

During an import, the DHCP server gives the client the lesser of these lease times:

- The time specified in the import file.
- The time that the client receives if it were to acquire a lease using the existing configuration of the DHCP server.

For example, at 2:00 P.M. your scope is configured for a one-hour lease. According to the file that you import, the lease time does not expire until 5:00 P.M. After you import the file, the lease expires at 3:00 P.M. and not at 5:00 P.M.

Note If your import file specifies a DNS zone name, the server does not use the zone name when it updates DNS. If the file specifies a hostname, then the server uses the hostname when updating DNS, unless hostname specification in a client or client-class entry overrides the hostname.

The only way to indicate to the DHCP server that the client hostname should be in a zone other than the zone associated with the DNS update configuration object used for the DNS update is to specify that zone in a client or client-class entry.

Pinging Hosts Before Offering Addresses

You can have the DHCP server use the Internet Control Message Protocol (ICMP) echo message capability (also known as **ping**) to see if anyone responds to an IP address before assigning it. This test allows the DHCP server to check that an address is not in use before assigning it. Using **ping** can help prevent two clients from using the same address. If a client responds to the ping, the DHCP server marks that address as *unavailable* and offers a different address. This test works only for powered-up clients; it is possible for clients to have a lease and be powered down.

Tip The ping timeout period is important. Because pinging helps to ensure that no client is using a particular IP address, each ping must wait the entire timeout period. This ping timeout period comes before an offer, so the time specified has a considerable effect on server performance. If you set this time too long, it slows down the lease offering process. If you set this time too short, it reduces the effectiveness of the ping packet to detect another client using the IP address.

To implement pinging hosts before offering IP addresses, modify the scope by:

- Enabling the *ping-clients* attribute. It is disabled by default.
- Setting the *ping-timeout* attribute. It is 300 milliseconds by default.

The server makes unavailable any IP address for which it receives a successful ECHO reply. You can control this action by enabling the DHCP server attribute *ignore-icmp-errors* (the preset value). If disabled, the DHCP server also uses ICMP DEST_UNREACHABLE and TTL_EXPIRED error messages that it receives after sending ICMP ECHO requests as reasons for making an IP address unavailable.

Deactivating Leases

Deactivating a lease moves a client off of it. If the lease is available, deactivating it prevents the DHCP server from giving it to a client. If the lease is active (held by a client), deactivating it prevents the client from renewing it and the server from giving the lease to another client. You can deactivate a lease only if the server is running. The DHCP server deactivates the lease immediately.

Tip If you did not set synchronous scope edit mode, you must reload the DHCP server.

To force a Windows client to release its lease, run **ipconfig /release** on the client machine.

Local Basic Web UI

To deactivate a lease, click the address of the lease on the List Leases for Scope page (see the "Viewing Leases" section on page 22-1). On the Manage DHCP Lease page, click **Deactivate**. The lease now shows as deactivated. To reactivate the lease, click **Activate**.

Local Advanced Web UI

To deactivate a lease, the same operations exist as in Basic mode, except that you click the address of the lease on the List DHCP Leases for Scope page, which opens the Manage DHCP Lease page.

CLI Commands

To deactivate a lease, use **lease** *ipaddr* **deactivate**. To reactivate a lease, use **lease** *ipaddr* **activate**.

Excluding Leases from Ranges

IP address ranges, by definition, must be contiguous. To exclude a lease from an existing range, you must divide the range into two smaller ones. The new ranges consist of the addresses between the original starting and ending range addresses and the address that you want to exclude.

Caution If the excluded address currently has an active lease, you should first follow the steps in the "Deactivating Leases" section on page 22-8, otherwise you will get a warning message. Deleting an active lease can result in a duplicate IP address if the deleted address is subsequently reconfigured and then reassigned. Information about that lease will no longer exist after you reload the server.

Local Basic Web UI

To exclude a lease from a scope address range:

Step 1 Click **DHCP**, then **Scopes** to open the Manage Scopes (Address Pools) page.

Step 2 Click the name of the scope to open the Edit DHCP Scope (Address Pool) page.

Step 3 In the Ranges area, click the Delete icon ( ) next to the IP address range you want to remove.

Step 4 Add a range that ends just before the excluded IP address.

Step 5 Add another range that begins just after the excluded IP address.

Step 6 Modify the scope. If you set staged scope edit mode, you must also reload the DHCP server.

Local Advanced Web UI

To exclude a lease from a scope address range, the same operations exist as in Basic mode, except that you click the name of the scope on the List/Add DHCP Scopes page, which opens the Edit DHCP Scope page.

CLI Commands

To exclude a lease from a scope address range, discover the lease range (**scope** *name* **listRanges**), deactivate the lease (**lease** *ipaddr* **deactivate**), then remove the range of just that IP address (**scope** *name* **removeRange**). The resulting ranges are then split appropriately.

The following example removes the 192.168.1.55 address from the range. Note that if the lease is in a scope with a defined VPN, you must explicitly define that VPN for the session, or you can include the VPN prefix in the **lease** command:

```
nrcmd> session set current-vpn=red
nrcmd> scope examplescope1 listRanges
nrcmd> lease red/192.168.1.55 deactivate
nrcmd> scope examplescope1 removeRange 192.168.1.55 192.168.1.55
nrcmd> scope examplescope1 listRanges
```

If you set staged edit mode for the scope, reload the server.

Searching Server-Wide for Leases

Network Registrar 7.0 provides a new function with which you can search for leases server-wide. The search is a filter mechanism whereby you can specify a combination of lease attributes to target one or more leases configured for the network. The search function is available at the local cluster only and is provided separately for DHCPv4 and DHCPv6 leases.

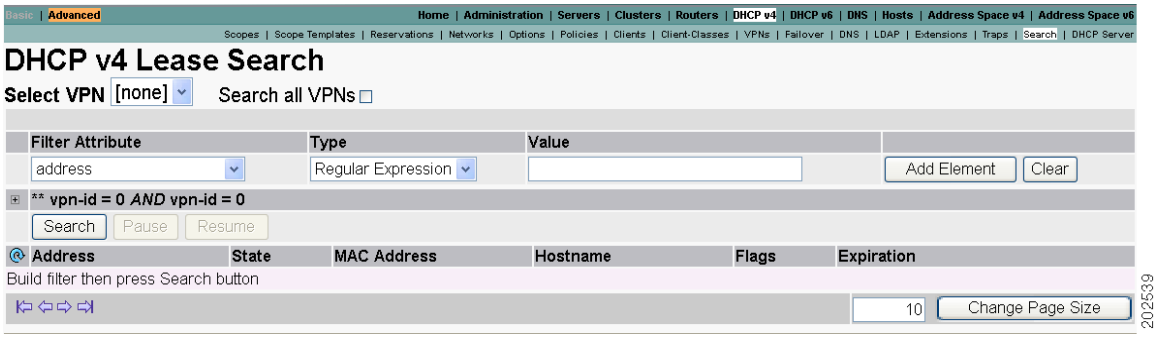
Local Advanced Web UI

Step 1 Click **DHCP**, then **Search** to open the DHCP Search page (see Figure 22-3 on page 22-10 for the DHCPv4 version of this page).

Step 2 If necessary, choose a specific VPN (the [none] VPN is the current one) or search in all VPNs.

Step 3 Choose a Filter Attribute from the drop-down list, such as client-id. DHCPv4 and DHCPv6 have separate lists of filter attributes. Attributes in the list are greyed out once they are selected as an element.

Figure 22-3 DHCPv4 Lease Search Page (Local Advanced)



Step 4 Choose a filter Type from the drop-down list. You can choose at least Binary or Regular Expression, but the list can contain one or more of the following, depending on the Filter Attribute selected:

- Binary—Value is in binary notation.
- Date Range—Range of date values, From a date and time To a date and time.
- Integer—Value is an integer.
- Integer Range—Integer From value to an integer To value.
- IP Address—Value is an IP address.
- IP Range—IP address From value to an IP address To value.
- IP Subnet—Value is an IP subnet.
- Regular Expression—Value is a Regular Expression in regex syntax. (For common regex usage, see Table 5-4 on page 5-22).

Step 5 Enter a Value, based on the Type selected. To clear the filter, click **Clear Filter**.

Step 6 Click **Add Element** to add the search element to the Filter Elements list. You can delete the element by expanding the filter display, then clicking the Delete icon ( ) next to the element.

Step 7 Once you assemble a list of elements, you can search on them, so that the elements are ANDed together for the result. Click **Search**.

Step 8 Check the table of resulting leases from the search, which shows for each an address, state, MAC address, hostname, flags, and expiration date. If necessary, change the page size to see more entries. The leases are ordered by IP address.

Tip The filter elements are ANDed together for the search. If you find that the search results do not yield what you expect, look at the Filter Elements list again and delete elements that can obstruct the results.

CLI Commands

Use **lease list –macaddr** *mac-addr* [**–vpn=***vpn-name*] to find leases in the DHCPv4 space. Specify the MAC address of the lease. If you omit the VPN designation, you base the search on the current VPN.

For leases in the DHCPv6 space, use the following **lease6 list** syntax:

```
nrcmd> lease6 list
    [-duid=client-id]
    [-lookup-key=key] [-blob | -string]]
    [-macaddr=mac-addr]
    [-cm-macadd=cm-mac-addr]
    [-vpn=vpn-name]
    [-count-only]
```

The **–macaddr** and **–cm-macddr** options are to search for leases identified by the CableLabs DOCSIS *vendor-opts* option (DHCPv6 option 17). For example, for these two commands:

```
nrcmd> lease6 -macaddr=01:02:03:04:05:06
nrcmd> lease6 -cm-macaddr=01:02:03:04:05:06
```

The **–macaddr** line lists leases where the option 17 *device-id* suboption (36) contains the requested MAC address. The **–cm-macddr** line lists leases where the option 17 *cm-mac-address* suboption (1026) matches the requested MAC address. (See Table C-4 on page C-7 for details on these suboptions.)

Creating Lease Reservations

To ensure that a client always gets the same lease, you can create a lease reservation. Managing lease reservations is available only to administrators having the dhcp-admin role at the local cluster, or the central-cfg-admin role with the dhcp-management subrole at the regional cluster.

Tip Try to limit the use of reservations. Because they are stored for scopes and not leases, using them adds to the administrative burden and complication in the case of failover. Limit reservations to less than one thousand per scope. To configure extended leases, it is often better to set long grace periods instead of reservations.

Caution If multiple DHCP servers distribute IP addresses on the same subnet, the client reservations must be identical. If not, a client for whom a lease reservation exists can receive offers of different IP addresses from different servers.

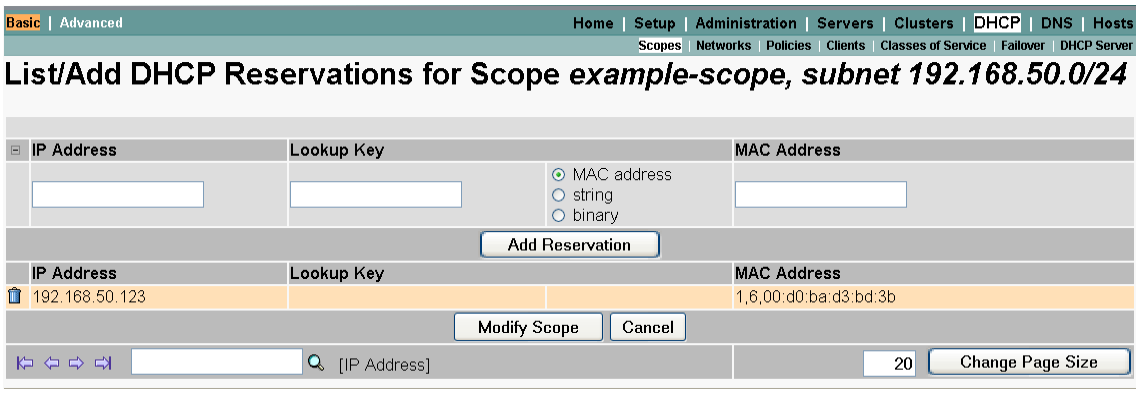
A lease reservation pairs an IP address with a MAC address or lookup key. You can choose any valid IP address on the network; it does not necessarily have to be in one of the scope ranges. You can use the IP addresses in the scope range for dynamic leases and the ones outside the range for reserved leases. However, even though a reserved IP address may not be in the scope range, the policy associated with the scope still applies to the address.

Local Basic Web UI

To view lease reservations, click **DHCP**, then **Scopes** to open the Manage Scopes (Address Pools) page, then click the View icon ( ) in the Reservations column to open the List/Add DHCP Reservations for Scope page (see Figure 22-4 on page 22-12).

To create a reservation on this page, enter the IP address of the lease you want to reserve, and either enter its MAC address in the MAC Address field or a lookup key in the Lookup Key field. If you enter a lookup key other than a MAC address in the Lookup Key field, also click the string or binary radio button, as appropriate for the entry. Click **Add Reservation**, then **Modify Scope**.

Figure 22-4 List/Add DHCP Reservations for Scope Page (Local Basic)



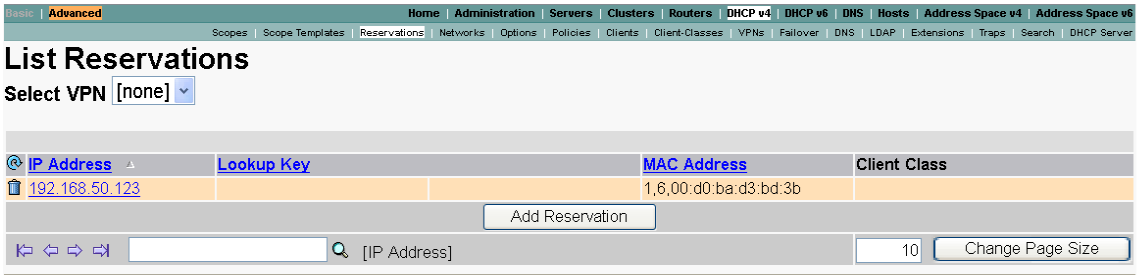
Viewing Reservations for Scopes in Local Advanced Web UI

To view lease reservations, click **DHCP**, then **Scopes** to open the List/Add DHCP Scopes page, then click the View icon () in the Reservations column to open the List/Add DHCP Reservations for Scope page. Proceed as for the Basic web UI.

Configuring Reservations Directly in Local Advanced Web UI

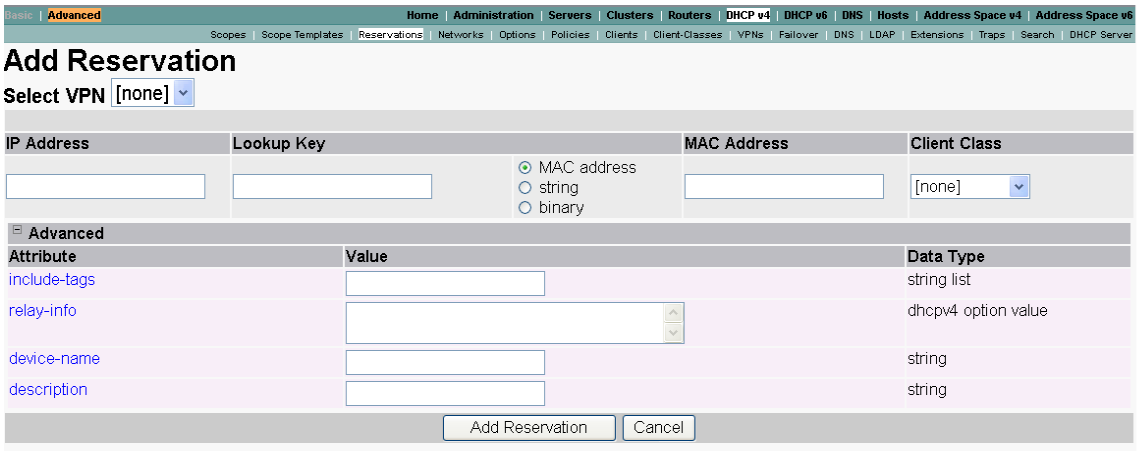
Advanced mode also provides a mechanism to create reservations independent of scopes. Click **DHCP**, then **Reservations** to open the List Reservations page (see Figure 22-5).

Figure 22-5 List Reservations Page (Local Advanced)



For reservations that could be in one of several scopes, this method requires that you enter client-class selection and other advanced attributes so that you can associate the reservation with the correct scope. On this page, click **Add Reservation** to open the Add Reservation page (see Figure 22-6 on page 22-13 that shows the Advanced attributes expanded).

Figure 22-6 Add Reservation Page (Local Advanced)



Enter the IP address of the reservation in the IP Address field, then enter either its MAC address or a string or binary lookup key (do not enter both a MAC address and lookup key).

Further down the page, you can set advanced attributes such as *include-tags*, *relay-info*, *device-name*, or a *description* (see the "Setting Reservation Attributes" section on page 22-13). (Choose either a client-class or enter an *include-tags* value, but not both.) Click **Add Reservation** when you are done.

Note In Expert mode, you can synchronize reservations on the CCM server from the DHCP server, and synchronize reservations on the DHCP server from the CCM server, on the List Reservations page. Click **Sync CCM Server from DHCP Server** to open the Sync Reservations on CCM Server from DHCP Server page. Click **Sync DHCP Server from CCM Server** to open the Sync Reservations on DHCP Server from CCM Server page. Click **Return** on both pages.

CLI Commands

To list the reservations for a scope, use **scope** *name* **listReservations**. To list all the available reservations, use **reservation list** [[*vpn*/]*ipaddr* | **–mac** | **–key**]. Using the **–mac** or **–key** option shows a value for the MAC address or lookup key for each reservation.

To add a reservation to a particular scope, use **scope** *name* **addReservation** *ipaddr* {*macaddr* | *lookupkey*}. To add a reservation just for the IP address, use **reservation** [*vpn*/]*ipaddr* **create** {*macaddr* | *lookupkey*}. If you use a *lookupkey* other than the MAC address, you must add the **–blob** or **–string** option to the command. The DHCP server assigns the reservation to the scope that encompasses its IP address. You can set attribute values for the reservation such as the *client-class*, *include-tags*, *relay-info*, *device-name*, or a *description*. (See the "Setting Reservation Attributes" section on page 22-13.) If you set the scope edit mode to staged, you must save (**save**), then send the reservation (**lease** [*vpn*/]*ipaddr* **send-reservation**).

Setting Reservation Attributes

You can set certain attributes for lease reservations other than the VPN, IPv4 or IPv6 address, MAC address, or lookup key (see Table 22-2 on page 22-14). In the CLI, you use **reservation** [*vpn*/]*ipaddr* **set** *attribute*=*value*.

Table 22-2 Reservation Attributes

Attribute	Description
description	Optional description of the device that this reservation represents.
device-name	Optional name of the device that this reservation represents.
include-tags	Scope selection criteria for this reservation. Do not set this value if you are setting a *client-class*.
relay-info	Relay agent information option (82) value required for Leasequery purposes (see the "Querying Leases" section on page 22-29).

Setting Advanced Lease and Reservation Properties

Setting advanced lease and reservation properties can include:

- **Reserving currently leased IP addresses**—See the "Reserving Currently Leased Addresses" section on page 22-14.
- **Unreserving leases**—See the "Unreserving Leases" section on page 22-16.
- **Extending leases to non-MAC addresses**—See the "Extending Reservations to Non-MAC Addresses" section on page 22-16.
- **Forcing lease availability**—See the "Forcing Lease Availability" section on page 22-18.
- **Inhibiting lease renewals**—See the "Inhibiting Lease Renewals" section on page 22-18.
- **Handling leases marked as unavailable**—See the "Handling Leases Marked as Unavailable" section on page 22-19.
- **Setting timeouts for unavailable leases**—See the "Setting Timeouts for Unavailable Leases" section on page 22-20.

Reserving Currently Leased Addresses

You can delete a reservation for one client while reusing it for another one, even though the first client still has the lease.

Local Advanced Web UI

To reserve an existing lease:

Step 1 Click **DHCP**, then **Scopes**, then the name of the scope to open the Edit DHCP Scope page.

Step 2 Click the View icon () under the Leases column.

Step 3 Click the IP address of the lease on the List DHCP Leases for Scope page.

Step 4 On the Manage DHCP Lease page, if the IP address is not leased (in available state), enter the lookup key or MAC address for the reservation.

Step 5 Click **Make Reservation**. On the List DHCP Leases for Scope page, the lease will appear as reserved.

Step 6 Modify the scope.

Step 7 To remove the reservation, click **Remove Reservation** on the Manage DHCP Lease page, then modify the scope. The lease no longer appears as reserved.

Example of Reserving an Existing Lease

This CLI command example creates a reservation from an existing lease:

```
nrcmd> reservation 192.168.1.110 create 1,6,00:d0:ba:d3:bd:3b
nrcmd> save
nrcmd> lease 192.168.1.110 send-reservation
nrcmd> lease 192.168.1.110 activate
nrcmd> save
```

Client 1,6,00:d0:ba:d3:bd:3b does a DHCPDISCOVER and gets an offer for 192.168.96.110. The client then does a DHCPREQUEST and gets an ACK message for the same IP address.

As time passes, client 1,6,00:d0:ba:d3:bd:3b does several DHCPREQUESTs that are renewals, which the server acknowledges. Then, at some time before the client lease expiration time, you terminate the reservation:

```
nrcmd> lease 192.168.1.110 deactivate
nrcmd> reservation 192.168.1.110 delete
nrcmd> save
nrcmd> lease 192.168.1.110 delete-reservation
```

You then add a reservation for a different client for that IP address, even though the address is still leased to the first client:

```
nrcmd> reservation 192.168.1.110 create 1,6,02:01:02:01:02:01
nrcmd> save
nrcmd> lease 192.168.1.110 send-reservation
nrcmd> lease 192.168.1.110 activate
nrcmd> save
```

This action results in an IP address that is leased to one client, but reserved for another. If the new client (1,6,02:01:02:01:02:01) does a DHCPDISCOVER before the original client (1,6,00:d0:ba:d3:bd:3b) does, the new client does not get 192.168.96.110, but gets a random IP address from the dynamic pool.

When the original client (1,6,00:d0:ba:d3:bd:3b) sends its next DHCPREQUEST/RENEW for the lease on 192.168.96.110, it gets a NAK message. Generally, upon receipt of the not-acknowledged message, the client immediately sends a DHCPDISCOVER. On receiving that DHCPDISCOVER, the server cancels the remaining lease time for 192.168.96.110.

The server then gives client 1,6,00:d0:ba:d3:bd:3b whatever lease is appropriate for it—some reservation other than 192.168.96.110, some dynamic lease (if one is available), or nothing (if no dynamic leases are available). When the new client (1,6,02:01:02:01:02:01) tries to renew the random IP address it received, the server sends it a NAK, because it wants to give it the reserved address. When the new client then does a DHCPDISCOVER, it gets the 192.168.96.110 reserved address.

You could also force availability of a lease (see "Forcing Lease Availability" section on page 22-18). However, doing so does not stop the original client (1,6,00:d0:ba:d3:bd:3b) from using 192.168.96.110. Also, it does not prevent the new client (1,6,02:01:02:01:02:01) from getting 192.168.96.110. In other words, this means that making a reservation for a client is independent of the lease state (and actual lease client) of the IP address for which the reservation is made.

Thus, making a reservation for one client does not cause another client to lose that lease right away, although that client receives a NAK response the next time it contacts the DHCP server (which could be seconds or days). Additionally, the client that reserved the IP address does not get that address if some other client already has it. Instead, it gets another IP address until the:

- IP address it is supposed to receive is free.
- Client sends a DHCPREQUEST as a renewal and receives a NAK response.
- Client sends a DHCPDISCOVER.

Unreserving Leases

You can remove lease reservations at any time. However, if the lease is still active, the client continues to use the lease until it expires. If you try to reserve the lease for a different client, you will get a warning.

Local Advanced Web UI

To unreserve a lease, click **DHCP**, then **Reservations** to open the List Reservations page, then click the Delete icon () next to the reservation you want to remove. This removes the reservation immediately, with no confirmation.

CLI Commands

To unreserve a lease, use **reservation** [*vpn/*]*ipaddr* **delete** or **scope** *name* **removeReservation** {*ipaddr* | *macaddr* | *lookupkey*} [**–mac** | **–blob** | **–string**].

You can also completely delete a reservation by using **lease** *ipaddr* **delete-reservation**. However:

- Ensure that the reservation is gone from the **nrcmd** internal database.
- If you use failover on the scope containing the reservation:
  1. Use **reservation** [*vpn/*]*ipaddr* **delete**, or **scope** *name* **removeReservation**, on both servers.
  2. On the backup server, if you are in staged scope edit mode, use **lease** [*vpn/*]*ipaddr* **delete-reservation**.
  3. Use the same command on the main server.

Save the result of this operation to preserve it across server reloads, because issuing **lease** *ipaddr* **delete-reservation** alone affects only the server internal memory.

Extending Reservations to Non-MAC Addresses

You might need to create lease reservations based on something other than the MAC address from the incoming client packet. Often, DHCP client devices attached to a switch port need to get the same IP address, regardless of the MAC address. This approach helps when you replace factory floor devices with identical devices (with different MAC addresses), but want to maintain the same IP address.

Overriding Client IDs

You can set an expression in a client-class *override-client-id* attribute that extracts the MAC address and port of a switch from the relay-agent-info option (82) and creates a client identity from it. Regardless of the client-id in the incoming packet, the identity that allocates an IP address is the same for any device

coming in through the same switch port. The expression you use for the attribute depends on the option 82 format. The DHCP server calculates the expression when it assigns the packet to the client-class. The *override-client-id* value becomes the identity of the client from that point onward.

However, when you enable the *use-client-id-for-reservations* attribute in a policy, the server turns the client-id of that request into a string of the form *nn*:*nn*:*nn* ... *nn*:*nn*, and uses that string to look up the reservation.

The *add-to-environment-dictionary* attribute for a client or client-class also serves to send attribute values to the DHCP extension environment dictionary (see Chapter 29, "Using Extension Points"), specified as name-value pairs. You can configure an *add-to-environment-dictionary* attribute on either a client or a client-class. If you choose to configure this attribute on both a client and client-class, you should ensure that the name-value pairs that you configure on the client have different names than the name-value pairs that you configure on the client-class, because they are all going to be put into the same environment dictionary (which can have only one value for a particular name). Generally, it is best to configure this attribute on a client or a client-class only, but not on both.

Local Advanced Web UI

You can find the *override-client-id* attribute on the Add DHCP Client-Class page (click **DHCP**, then **Client-Class**, then **Add Client-Class**) or Edit DHCP Client-Class page (click **DHCP**, then **Client-Class**, then the name of the client-class).

You also need to configure a client-class lookup ID for the DHCP server, to put every packet into a particular client-class where you configure the *override-client-id* expression. Click **DHCP**, then **DHCP Server**, then the Local DHCP Server link to open the Edit DHCP Server page. In the Client Class attributes, enter a *client-class-lookup-id* expression.

To use the client ID for the reservation, configure the policy to enable the *use-client-id-for-reservations* attribute on the Add DHCP Policy page (click **DHCP**, then **Policy**, then **Add Policy**) or Edit DHCP Policy page (click **DHCP**, then **Policies**, then the name of the policy).

CLI Commands

The syntax for setting the *override-client-id* attribute is **client-class** *name* **set override-client-id="***expression***"**. The syntax for setting the *client-class-lookup-id* attribute is **dhcp set client-class-lookup-id="***expression***"**. The syntax for setting the *use-client-id-for-reservations* attribute is **policy name enable use-client-id-for-reservations**.

Reservation Override Example

The following example shows how to override a client ID for a reservation:

Step 1 Create a scope for the reservation:

- **a.** Enter a subnet address.
- **b.** If you want dynamic reservations, add an IP address range.

Step 2 Add the reservation for the scope:

- **a.** Include a value for the lookup key.
- **b.** Specify the lookup key type as binary.

Step 3 Create a policy for the purpose, enabling the *use-client-id-reservations* attribute.

Step 4 Create a client-class for the purpose:

a. Specify the policy created in the previous step.

b. Include an expression for the *override-client-id* attribute that returns a blob value with the client ID you want, based on the contents of the packet.

Step 5 Get a lease for a client with the MAC address. This client will then get the override ID.

Forcing Lease Availability

You can force a current lease to become available. You should request that the user release the lease, or do so yourself, before forcing its availability. Forcing lease availability does not require a server reload.

Note After a lease is forced to be available, the client continues to use it until the client contacts the DHCP server.

Local Advanced Web UI

To force lease availability:

Step 1 Click **DHCP**, then **Scopes** to open the List/Add DHCP Scopes page.

Step 2 Click the View icon ( ) under the Lease column for the scope that has leases.

Step 3 Click the IP address of the lease on the List DHCP Leases for Scope page.

Step 4 On the Manage DHCP Lease page, click **Force Available**. On the List DHCP Leases for Scope page, the lease will now show an empty value in the Flags column.

CLI Commands

To force lease availability, use **lease** [*vpn*/]*ipaddr* **force-available**. Use **scope** *name* **clearUnavailable** to force all leases in the scope to become available.

Inhibiting Lease Renewals

Normally, the Network Registrar DHCP server retains the association between a client and its leased IP address. The DHCP protocol explicitly recommends this association and it is a usually desirable feature. However, for some customers, such as ISPs, clients with long-lived lease associations may be undesirable, because these clients should change their IP addresses periodically. Network Registrar includes a feature that allows customers to force lease associations to change when DHCP clients attempt to renew their leases or reboot.

A server can never force a client to change its lease, but can compel the client to do so based on a DHCPRENEW or DHCPDISCOVER request. Network Registrar offers configuration options to allow customers to choose which interactions to use to force a client to change its IP address:

- **Inhibiting all lease renewals**—While a client is using a leased address, it periodically tries to extend its lease. At each renewal attempt, the server can reject the lease, forcing the client to stop using the IP address. The client might have active connections that are terminated when the lease terminates, so that renewal inhibition at this point in the DHCP interaction is likely to be user-visible.
- **Inhibiting renewals at reboot**—When a DHCP client reboots, it might have recorded a valid lease binding that did not expire, or it might not have a valid lease. If it does not have a lease, you can prevent the server from granting the last held lease. If the client has a valid lease, the server rejects it, forcing the client to obtain a new one. In either case, no active connections can use the leased address, so that the inhibition does not have a visible impact.
- **Effect on reservations**—Reservations take precedence over renewal inhibition. If a client has a reservation, it can continue to use the reserved IP address, whether or not renewal inhibition is configured.
- **Effect on client-classes**—Client-class testing takes place after renewal inhibition testing. If a client is forced to change IP addresses by renewal inhibition, then client-class processing might influence which address the server offers to the client.

You can enable or disable lease renewal inhibition for a policy, which you can set system wide, for a scope or on a client-by-client basis. The *inhibit-all-renews* attribute causes the server to reject all renewal requests, forcing the client to obtain a new IP address any time it contacts the DHCP server. The *inhibit-renews-at-reboot* attribute permits clients to renew their leases, but the server forces them to obtain new addresses each time they reboot.

The DHCP server needs to distinguish between a client message that it should reject (such as a renewal request) and one that represents a retransmission. When the server processes a message, it records the time the packet arrived. It also records the time at which it made a lease binding to a client, and the last time it processed a message from the client about that binding. It then compares the packet arrival time with the lease binding time (the start-time-of-state) and processes packets from the client within a certain time interval from the start time of the binding. By default, this time interval is one minute.

Local Advanced Web UI

To inhibit lease renewals, create a policy on the Edit DHCP Policy page (click **DHCP**, then **Policies**, then the name of the policy), then enable the *inhibit-all-renews* or *inhibit-renews-at-reboot* attribute. (Both attributes are preset to disabled). Then, modify the policy.

Handling Leases Marked as Unavailable

One of the aspects of effective lease maintenance is determining the number of unavailable leases in a scope. This number is sometimes higher than expected. Each unavailable lease is probably an indication of a serious problem. Possible causes for an unavailable lease are:

- **The DHCP server is configured for a ping before an offer, and the ICMP echo message is returned successfully**—A currently active client is using that IP address, causing the DHCP server to mark it as *unavailable*. To prevent the server from doing so, disable pinging an address before offering it to a client. See the "Pinging Hosts Before Offering Addresses" section on page 22-7.

- **The server receives a DHCPDECLINE message from a client to which it leased what it considered to be a good IP address**—The client does an address resolution (ARP) request for the IP address on its local LAN segment, and another client responds to it. The client then returns the address to the server with a DHCPDECLINE packet and sends another DHCPDISCOVER packet to get a new address. The server marks as *unavailable* the address that the client returns. To prevent the server from reacting to DHCPDECLINE messages, you can set a scope attribute, *ignore-declines*.
- **The server receives "other server" requests from the client**—Because all DHCPREQUEST messages that follow DHCPOFFER messages are broadcast, the server can see messages directed to other DHCP servers. A server knows that a message is directed to it by the value of the *server-id* option in the packet. If the Network Registrar server recognizes a message directed at another server, in that its own IP address does not appear in the *server-id* option, but the address leased in the message is one that the server controls, it believes that two servers must be trying to manage the address simultaneously. It then marks the local address as *unavailable*. This behavior does not apply in a DHCP failover configuration. Either the two servers are configured with some or all of the same IP addresses, or (in rare cases) the DHCP client placed a wrong *server-id* option value in the packet.

  If you have reason to believe that the client is sending bad *server-id* options (rather than packets actually directed to other servers), Network Registrar has a server attribute you can enable that turns this behavior off, the *ignore-requests-for-other-servers* attribute.
- **Inconsistent lease data**—Extremely rare and occurring only during server startup when, while configuring a lease, the server reads the lease data from disk during a refresh of the internal cache. The lease state appears as *leased*, but there is incomplete data to construct a client for that lease, such as that the lease might not yet have a *client-id* option value. The server considers the data to be inconsistent and marks the IP address as *unavailable*. Forcing the lease to be available (such as by using the **lease** *ipaddr* **force-available** command in the CLI) should clear up this problem.

Setting Timeouts for Unavailable Leases

During the times when leases become unavailable, as described in the "Handling Leases Marked as Unavailable" section on page 22-19, all unavailable leases remain in that state for a configured time only, after which time they again become available. A policy attribute, *unavailable-timeout*, controls this time. The *system_default_policy* policy sets this value to one day by default.

To handle upgrades from previous releases of Network Registrar that do not have this timeout feature, a special upgrade timeout attribute, *upgrade-unavailable-timeout* (which is preset to one day) is included at the server level. The *upgrade-unavailable-timeout* value is the timeout given to leases set to unavailable before the Network Registrar upgrade. This setting affects the running server only and does not rewrite the database. If the server stays up for one day without reloading, all the unavailable leases that were present at the last reload will time out. If the server reloads in less than a day, the entire process restarts with the next reload. Note that this process occurs only for leases that were set unavailable before the upgrade. Leases that become unavailable after the upgrade receive the *unavailable-timeout* value from the policy, as previously described.

If a Network Registrar failover server receives an update from a Network Registrar DHCP server running prior to Network Registrar 6.0, the unavailable leases do not have a timeout value. In this case, the upgraded Network Registrar server uses the *unavailable-timeout* value configured in the scope policy or *system_default_policy* policy as the timeout for the unavailable lease. When the lease times out, the policy causes the lease to transition to available in both failover partners.

Running Address and Lease Reports

You can run these reports on IP addresses and leases:

- **Address Usage**—See the "Running Address Usage Reports" section on page 22-21.
- **Lease History**—See the "Running IP Lease Histories" section on page 22-21.
- **Current Utilization**—See the "Running Lease Utilization Reports" section on page 22-27.
- **Lease Notification**—See the "Receiving Lease Notification" section on page 22-27.

Running Address Usage Reports

The address usage reports show the IP addresses that are assigned leases.

Local Advanced Web UI

To view the leases for IP addresses, on the Edit DHCP Scope page, in the Leases area, click **List Leases** to open the List DHCP Leases for Scope page. To manage a specific lease, click its IP address on the page. This opens the Manage DHCP Lease page.

CLI Commands

To view the IP address usage for specified servers, use **report**.

Tip If you are not already using **lease-notification** in an automated way, try **lease-notification available=100%** for a concise scope-by-scope summary of the state of the servers.

Running IP Lease Histories

You can extract IP lease history data from a special database so that you can determine past allocation information for a given IP address. You can get a historical view of when a client was issued a lease, for how long, when the client or server released the lease before it expired, and if and when the server renewed the lease and for how long.

Network Registrar provides a client to control querying IP history data. Through this client, you can:

- Get the MAC addresses associated with a given IP address over a given time.
- See the entire IP history database as a comma-separated file.
- View the attributes of the lease history (the lease history detail report)—See the "Querying IP Lease History" section on page 22-22.

You must use additional administrative functions to trim the IP history database of records, to keep the size of the database from growing without bounds.

Note When the state of an existing lease changes (for example, when it is configured as a reserved IP address or it is deactivated), the change does not appear as a lease history change at the regional cluster, unless you enable the *ip-history-detail* attribute. With detail collection disabled, a lease history change appears only when the lease transitions from leased to not leased or is assigned to another client.

See Also

Enabling Lease History Recording at the Local Cluster
Querying IP Lease History
Trimming Lease History Data, page 22-26

Enabling Lease History Recording at the Local Cluster

You must explicitly enable lease history recording for the local cluster DHCP server. The DHCP server logs IP history recording errors in the usual DHCP log files.

Local Advanced Web UI

To enable lease history recording:

Step 1 Click **DHCP**, then **DHCP Server** to open the Manage DHCP Server page.

Step 2 Click the **Local DHCP** Server link.

Step 3 On the Edit DHCP Server page, look for the Lease History attributes:

- ***Lease History*** **(*ip-history*)**—Be sure this is set to enabled.
- ***ip-history-detail***—Enable this to get detailed lease history data.
- ***ip-history-max-age***—Maximum age of the lease history to collect. With lease history enabled, the DHCP server periodically examines the lease history records and deletes any records with lease history bindings older than this age threshold.

Step 4 Click **Modify Server** at the bottom of the page.

Step 5 If you set staged edit mode for the scope, reload the server.

CLI Commands

To enable lease history recording, you must explicitly enable recording IP (lease) history for IP addresses by using **dhcp enable ip-history**.

Querying IP Lease History

Once you have leases, you can query for their history. You set up the local cluster containing the DHCP server as part of the regional cluster, and enable polling for the lease history data from the regional cluster (see the "Enabling Lease History Collection" section on page 6-12).

You can adjust the polling criteria for the cluster in the regional cluster web UI by using the attributes described in the "Polling Subnet Utilization and Lease History Data" section on page 6-9.

You must also set the selection criteria for querying the lease history data, as described in the following sections.

Regional Web UI

To query IP lease history:

Step 1 Click **Address Space**, then **Lease History** to open the Query Lease History page. Using this functionality requires DHCP rights locally and address space rights at the regional cluster.

Step 2 You can query lease history based on the following criteria:

- **Virtual private network (VPN) for the addresses to be polled for lease data**—A VPN choice is available only if at least one was defined or pulled to the regional cluster (see the "Managing Virtual Private Networks" section on page 6-16). By default, the query is based on no specific VPN unless you choose it from the VPN drop-down list on the page. You can also query based on all VPNs.
- **Time range for the query**—Choose from one of the following time ranges for the lease history data:
  - last 10 days
  - last 30 days
  - last 60 days
  - last 90 days
  - from/to (limited to 90 days)

    If you choose the time range value, also choose the Start Date and End Date month, day, and year from the drop-down lists. The result depends on the value of the *poll-lease-hist-interval* attribute. Note that if the time range is set back one month, but you set the polling interval to greater than a month, no data will appear.
- **Criteria**—Choose the criteria on which you want to base the query (per VPN and time range):
  - By IP Address—Enter the IP address in the adjacent field.
  - By MAC Address—Enter the MAC address in the adjacent field.
  - By IP Range—Enter the IP address range in the adjacent field, the start of range in the left field and end of range in the right field.
  - All—Choose all the leases by no particular criteria.
  - Current Lease by IP—Show the current lease for an IP address (entered in the adjacent field).

Note The regional CCM server references the DHCP server to obtain the most recent lease data for the IP address. Therefore, the regional address space must include the matching subnet from the local cluster, and the particular DHCP server must be running.

Step 3 Click **Query Lease History**.

Tip At the upper-left corner of the List Lease History Records page is either the Log icon () for the Netscape browser that you can click to view a text version of the report, or the Save icon () for the Internet Explorer browser so that you can save the report to a file (.txt by default). The List Lease History page also has a View Detail column with the View icon () if you set the CCM server with the *ip-history-detail* attribute enabled. Click this icon to open the View Lease History Detail page. This page shows the changeset for each history record.

iphist Utility

You can query the IP history database at the local cluster and direct the results to standard output or a file by using the **iphist** utility. You must run this utility on the same machine as the DHCP server, and you must have superuser/root privileges to read and modify the database file. The default location is:

- **Windows**—\Program Files\Network Registrar\bin
- **Solaris and Linux**—/opt/nwreg2/usrbin

From the command prompt, change to the location and run the utility using the syntax:

iphist [*options*] {*ipaddr* | **all**} [*start-date* | **start** [*end-date* | **end**]]

The IP address is a single address or the keyword **all**, the start date is in local time or the keyword **start** for the earliest date in the database, and the end date is in local time or the keyword **end** for the last date in the database. However, the output is in Greenwich Mean Time (GMT) by default, unless you use the **–l** option to specify local time.

The full list of command options appears in Table 22-3.

Table 22-3 iphist Command Options

Option	Description
–N *username*	Administrator username. If omitted, you are prompted for the username.
–P *password*	Administrator password. If omitted, you are prompted for the password.
–C *cluster*[**:***port*]	Destination server and optional SCP port.
–a	Shows the lease attributes, visibility 5 and 3.
–b	Displays the local and backup server failover leases.
–f "*format***"**	Format of the output lines. The default format is: **"address,client-mac-addr,binding-start-time,binding-end-time"**
–l	Displays output in local time rather than the default GMT.
–m	Displays the local and main server failover leases.
–n *vpn*	Name or ID of an associated VPN, or the word **all** (for all VPNs) or **global** (for IP addresses without a VPN). If omitted, the query is based on the global VPN, or the current one set by the **session set current-vpn** command, unless you use the **all** value with the option.
–o *file*	Sends output to a file.
–v	Displays the database version.
–V *visibility*	Sets the visibility level of the output attributes. The visibility is 3 by default.
–z *debug-args*	Sets the debug output levels.

Dates can use this syntax (quotation marks are required if space characters are included):

- *month*/*day*/*year*@*hour*:*min*:*sec* (for example, 8/28/2007@10:01:15), with the time optional
- *month*/*day*/*year hour*:*min*:*sec* (for example, “8/28/2007 10:01:15”), with the time optional
- *month day year hour*:*min*:*sec* (for example, “Aug 28 2007 10:01:15”), with the time optional
- Keywords **start**, **end**, or **now** (for the current time)

The date filtering is intended to limit the output to leases that were active during that time. This means that they can begin before the specified start date, as long as they do not end before the start date. They can also not begin after the specified end date. For example, invoking the command:

```
# ./iphist -N user -P password all Aug 28 2008 Dec 31 2008
```

for the following leases:

Lease 1	Begin	Jan 01 2008	End	Jun 30 2008
Lease 2	Begin	Mar 10 2008	End	Sep 01 2008
Lease 3	Begin	Jun 01 2008	End	Sep 30 2008
Lease 4	Begin	Jan 01 2009	End	Mar 10 2009

would return just Lease 2 and Lease 3, because they both end after the specified start date of the query, even though they both begin before that date. The other two are out of range, because they either end before the specified start date or begin after the specified end date of the query.

The values on each line depend on the specific lease object that the DHCP server stores. You can specify the values to include using the **iphist –f** *format* command. The *format* argument is a list of lease attribute names, enclosed in quotation marks with the names separated by commas, that provides the template for the output lines. The default output is *ipaddress*,*client-mac-addr*,*binding-start-time*, *binding-end-time*.

For example:

```
# ./iphist -f "address,client-mac-addr,binding-start-time,binding-end-time" all
```

The output is a sequence of lines terminated with a newline sequence appropriate to the operating system (\n on UNIX or \r\n on Windows). Each line contains data on a single lease record. The format of the lines is generally comma-separated values enclosed in quotation marks. To use a literal backslash (\) or quotation mark (") inside quotation marks, precede each with a single backslash (\). New lines in attributes are printed as **\n**.

Table 22-4 lists some of the common lease object attributes you can include in the output. Also, see the help for the **lease** command. To get a full list, use **iphist -a**.

Table 22-4 IP History Query Output Attributes

Lease Attribute	Description
address	IP address of the lease.
binding-start-time	Start time of the lease binding.
binding-end-time	End time of the lease binding.
client-binary-client-id	Binary form of the client MAC address.
client-dns-name	Latest DNS name of the client known by the DHCP server.
client-domain-name	Domain where the client resides.
client-flags	A number of client flags.
client-host-name	Hostname that the client requested.
client-id	Client ID requested by or synthesized for the client.
client-last-transaction-time	Date and time when the client most recently contacted the server.
client-mac-addr	MAC address that the client presented to the DHCP server.
client-os-type	Operating system of the leased client.
expiration	Date and time when the lease expires.

Table 22-4 IP History Query Output Attributes (continued)

Lease Attribute	Description
flags	Either reserved or deactivated.
lease-renewal-time	Minimal time that the client is expected to issue a lease renewal.
relay-agent-circuit-id	Contents of the *circuit-id* suboption (1).
relay-agent-option	Contents of the option from the most recent client interaction.
relay-agent-remote-id	Contents of the *remote-id* suboption (2).
relay-agent-server-id-override	IP address in the *server-id-override* suboption.
relay-agent-subnet-selection	IP address in the *subnet-selection* suboption.
relay-agent-vpn-id	Contents of the *vpn-id* suboption.
start-time-of-state	Date and time when the lease changed its state.
state	One of available, expired, leased, offered, or unavailable.
vendor-class-id	Vendor class ID requested by the client.
vpn-id	Identifier for the VPN, if any.

Trimming Lease History Data

If you enabled IP history trimming at the regional cluster, the IP history database is automatically trimmed so that you can reclaim disk space. Each history record has an expiration time. Trimming is necessary for the DHCP server itself, as well as for the CCM regional server that polls the DHCP server for history data.

The CCM server performs background trimming at the regional cluster, which trims off the lease history data older than a certain age at regular intervals. The trimming interval is set by default to 24 hours, and the age (how far back to go in time before trimming) to 24 weeks. The DHCP server at the local cluster performs daily automatic trimming (at 3:00 A.M. local time), and stores four weeks of data by default.

Regional Web UI

To trim lease history data, you must be a central configuration administrator:

Step 1 Click **Servers** to open the Manage Servers page.

Step 2 Click the **Local CCM Server** link to open the Edit CCM Server page.

Step 3 Under Lease History Settings, set the following attributes (you can use the **s**, **m**, **h**, **d**, **w**, **m**, or **y** suffix with values you enter):

- ***trim-lease-hist-interval***—How often to trim the old lease history data automatically, the default being daily. If set to 0, no automatic lease trimming occurs, which is not recommended due to the increasing disk space used. The bounded values are 0 to one year.
- ***trim-lease-hist-age***—Provided that the *trim-lease-hist-interval* is not set to 0, how far back in time to trim the old lease history data automatically, the default being 24 weeks. The bounded values are one day to one year.

Step 4 To force immediate trimming, at the bottom of the page find the Trim/Compact Inputs section (compacting is available only for subnet utilization data). Set the Trim/Compact age to a desired value. This age is how far in time to go back to trim the lease history data. There are no bounds to this value. However, if you set a very small value (such as 1m), it trims or compacts very recent data, which can be undesirable. In fact, if you set it to zero, you lose all of the collected data. Setting the value too high (such as 10y) may end up not trimming or compacting any data.

Step 5 If you are trimming immediately, click **Trim All Lease History**.

You can adjust the trimming that the DHCP server itself performs by setting the *ip-history-max-age* attribute. If *ip-history* is enabled, the DHCP server accumulates database records over time as lease bindings change. This parameter establishes a limit on the age of the history records kept in the database. The server periodically examines the lease history records, establishes an age threshold based on this parameter, and deletes any records that represent bindings that ended before the threshold. The preset value is four weeks.

Running Lease Utilization Reports

Lease utilization reports show the current utilization of address blocks, subnets, and scopes. For both user interfaces, see the "Generating Subnet Utilization History Reports" section on page 9-14.

Local Advanced Web UI

View the current utilization for address blocks, subnets, and scopes from pages in the Address Space function.

CLI Commands

To view lease utilization reports, use **report**.

Receiving Lease Notification

The CLI provides the feature of sending notifications if the number of available IP addresses equals or falls below a certain threshold. The **lease-notification** command specifies, through an *available* attribute, when the notification should occur if the number of available leases reaches or falls below a certain threshold. You can e-mail the report to a user. Although you can use the command interactively, its primary use is in an automated procedure such as a UNIX **cron** task or Windows Scheduled Task.

The following example sets up lease notification for examplescope for when its free addresses fall to 10%. It sends the report to recipients billy, joe, and jane, on a specific Windows mail host:

```
nrcmd> lease-notification available=10% scopes=examplescope recipients=billy,joe,jane
       mail-host=mailhost
```

The output consists of an explanatory header, a table containing a row for each scope in which the number of free addresses is equal to or less than the threshold, and possible warnings related to the scopes and clusters requested.

Network Registrar uses the default cluster and the .nrconfig file by default, unless you specify otherwise. For the command syntax, see the help for the **lease-notification** command.

See Also

Running Lease Notification Automatically in Solaris and Linux
Running Lease Notification Automatically in Windows
Specifying Configuration Files for Lease Notification, page 22-29

Running Lease Notification Automatically in Solaris and Linux

You can run **lease-notification** periodically by means of the **cron(1)** command by supplying **crontab(1)** with the command to run. This example, specified to **crontab**, runs **lease-notification** at 00:15 and 12:15 (15 minutes after midnight and noon), Monday through Friday (note that this encompasses a single command line):

```
15 0,12 * * 1-5 . .profile; /opt/nwreg2/usrbin/nrcmd lease-notification available=10\%
config=/home/jsmith/.nrconfig addresses=192.32.1.0-192.32.128.0
recipients=jsmith,jdoe@example.com >/dev/null 2>&1
```

You can perform **crontab** editing by running the UNIX **crontab –e** command. Set your EDITOR environment variable before running the command, unless you want to use **ed(1)**. See the **crontab(1)** man page for additional details.

Note that you must supply the full path of the CLI command on the **crontab** command line. You can determine the full path in your environment with the UNIX **which nrcmd** command.

Also, when you run the **lease-notification** command by means of **crontab**, the **nrcmd** command ignores the user environment variables CNR_CLUSTER, CNR_NAME, and CNR_PASSWORD. Because other viewers can view the command being run, do not provide the password through the **–P** option on the command line, for security reasons.

Supply the cluster name, user, and password information for the cluster you want the **nrcmd** command to run from in a .profile or other file in the home directory of the user running **crontab –e**. For example:

```
CNR_CLUSTER=host1
export CNR_CLUSTER
CMR_NAME=admin1
export CNR_NAME
CNR_PASSWORD=passwd1
export CNR_PASSWORD
```

The **. .profile** specification in the **crontab** entry explicitly reads the file. The first dot (.) is the shell command that reads the file and you must follow it with at least one space character. For notification on a different cluster (or clusters) than where **nrcmd** is running, specify this information:

- Clusters to check in a config file (see the "Specifying Configuration Files for Lease Notification" section on page 22-29).
- Fully specified path as in the sample **crontab** entry at the beginning of this section.

You can prevent others from examining or changing the contents of the .profile and the configuration file that you create by changing its permissions with the **chmod go-rwx** *config-file* UNIX command.

Running Lease Notification Automatically in Windows

Use the Scheduled Tasks service available in Windows Explorer under My Computer to schedule the **lease-notification** command. If you do not find a Scheduled Tasks folder under My Computer, you need to add this optional component from Microsoft Internet Explorer 4.0 or later, or use some third-party task scheduler. You can also use the **at** command to schedule the **nrcmd lease-notification** command. Put multiple entries in the **at** queue, one for each time of day at which you want to run the job.

Specifying Configuration Files for Lease Notification

If you omit a configuration file, **lease-notification** looks for a default .nrconfig file in your current directory, then in your home directory, and finally in the CNR_INSTALL_PATH/conf directory. Network Registrar uses the first file it encounters. Each line of the file must either begin with the character # (comment), a section header enclosed in square brackets, or a parameter/value pair or its continuation. Network Registrar strips leading space characters from each line and ignores blank lines.

Querying Leases

Network Registrar can work together with Cisco routers to provide enhanced provisioning capabilities. This function is described in the DHCP Leasequery specification (RFC 4388), with which Network Registrar conforms. Part of the implementation of the Cisco uBR access concentrator relay agent is to capture and glean information from DHCP lease requests and responses. It uses this information to:

- Associate subscriber cable modem and client MAC addresses with server-assigned IP addresses.
- Verify source IP addresses in upstream datagrams.
- Encrypt unicast downstream traffic through the DOCSIS Baseline Privacy protocol.
- Avoid broadcasting downstream Address Resolution Protocol (ARP) requests, which can burden the the uBR as well as the subscriber hosts, and which malicious clients can compromise.

The uBR device does not capture all DHCP state information through gleaning. The uBR device cannot glean from unicast messages (particularly renewals and releases) because capturing them requires special processing that would degrade its forwarding performance. Also, this data does not persist across uBR reboots or replacements. Therefore, the only reliable source of DHCP state information for the uBR device is the DHCP server itself.

For this reason the DHCP server supports the DHCPLEASEQUERY message, which is similar to a DHCPINFORM message. Access concentrators and relay agents can thereby obtain client location data directly from the DHCP server, for DHCPv4 and DHCPv6 addresses.

See Also

Leasequery Implementations
Pre-RFC Leasequery for DHCPv4, page 22-30
RFC 4388 Leasequery for DHCPv4, page 22-31
Leasequery for DHCPv6, page 22-31
Leasequery Statistics, page 22-32
Leasequery Example, page 22-32

Leasequery Implementations

Network Registrar provides three Leasequery implementations:

- **DHCPv4 Cisco-proprietary for pre-RFC 4388**—See the "Pre-RFC Leasequery for DHCPv4" section on page 22-30.
- **DHCPv4 compliant with RFC 4388**—See the "RFC 4388 Leasequery for DHCPv4" section on page 22-31.
- **DHCPv6**—See the "Leasequery for DHCPv6" section on page 22-31.

The Cisco-proprietary and the more recent RFC-compliant implementations for DHCPv4 differ in only minor ways and will coexist. The DHCP server accepts Leasequery requests at the same port and returns the specified data for both implementations. The DHCPv6 implementation, new to Network Registrar 7.0, conforms with an IETF draft that is soon to be approved as a DHCPv6 Leasequery RFC.

The DHCP server can include lease reservation data in Leasequery responses for DHCPv4. Network Registrar returns a default lease time of one year (31536000 seconds) for reserved DHCPv4 leases in a response. If the IP address is actually leased, Network Registrar returns its remaining lease time.

Leasequery is preset to be enabled for all the implementations. To disable it, disable an Expert mode attribute, *leasequery.*

Pre-RFC Leasequery for DHCPv4

Leasequery messages usually contain request fields and options. To illustrate, suppose that after a relay agent reboot or replacement, the relay agent receives a request to forward a datagram downstream to the public broadband access network. Because the relay agent no longer has the downstream location data, it sends a LEASEQUERY message to the DHCP server that includes the gateway IP address (*giaddr*) of the relay agent and the MAC address or *dhcp-client-identifier* (option 61) of the target client. If the DHCP server finds the client, it returns the client IP address in the client address (*ciaddr*) field in the response to the leasequery. If the server cannot find the client address, it returns a DHCPNACK.

In the pre-RFC implementation for DHCPv4, the requestor can query by IP address, client ID option (61), or MAC address, and receives from the server a DHCPACK (with the returned data) or a DHCPNACK message, or the server drops the packet. If the request includes multiple query types, the DHCP server responds to the first one it can find. The *giaddr* value from the requestor is independent of the *ciaddr* searched and is simply the return IP address for any responses from the server. The three possible query types are:

- **IP address (*ciaddr*)**—The request packet includes an IP address in the *ciaddr* field. The DHCP server returns data for the most recent client to use that address. A packet that includes a *ciaddr* value must be a request by IP address, despite the values in the MAC address fields (*htype*, *hlen*, and *chaddr*) or *dhcp-client-identifier* option. Querying by IP address is the most efficient method and the one most widely used, in that the other two methods can put more load on the DHCP server.
- ***dhcp-client-identifier* option (61)**—The request packet includes a *dhcp-client-identifier* option value. The DHCP server returns a DHCPACK packet containing the IP address data for the most recently accessed client. If the request omits a MAC address, the server returns all IP addresses and their data for the requested client ID in the *cisco-leased-ip* (also called the *associated-ip*) option. If the request includes the MAC address, the server matches the *dhcp-client-identifier* and MAC address with the client data for the IP address and returns that data in the *ciaddr* field or *cisco-leased-ip* (also called the *associated-ip*) option.
- **MAC address**—The request packet includes a MAC address in the hardware type (*htype*), address length (*hlen*), and client hardware address (*chaddr*) fields, and a blank *ciaddr* field. The server returns all the IP addresses and most recent lease data for the MAC address in the *cisco-leased-ip* (also called the *associated-ip*) option of the reply packet.

The DHCPLEASEQUERY message number in the *dhcp-message-type* option (53) for the pre-RFC implementation is 13. A server that does not support this type of message is likely to drop the packet. The DHCPACK message reply always contains the physical address of the lease owner in the *htype*, *hlen*, and *chaddr* fields. If the request contains the *ciaddr*, the data returned is always based on the *ciaddr* and never the client ID or MAC address.

The requestor can include the *parameter-request-list* option (55) to request specific options about an address. The reply often contains the *dhcp-lease-time* option (51) and the original content of the *relay-agent-info* option (82) that the client sent. If the server does not detect a valid lease for a client, it does not return option 51, and the requestor needs to determine if there is a valid lease.

A DHCPACK from the server can also contain the following Leasequery options:

- ***cisco-leased-ip* (161)**—Data on all the IP addresses associated with the client; also known as (and later renamed) the *associated-ip* option.
- ***cisco-client-requested-host-name* (162)**—Hostname that the client requested in the *host-name* option (12) or *client-fqdn* option (81). The requested hostname was dropped in the RFC 4388 implementation.
- ***cisco-client-last-transaction-time* (163)**—Most recent time duration that a DHCP server contacted the client.

RFC 4388 Leasequery for DHCPv4

Leasequery became an official RFC 4388 for DHCPv4 in February 2006. Network Registrar provides the RFC 4388 implementation alongside the pre-RFC one (see the "Pre-RFC Leasequery for DHCPv4" section on page 22-30) and there are no conflicts between them. However, the RFC 4388 implementation includes a few notable changes:

- The DHCPLEASEQUERY message type contained in the *dhcp-message-type* option (53) changed its message ID to 10 (the ID 13 was given to the DHCPLEASEACTIVE message), and the reply messages were changed from just DHCPACK and DHCPNACK to be more specific:
  - DHCPLEASEQUERY (10) for queries
  - DHCPLEASEUNASSIGNED (11) for replies of unassigned addresses
  - DHCPLEASEUNKNOWN (12) for replies of unknown addresses
  - DHCPLEASEACTIVE (13) for replies of active addresses
- The reply option names and IDs changed, and the *cisco-client-requested-host-name* option was dropped so that there are only two reply options:
  - ***client-last-transaction-time* (91)**—Most recent time duration that a DHCP server contacted the client.
  - ***associated-ip* (92)**—Data on all the IP addresses associated with the client.
- If querying by client ID or MAC address, the request can contain only the *dhcp-client-identifier* option (61) or MAC address; if the packet contains both, the server drops it.

Leasequery for DHCPv6

The Network Registrar Leasequery for DHCPv6 available in Release 7.0 (based on RFC 5007) is designed to:

- Support the source-verify capability of DHCPv6-aware cable modem termination systems (CMTSs).
- Provide data to additional processes that need real-time information about DHCP lease and client activity (such as Cisco Broadband Access Center).
- Implement a pending RFC for DHCPv6 Leasequery.

The message types for DHCPv6 Leasequery are:

- LEASEQUERY (14)
- LEASEQUERY_REPLY (15)

A query can be by:

- QUERY_BY_ADDRESS (1)
- QUERY_BY_CLIENTID (2)

A DHCPv6 LEASEQUERY_REPLY message can contain one or more of the following options:

- ***lq-query* (44)**—Query being performed. The option, used in a request only, includes the query type, link-address (or 0::0), and options to provide data needed for the query.
- ***client-data* (45)**—Encapsulates the data for a single client on a single link. The client data can include any number of these or other requested options.
- ***clt-time* (46)**—Client last transaction time encapsulated in a *client-data* option (45); identifies how long ago (in seconds) the server last communicated with the client.
- ***lq-relay-data* (47)**—Relay agent data used when the client last communicated with the server. Fields are the peer-address and the relay-message. This option can include further options.
- ***lq-client-link* (48)**—Links on which the client has any bindings. Used in reply to a client query when the link-address is omitted and the client is found to be on more than one link.

DHCPv6 uses the Option Reply option (*oro*) to request a list of options in the Leasequery reply.

Leasequery Statistics

As of Network Registrar 7.0, lease queries provide statistics attributes, in the web UI, on the DHCP Server Statistics page (see the "Displaying Statistics" section on page 7-11), and in the CLI by using **dhcp getStats**. The Leasequery statistics are:

- **lease-queries**—Number of RFC 4388 message ID 10 (or pre-RFC message ID 13) DHCPv4 Leasequery packets received in the given time interval.
- **lease-queries-active**—Number of RFC 4388 DHCPLEASEACTIVE packets.
- **lease-queries-unassigned**—Number of RFC 4388 DHCPLEASEUNASSIGNED packets.
- **lease-queries-unknown**—Number of RFC 4388 DHCPLEASEUNKNOWN packets.
- **leasequeries**—Number of DHCPv6 Leasequery packets received.
- **lease-query-replies**—Number of responses to DHCPv6 Leasequery packets that might or might not be successful.

Leasequery Example

Example 22-2 on page 22-33 shows a packet trace of a DHCPv6 query by client ID without a link-address, but with addresses on more than one link. The first part of the output shows the query message and the second part shows the reply data. The *lq-query* option identifies the type of query. Note the list of requested options via the Option Request option (*oro*) in the request, and the two addresses returned in the *lq-client-links* option in the reply.

Example 22-2 Packet Trace of Sample Lease Query

```
>> +- Start of RELAY-FORW message
> > |  msg-type = RELAY-FORW     xid = 0x0       hop-count = 0
> > |  link-address = 1:2:5::
> > |  peer-address = fe80::d02:3ff:fe04:50d
> > |  options:
> > |    relay-message =
> > |      +- Start of LEASEQUERY message
> > |      |  msg-type = LEASEQUERY      xid = 0x10
> > |      |  options:
> > |      |    lq-query = 2 query-by-clientid|::|options:
> > |      |      client-id = duid-ll|1|0f:02:03:04:05:0d
> > |      |      oro = client-id,server-id,ia-na,ia-ta,iaaddr,domain-list,lq-relay-data
> > |      |    client-id = duid-ll|1|01:03:05:07:09:11
> > |      +-  End of LEASEQUERY message
> > |    remote-id = 4491|01:02:03
> > |    relay-agent-subscriber-id = 03:04:05
> > |    interface-id = 05:06:07
> > +-  End of RELAY-FORW message
>
> < +- Start of RELAY-REPL message
> < |  msg-type = RELAY-REPL     xid = 0x0       hop-count = 0
> < |  link-address = 1:2:5::
> < |  peer-address = fe80::d02:3ff:fe04:50d < |  options:
> < |    interface-id = 05:06:07
> < |    relay-message =
> < |      +- Start of LEASEQUERY_REPLY message
> < |      |  msg-type = LEASEQUERY_REPLY        xid = 0x10
> < |      |  options:
> < |      |    server-id = duid-llt|1|5/4/07 12:42 PM|00:0a:e4:3b:42:4c
> < |      |    lq-client-links = 1:2:5:0:955f:926e:b9d3:ca8e,1:2:7:0:9494:8f2a:a9c2:fad1
> < |      +-  End of LEASEQUERY_REPLY message
> < +-  End of RELAY-REPL message
```



CHAPTER 23

Advanced DHCP Server Properties

This chapter describes how to set up some of the more advanced DHCP server properties. Before clients can use DHCP for address assignment, you must add at least one scope to the server. This is described in Chapter 20, "Configuring Scopes and Networks." The additional properties are:

- Configuring BOOTP, page 23-1
- Defining Advanced Server Parameters, page 23-4
- Integrating Windows System Management Servers, page 23-7
- Using Extensions to Affect DHCP Server Behavior, page 23-9
- Tuning the DHCP Server, page 23-12
- Configuring Virtual Private Networks and Subnet Allocation, page 23-14
- Setting DHCP Forwarding, page 23-21

Configuring BOOTP

BOOTP (the BOOTstrap Protocol) was originally created for loading diskless computers. It was later used to allow a host to obtain all the required TCP/IP information to use the Internet. Using BOOTP, a host can broadcast a request on the network and get information required from a BOOTP server. The BOOTP server is a computer that listens for incoming BOOTP requests and generates responses from a configuration database for the BOOTP clients on that network. BOOTP differs from DHCP in that it has no concept of lease or lease expiration. All IP addresses that a BOOTP server allocates are permanent.

You can configure Cisco Network Registrar to act like a BOOTP server. In addition, although BOOTP normally requires static address assignments, you can choose to either reserve IP addresses (and, therefore, use static assignments) or have IP addresses dynamically allocated for BOOTP clients.

See Also

About BOOTP, page 23-2
Enabling BOOTP for Scopes, page 23-2
Moving or Decommissioning BOOTP Clients, page 23-3
Using Dynamic BOOTP, page 23-3
BOOTP Relay, page 23-3

About BOOTP

When you configure the DHCP server to return a BOOTP packet, be aware that BOOTP requires information in the DHCP packet in fields other than the option space. BOOTP devices often need information in the boot file (*file*), server IP address (*siaddr*), and server hostname (*sname*) fields of the DHCP packet (see RFC 2131).

Every Network Registrar DHCP policy has attributes with which you can configure the information you want returned directly in the *file*, *siaddr*, or *sname* fields. The Network Registrar DHCP server also supports a configuration parameter with which you can configure the policy options and determine which of the *file*, *sname*, or *siaddr* values you want returned to the BOOTP device.

Network Registrar supports an analogous configuration parameter with which you can configure the options and *file*, *sname*, or *siaddr* values you want returned to the DHCP client. This is in addition to any options requested by the DHCP clients in the *dhcp-parameter-request* option in the DHCP request. Thus, you can configure both the BOOTP and DHCP response packets appropriately for your devices.

Step 1 Decide which values you want for the BOOTP attributes:

- ***file***—Name of the boot file
- ***siaddr***—Server IP address
- ***sname***—Optional server hostname

Step 2 Decide the list of options and their values that you want returned to the BOOTP client.

Step 3 Set these values in the policy you want associated with the BOOTP request:

- Attributes (*packet-siaddr*, *packet-file-name*, *packet-server-name*) to send to the BOOTP client.
- Option values, such as the server addresses and domain name to return to the BOOTP client.
- List of fields and options you want returned to the BOOTP client.

Step 4 Enable the associated scope or scopes for BOOTP processing.

Step 5 Enable dynamic BOOTP processing if you want to have this scope provide an address for any BOOTP client that requests one. If you do not enable dynamic BOOTP, you must make reservations for each BOOTP client for which you want this scope to provide an address.

Enabling BOOTP for Scopes

You can enable BOOTP processing for a scope. Set certain attributes and BOOTP reply options for a created policy in the local cluster web UI, or use **policy** *name* **create** and **policy** *name* **set** in the CLI, to configure BOOTP. Set the policy attributes and options as a comma-separated list. The attributes are entities to use in a client boot process:

- ***packet-siaddr***—IP address of the next server
- ***packet-file-name***—Name of the boot file
- ***packet-server-name***—Hostname of the server

The server looks through the policy hierarchy for the first instances of these attribute values.

In the CLI, **policy** *name* **setOption** requires spaces (not equal signs) before values.

Also, enable BOOTP and dynamic BOOTP, if desired, and ensure that the DHCP server updates the DNS server with BOOTP requests. The options are:

- Set the option *dhcp-lease-time*.
- Enable the *dynamic-bootp* attribute.
- Enable the *update-dns-for-bootp* attribute.
- Enable the *update-dns-for-bootp* attribute.

Moving or Decommissioning BOOTP Clients

When you move or decommission a BOOTP client, you can reuse its lease. To decommission a BOOTP client, you must remove its lease reservation from the scope and force its lease to be available.

Force the lease to be available in the local cluster web UI, or set **scope** *name* **removeReservation** and **lease** *ipaddr* **force-available** in the CLI.

Using Dynamic BOOTP

When you use dynamic BOOTP, there are additional restrictions placed on the address usage in scopes, because BOOTP clients are allocated IP addresses permanently and receive leases that never expire.

If you are using DHCP failover, when a server whose scope does not have the *dynamic-bootp* option enabled goes into PARTNER-DOWN state, it can allocate any available IP address from that scope, no matter whether it was initially available to the main or backup server. However, when the *dynamic-bootp* option is enabled, the main server and backup servers can only allocate their own addresses. Consequently scopes that enable the *dynamic-bootp* option require more addresses to support failover.

When using dynamic BOOTP:

1. Segregate dynamic BOOTP clients to a single scope. Disable DHCP clients from using that scope. In the local cluster web UI, under the BOOTP attributes for the scope, disable the *dhcp* attribute. In the CLI, use **scope** *name* **disable dhcp**.
2. If you are using DHCP failover, set the *failover-dynamic-bootp-backup-percentage* attribute for the DHCP server to allocate a greater percentage of addresses to the backup server for this scope. This percentage can be as much as 50 percent higher than a regular backup percentage.

BOOTP Relay

Any router that supports BOOTP relay usually has an address that points to the DHCP server. For example, if you are using a Cisco router, it uses the term *IP helper-address*, which contains an address for a specific machine. In this case, use this address to forward all BOOTP (and therefore DHCP) broadcast packets. Be sure that you configure this address on the router closest to your host.



Tip If your DHCP clients are not receiving addresses from the DHCP server, check the network configuration, particularly the router or relay agent configuration, to verify that your network devices are set up to point to your Network Registrar DHCP server address.

Defining Advanced Server Parameters

You can set advanced DHCP server parameters, including custom DHCP options.

See Also

Setting Advanced DHCP Server Parameters
Enabling BOOTP for Scopes, page 23-2
Moving or Decommissioning BOOTP Clients, page 23-3
Using Dynamic BOOTP, page 23-3
BOOTP Relay, page 23-3

Setting Advanced DHCP Server Parameters

Table 23-1 describes the advanced DHCP server parameters that you can set in the local cluster web UI and CLI.

Table 23-1 DHCP Advanced Parameters

Advanced Parameter	Action	Description
max-dhcp-requests	set/ unset	Controls the number of buffers that the DHCP server allocates for receiving packets from DHCP clients and failover partners. If this setting is too large, a burst of DHCP activity can clog the server with requests that become stale before being processed. This results in an increasing processing load that can severely degrade performance as clients try to obtain a new lease, and affects the ability to handle bursts. A low buffer setting throttles requests and could affect server throughput. If the server runs out of buffers, packets are dropped. A good rule or thumb is to increase the buffers if you expect a high load (in a steady state or when experiencing frequent stress times) or you have a fast multiprocessor system. In a nonfailover deployment, the default setting (500) is sufficient. In a failover deployment, you can increase it to 1000 if the DHCP logs indicate a consistently high number of request buffers. You should then also modify the number of DHCP responses (see the *max-dhcp-responses* parameter) to four times the request buffers. When using LDAP client lookups, buffers should not exceed the LDAP lookup queue size defined by the total number of LDAP connections and the maximum number of requests allowed for each connection. Set the LDAP queue size to match the capacity of the LDAP server to service client lookups.

Table 23-1 DHCP Advanced Parameters (continued)

Advanced Parameter	Action	Description
		Some log messages that might trigger a change in this value are: `4493 DHCP ERROR "DHCP has used xx of its yy request buffers: the server is dropping a request."` `4494 DHCP WARNING "DHCP has used xx of yy request packets. Requests will be ignored if no packet buffers are available."` `5270 DHCP WARNING "DHCP has used xx of its yy request buffers: the server is congested -- will not keep the client last-transaction-time to within value but will keep it to within value seconds."` Required. The default is 500.
max-dhcp-responses	set/ unset	Controls the number of response buffers that the DHCP server allocates for responding to DHCP clients and performing failover communication between DHCP partners. In a non-failover deployment, the default setting of twice the number of request buffers is sufficient. In a failover deployment, you can increase this so that it is four times the number of request buffers. In general, increasing the number of response buffers is not harmful, while reducing it to below the previously recommended ratios might be harmful to server responsiveness. Some log messages that might trigger a change in this value are: `4721 DHCP ERROR "DHCP has used all xx response packets. A request was dropped and they will continue to be dropped if no responses are available."` `5289 DHCP WARNING "DHCP has used xx of yy response packets. Requests will be dropped if no responses are available."` Required. The default is 1000.
max-ping-packets	set/ unset	Controls the number of buffers that the server has available to initiate Ping requests to clients. If you enable the *Ping address before offering it* option at the scope level, packet buffers are used to send and receive ICMP messages. If you enable pinging, you should have enough ping packets allocated to handle the peak load of possible ping requests. The default is 500 ping packets.
hardware-unicast	enable/ disable	Controls whether the DHCP server sends unicast rather than broadcast responses when a client indicates that it can accept a unicast. This feature is only available on Windows NT, Windows 2000, and Solaris platforms; other operating systems broadcast instead. The default is enabled.
defer-lease-extensions	enable/ disable	Controls whether the DHCP server extends leases that are less than half expired. This is a performance tuning attribute that helps minimize the number of disk writes to the lease state database. The default is checked or true. This means that a client renewing a lease less than halfway through can get the remaining part of it only and not be extended. See the "Deferring Lease Extensions" section on page 23-6.

Table 23-1 DHCP Advanced Parameters (continued)

Advanced Parameter	Action	Description
last-transaction-time-granularity	set/ unset	Specifies the number of seconds that guarantees the last transaction time to be accurate. This is a performance tuning attribute whereby the server avoids frequent disk writes to the lease state database by ignoring duplicate client activity (such as renews) within the set granularity. Do not set this lower than 30 seconds. For optimal performance, set it to a value that is greater than half the lease interval. The default is 60 seconds.

Local Basic or Advanced Web UI

Step 1 Click **DHCP**, then **DHCP Server** to open the Manage DHCP Server page.

Step 2 Click the name of the server.

Step 3 Add or modify attributes on the Edit DHCP Server page.

Step 4 Click **Modify Server** to make the changes.

CLI Commands

Use **dhcp show** and **dhcp get** to show the current server parameters, then use **dhcp set**, **dhcp unset**, **dhcp enable**, and **dhcp disable** to change them (see Table 23-1 on page 23-4).

Deferring Lease Extensions

Enabling the *defer-lease-extensions* attribute (which is its preset value) allows the DHCP server to optimize response to a sudden flood of DHCP traffic. An example of a network event that could result in such a traffic spike is a power failure at a cable internet service provider (ISP) data center that results in all of its cable modem termination systems (CMTS) rebooting at once. If this happens, the devices attached to the CMTSs produce a flood of DHCP traffic as they quickly come back online.

With the *defer-lease-extensions* attribute enabled, the DHCP server might defer extending the lease expiration time for a client's renewal request, which typically occurs before T1 (usually before halfway through the lease). Instead of giving the client the full configured lease time, the server grants the remaining time on the existing lease. Because the absolute lease expiration time does not change, the server can avoid database updates that result in a significantly higher server throughput. Another benefit is avoiding having to update the failover partner with an extended lease expiration time.

If a client is at or beyond T1 (typically halfway to its expiration), enabling or disabling this attribute has no effect, and the server always tries to extend the lease expiration time. However, failover and other protocol restrictions can prevent the server from extending the lease for the full configured time.

Note Deferring lease extensions significantly increases the server performance while remaining in compliance with the DHCP RFC, which stipulates that client binding information is committed to persistent storage when the lease changes.

When deferring lease extensions, it is advisable to leave the policy attribute *allow-lease-time-override* to its default of disabled, or to change it to disabled if it is enabled.

These three specific situations are described from the server point of view:

- **Client retries**—When the server gets behind, it is possible for a client to retransmit requests. The DHCP server does not maintain enough information to recognize these as retransmissions, and processes each to completion, granting a full lease duration again and updating the database. When the server is already behind, doing extra work worsens the situation. To prevent this, the DHCP server does not extend leases that are less than 30 seconds old, regardless of the state of the *defer-lease-extensions* attribute.
- **Client reboots**—The effective renew time for a client lease is really the minimum of the configured renew time and the time between client reboots. In many installations this may mean that clients get fresh leases one (in a typical enterprise) or two (in a typical cable network) times per day, even if the renew time is set for many days. Setting the *defer-lease-extensions* attribute can prevent these early renews from causing database traffic.
- **Artificially short renewal times**—Because there is no way for a DHCP server to proactively contact a DHCP client with regard to a lease, you might configure short lease times on the DHCP server to provide a means of doing network renumbering, address reallocation, or network reconfiguration (for example, a change in DNS server address) in a timely fashion. The goal is to allow you to do this without incurring unacceptable database update overhead.

As a complication, the server also keeps track of the time when it last heard from the client. Known as the last transaction time, sites sometimes use this information as a debugging aid. Maintaining this time robustly requires a write to the database on every client interaction. The *last-transaction-time-granularity* attribute is the one to set. (See the attribute description in Table 23-1 on page 23-4.) Because it is primarily a debugging aid, the value need not be entirely accurate. Furthermore, because the in-memory copy is always accurate, you can use **export leases –server** to display the current information, even if the data is not up to date in the database.

Integrating Windows System Management Servers

You can have the DHCP server interact with the Microsoft System Management Server (SMS) so that SMS is current with DHCP changes. Normally, SMS pulls updated data through a DHCPDISCOVER request from the server about any new clients that joined the network. Network Registrar, however, pushes these updates to SMS when you use **dhcp updateSms**. Before you do, verify that:

- SMS client installation and initialization step is complete.
- Network Registrar Server Agent is set to run under a login account with sufficient privileges.
- SMS site ID is correct and matches that of the SMS server.

These steps describe how to integrate Windows SMS into Network Registrar.

Step 1 Install the Microsoft BackOffice 4.5 Resource Kit on the same machine as the Network Registrar DHCP server. Follow the installation instructions and choose the default settings.

Step 2 After the installation, modify the User Variable search path on the Environment tab of the System control panel to:

```
\program files\ResourceKit\SMS\diagnose
```

Step 3 If the DHCP and SMS servers are on different machines, install the SMS client on the same machine as the DHCP server. The SMS library has the necessary API calls to communicate with the SMS server. You must assign the correct site code from the DHCP server machine. In your Network Neighborhood, go to the path *SMS-servername*\SMSLOGON\x86.bin\00000409\smsman.exe.

Run the program and follow the instructions, using the default settings. The program creates two icons that you can use later from the control panel, marked SMS and Remote Control.

Step 4 Stop and then restart the Network Registrar server agent under a trusted domain account with sufficient privileges. Both the DHCP and SMS servers must be aware of this account. Use this short procedure:

a. Stop the local cluster server agent process.

b. Configure the account under which the Network Registrar services run. Create an account name that is a member of both the trusted SMS site server group and a member of the DHCP server administrator group, with the corresponding password.

c. Restart the local cluster server agent process.

Step 5 Use **dhcp set sms-library-path** (or the *sms-library-path* attribute under the Microsoft Systems Management Server category on the Edit DHCP Server page) to configure the DHCP server to push lease information to SMS. Include the full path to the SMSRsGen.dll. If you omit a value, the path defaults to the internal server default location of this file. For example:

```
nrcmd> dhcp set sms-library-path /conf/dll
```

When you install the Microsoft BackOffice Resource Kit, the system path is not updated to reflect the location of the SMS data link library (DLL). Use one of these methods to configure this attribute:

a. Set the *sms-library-path* attribute to a relative path:

- First, modify the system PATH variable to append the path of the directory where the DLL is installed:

  ```
  sms-install-directory\diagnose
  ```

- Then, set *sms-library-path* to the name of the DLL, such as smsrsgen.dll. You can also accept the system default by unsetting the attribute.

b. Set *sms-library-path* to an absolute path. If you do not want to change the system path, set this attribute to the absolute path of the DLL location:

```
"\\Program Files\\Resource Kit\\sms\\diagnose\\smsrsgen.dll"
```

Step 6 Set the *sms-network-discovery* DNS attribute to 1 to turn SMS network discovery on.

If you use the default of 0, you disable SMS network discovery.

Step 7 Set the *sms-site-code* DHCP server attribute by entering the SMS site code from Step 3.

The default string is empty, but for data discovery to be successful, you must provide the site code.

Step 8 Set the *sms-lease-interval* attribute to the SMS lease interval.

The lease interval is the time between sending addresses to SMS, or how long, in milliseconds, the DHCP server should wait before pushing the next lease to the SMS server when you run **server dhcp updateSms**. Early versions of the SMSRsGen.dll file (SMS Version 2.0) did not allow SMS to reliably receive multiple updates within a one-second window (1000 ms); the default value, therefore, was set to 1.1 second (1100 ms). If you install a future version of the Microsoft BackOffice Resource Kit, which might contain an enhanced version of the SMSRsGen.dll file, then reduce this interval or set it to 0 to increase performance.

Step 9 If in staged scope edit mode, reload the DHCP server and check the name_dhcp_1_log file.

Step 10 In the CLI, use **server dhcp updateSms** to initiate SMS processing. (This command can take an optional **all** keyword to send all leased addresses from the DHCP server to SMS. If you omit this keyword, the DHCP server sends only new leases activated since the last time the command ran.) Then, verify that both the DHCP and SMS logs indicate successful completion. Note that a server reload during SMS updating interrupts the process, but the process resumes (or restarts) after the server is back up.

Using Extensions to Affect DHCP Server Behavior

Network Registrar provides the ability to alter and customize the operation of the DHCP server through *extensions*, programs that you can write in TCL or C/C++. Extensions interact with the server in two ways: by modifying request or response packets, and through environment variables stored in the environment dictionary (see Chapter 29, "Using Extension Points" for details).

For example, you might have an unusual routing hub that uses BOOTP configuration. This device issues a BOOTP request with an Ethernet hardware type (1) and MAC address in the *chaddr* field. It then sends out another BOOTP request with the same MAC address, but with a hardware type of Token Ring (6). The DHCP server normally distinguishes between a MAC address with hardware type 1 and one with type 6, and considers them to be different devices. In this case, you might want to write an extension that prevents the DHCP server from handing out two different addresses to the same device.

You can solve the problem of the two IP addresses by writing either of these extensions:

- One that causes the DHCP server to drop the Token Ring (6) hardware type packet.
- One that changes the Token Ring packet to an Internet packet and then switches it back again on exit. Although this extension would be more complex, the DHCP client could thereby use either return from the DHCP server.

See Also

Writing Extensions
Preventing Chatty Clients by Using an Extension, page 23-11

Writing Extensions

You can write extensions in TCL or C/C++:

- **TCL**—Makes it a bit easier and quicker to write an extension. If the extension is short, the interpreted nature of TCL does not have a serious effect on performance. When you write an extension in TCL, you are less likely to introduce a bug that can crash the server.
- **C/C++**—Provides the maximum possible performance and flexibility, including communicating with external processes. However, the complexity of the C/C++ API is greater and the possibility of a bug in the extension crashing the server is more likely than with TCL.

You create extensions at specific extension points. Extension points include three types of dictionaries—request, response, and environment. One or more of these dictionaries are available for each of the following extension points:

1. **init-entry**—Extension point that the DHCP server calls when it configures or unconfigures the extension. This occurs when starting, stopping, or reloading the server. This entry point has the same signature as the others for the extension. It is required for DHCPv6 processing. Dictionaries: environment only.
2. **pre-packet-decode**—First extension point that the DHCP server encounters when a request arrives, and calls it before decoding the packet. Dictionaries: request and environment.
3. **post-packet-decode**—Rewrites the input packet. Dictionaries: request and environment.
4. **post-class-lookup**—Evaluates the result of a *client-class-lookup-id* operation on the client-class. Dictionaries: request and environment.
5. **pre-client-lookup**—Affects the client being looked up, possibly by preventing the lookup or supplying data that overrides the existing data. Dictionaries: request and environment.
6. **post-client-lookup**—Reviews the operation of the client-class lookup process, such as examining the internal server data structures filled in from the client-class processing. You can also use it to change any data before the DHCP server does additional processing. Dictionaries: request and environment.
7. **generate-lease**—Generates and controls a DHCPv6 address or prefix. Dictionaries: request, response, and environment.
8. **check-lease-acceptable**—Changes the results of the lease acceptability test. Do this only with extreme care. Dictionaries: request, response, and environment.
9. **lease-state-change**—Determines when the lease state changes this only with extreme care. Dictionaries: response and environment.
10. **pre-packet-encode**—Changes the data sent back to the DHCP client in the response, or change the address to which to send the DHCP response. Dictionaries: request, response, and environment.
11. **post-packet-encode**—Allows the server to examine and alter the packet before it sends the packet to the client, or drops the packet. Dictionaries: request, response, and environment.
12. **pre-dns-add-forward**—Alters the name used for the DNS forward (A record) request. Dictionaries: environment only.
13. **post-send-packet**—Used after sending a packet for processing that you want to perform outside of the serious time constraints of the DHCP request-response cycle. Dictionaries: request, response, and environment.
14. **environment-destructor**—Allows an extension to clean up any context that it might be holding. Dictionaries: environment.

To extend the DHCP server:

Step 1 Write the extension in Tcl, C or C++ and install it in the server extensions directory, on:

- **UNIX**:
  - **Tcl**—/opt/nwreg2/extensions/DHCP/tcl
  - **C or C++**—/opt/nwreg2/extensions/DHCP/dex
- **Windows**:
  - **Tcl**—\program files\Network Registrar\extensions\dhcp\tcl
  - **C or C++**—\program files\Network Registrar\extensions\dhcp\dex

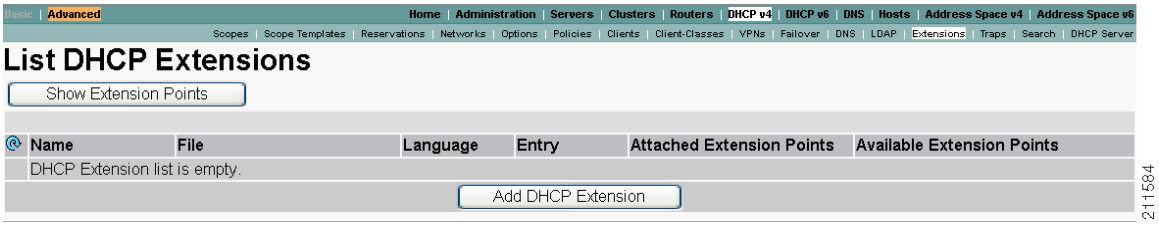
It is best to place these extensions in the appropriate directory for TCL or C/C++ extensions. Then, when configuring the filename, just enter the filename itself, without slash (/) or backslash (\).

If you want to place extensions in subdirectories, enter the filename with a path separator. These are different depending on the operating system on which your DHCP server is running.

Note When entering a filename that contains a backslash (\) character in Windows, you must enter it with a double-backslash (\\), because backslash (\) is an escape character in the CLI. For example, enter the filename debug\myextension.tcl as **debug\\myextension.tcl**.

Step 2 Use the List DHCP Extensions page in the web UI (see Figure 23-1) or the **extension** command in the CLI to configure the DHCP server to recognize this extension.

Figure 23-1 List DHCP Extensions Page (Local)



Step 3 Attach the configured extension to one or more DHCP extension points by using **dhcp attachExtension**.

Step 4 Reload the server.

Preventing Chatty Clients by Using an Extension

One example of an effective use of an extension is to protect against clients flooding the server with unnecessary traffic. You can use the ChattyClientFilter extension to keep the server from having to do much of the work of processing these chatty client packets. If you have large numbers of clients in your network, you might want to consider implementing this extension.

The ChattyClientFilter extension is available in the /examples/dhcp/dex directory of the Network Registrar installation, and compiled and ready to use in /extensions/dhcp/dex/dexextension.so or /extensions/dhcp/dex/dexextension.dll. The extension monitors client requests, based on the MAC address, and disables the client if it generates more than a certain number of packets in a time interval. Disabling a client means that the server discards packets from it. However, the server does not ignore the client entirely, because it continues to monitor traffic from it. If the server detects that the client starts to generate fewer than a certain number of packets in a time interval, it reenables the client and begins to honor packets from it again.

The criteria for disabling and reenabling are set through arguments to the ChattyClientFilter extension. By default, the server disables a client when it receives more than 15 packets within 30 seconds; the server reenables the client when it sends fewer than 5 packets within 10 seconds. Note that these defaults are conservative and do not protect against all situations. For example, the server does not disable a client that sends packets every three seconds. Even allowing for a few retransmissions, a client should never need to send more than a half dozen packets in a short interval.

If you suspect chatty clients, review the DHCP server logs to determine incoming rates, then set the following arguments in the ChattyClientFilter code appropriately:

- **–h *packet-count***—SampleHitsToDisable; default 15 packets.
- **–i *seconds***— SampleTimeInterval; default 30 seconds.
- **–l *packet-count***—QuietHitsToLeaveDisabled; default 5 packets.
- **–q *seconds***—QuietTimeInterval; default 10 seconds.
- **–s**—Silently discards dropped packets; default off.
- **–n**—NAKs the client if renewing or rebinding; default off. If the client exceeding the SampleHitsToDisable rate does a DHCPREQUEST, the server sends it a DHCPNAK instead of discarding the packet. This can fix problems with some clients (such as cable modems) that cannot renew leases for some reason. Sending the DHCPNAK causes the client to restart its DHCP state machine and send a DHCPDISCOVER. If you use this argument, be sure also to attach the ChattyClientFilter to the **check-lease-acceptable** extension point.
- **–r *seconds***—Statistics reporting interval; default 300 seconds (5 minutes). This argument controls the frequency of periodic logging of the number of clients disabled and reenabled.

Review the comments in the ChattyClientFilter.cpp file for how to set the arguments and enable the extension. In most cases, you would attach it to the **post-packet-decode** extension point (along with **check-lease-acceptable** if you use the **–n** argument).

Tuning the DHCP Server

Other helpful hints in tuning your DHCP performance include:

- Set the request (*max-dhcp-requests*) and response (*max-dhcp-responses*) buffers for optimal throughput. See Table 23-1 on page 23-4 for details.
- Keep the *defer-lease-extensions* attribute enabled. This reduces writes to the database.
- Set the *last-transaction-time-granularity* attribute to at least 60 seconds, optimally a value greater than half your lease interval.
- Disable the *allow-lease-time-override* attribute for policies offering production leases.
- Minimize your logging and debugging settings. If you require logging, use the *log-settings* attribute for the DHCP server with a controlled number of attributes, as described in Table 23-2.

Table 23-2 DHCP Log Settings

Log Setting (Numeric Equivalent)	Description
default (1)	Displays basic DHCP activity logging (the default setting).
incoming-packets (2)	Logs a separate line for each incoming DHCP packet (the default).
missing-options (3)	Displays missing policy options expected by a client (the default).
incoming-packet-detail (4)	The same as *incoming-packets*, but in human-readable form.
outgoing-packet-detail (5)	Logs each incoming DHCP packet in a human-readable form.
unknown-criteria (6)	Logs whenever a client entry has a *selection-criteria* or *selection-criteria-excluded* that is not found in any scope appropriate for the current network location of that client.

Table 23-2 DHCP Log Settings (continued)

Log Setting (Numeric Equivalent)	Description
dns-update-detail (7)	Logs each sent and replied DNS update.
client-detail (8)	After every client-class client lookup operation, logs the composite of the data found for the client and its client-class. Useful when setting a client-class configuration and debugging problems in client-class processing.
client-criteria-processing (9)	Logs whenever a scope is examined to find an available lease or to determine if a lease is still acceptable for a client who already has one. Can be very useful when configuring or debugging client-class scope criteria processing. (Causes moderate amount of information to be logged and should not be left enabled as a matter of course.)
failover-detail (10)	Logs detailed failover activity.
ldap-query-detail (11)	Logs whenever the DHCP server initiates a query to an LDAP server, receives a response, and retrieves a result or error messages.
ldap-update-detail (12)	Logs whenever the DHCP server initiates an update lease state to the LDAP server, receives a response, and retrieves a result or error messages.
ldap-create-detail (13)	Logs whenever the DHCP server initiates a lease state entry create to the LDAP server, receives a response, and retrieves a result or error messages.
leasequery (14)	Logs a message for every ACK- or NAK-responded lease query packet.
dropped-waiting-packets (15)	If the value of *max-waiting-packets* is non-zero, packets can be dropped if the queue length for any IP address exceeds the value. If *dropped-waiting-packets* is set, the server logs whenever it drops a waiting packet from the queue for an IP address.
no-success-messages (16)	Inhibits logging successful outgoing response packets.
no-dropped-dhcp-packets (17)	Inhibits logging dropped DHCP packets.
no-dropped-bootp-packets (18)	Inhibits logging dropped BOOTP packets.
no-failover-activity (19)	Inhibits logging normal activity and some warning messages logged for failover. However, serious error log messages continue to appear.
activity-summary (20)	Enables logging a summary message every five minutes (useful if the following **no-** type flags are set), showing the activity in the previous interval (you can adjust this interval using **dhcp set activity-summary-interval**).
no-invalid-packets (21)	Inhibits logging invalid packets.
no-reduce-logging-when-busy (22)	Inhibits reducing logging when receive buffers reach 66%.
no-timeouts (23)	Inhibits logging timeouts of leases and offers.
minimal-config-info (24)	Reduces the number of configuration messages in the log.
no-failover-conflict (25)	Logs conflicts between failover partners.
atul-detail (26)	Logs messages when the DHCP server receives Address-to-User Lookup (ATUL) packets from an ATUL protocol server.

- Adjust the *mcd-blobs-per-bulk-read* attribute value to tune DHCP start and reload times. Generally, a higher *mcd-blobs-per-bulk-read* attribute value results in faster server start and reload times, at the cost of using more memory. Values can be set to any number between 1 and 2500 using the *mcd-blobs-per-bulk-read* DHCP server attribute. The current default is 256 blobs.
- Consider setting client caching (see the "Setting Client Caching Parameters" section on page 24-14).
- Check the server statistics to aid in monitoring server performance (see the "Displaying Statistics" section on page 7-11).
- Consider setting the scope allocation priority (see the "Configuring Multiple Scopes Using Allocation Priority" section on page 20-12).
- If pinging hosts before offering addresses, consider adjusting the ping timeout period (see the "Pinging Hosts Before Offering Addresses" section on page 22-7).
- To boost performance, consider limiting the number of scope-selection tags.
- If using Lightweight Directory Access Protocol (LDAP) servers, consider the performance issues described in the "Configuring Network Registrar to Use LDAP" section on page 24-19.
- If using DHCP failover, consider using the load balancing feature (see the "Setting Load Balancing" section on page 27-19).

Tip Be sure to follow any DHCP server attribute changes with a server reload.

Configuring Virtual Private Networks and Subnet Allocation

This section describes how to configure the Cisco Network Registrar DHCP server to support virtual private networks (VPNs) and subnet allocation for on-demand address pools.

Configuring VPNs involves an adjustment to the usual DHCP host IP address designation. VPNs use private address spaces that might not be unique across the Internet. Because of this, Network Registrar supports IP addresses that are distinguished by a VPN identifier. Relay agents on routers must support this capability as well. The VPN identifier selects the VPN to which the client belongs. VPN for DHCP is currently only supported by Cisco IOS software, the newest versions of which can include VPN IDs in the relayed DHCP messages.

Subnet allocation is a way of leasing subnets to clients (usually routers or edge devices) so that they can, in turn, provide DHCP services. This can occur along with or instead of managing individual client addresses. Subnet allocation can vastly improve IP address provisioning, aggregation, characterization, and distribution by relying on the DHCP infrastructure to dynamically manage subnets. Subnet allocation through DHCP is currently only supported by Cisco IOS software, the newest versions of which incorporate the on-demand address pools feature.

See Also

Configuring Virtual Private Networks Using DHCP
Configuring DHCP Subnet Allocation, page 23-19
VPN and Subnet Allocation Tuning Parameters, page 23-21

Configuring Virtual Private Networks Using DHCP

VPNs that you create provide a filtering mechanism for:

- Viewing the unified address space (see the "Viewing Address Space" section on page 9-2).
- Listing address blocks (see the "Adding Address Blocks" section on page 9-5).
- Listing subnets (see the "Address Blocks and Subnets" section on page 9-4).
- Querying subnet utilization (see the "Generating Subnet Utilization History Reports" section on page 9-14).
- Querying lease history (see the "Running IP Lease Histories" section on page 22-21).

If you do not configure a VPN, Network Registrar uses the global VPN of 0 on each scope.

To configure a VPN whereby a client can request IP addresses from a DHCP server using a relay agent, you must define the VPN and associate a scope with it. Specifically:

1. Ensure that the relay agents that handle DHCP VPN traffic are configured with a version of Cisco IOS software that supports the *vpn-id* suboption of the *relay-agent-info* option (82) in DHCP.
2. Coordinate with the Cisco IOS relay agent administrator that the VPN is identified either by a VPN ID or a VPN Routing and Forwarding instance (VRF) name.
3. Create a scope for the VPN.

See Also

Typical Virtual Private Networks
Creating and Editing Virtual Private Networks, page 23-16
VPN Usage, page 23-18

Typical Virtual Private Networks

Figure 19-4 on page 19-5 shows a typical VPN scenario with DHCP client 1 as part of VPN blue and DHCP client 2 in VPN red. For example, both DHCP client 1 in VPN blue and client 2 in VPN red have the same private network address: 192.168.1.0/24. The DHCP relay agent has gateway addresses that are in the two VPNs as well as a global one (172.27.180.232). There are two failover DHCP servers, both of which know the relay agent through its external gateway address.

Here is the processing that takes place for the server to issue a VPN-supported address to a client:

1. DHCP client 1 broadcasts a DHCPDISCOVER packet, including its MAC address, hostname, and any requested DHCP options.
2. DHCP relay agent at address 192.168.1.1 picks up the broadcast packet. It adds a *relay-agent-info* option (82) to the packet and includes the *subnet-selection* suboption that identifies 192.168.1.0 as the subnet. The packet also includes the *vpn-id* suboption that identifies the VPN as *blue*. Because the DHCP server cannot communicate directly with the requesting client, the *server-id-override* suboption contains the address of the relay agent as known by the client (192.168.1.1). The relay agent also includes in the packet its external gateway address (*giaddr*), 172.27.180.232.
3. The relay agent unicasts the DHCPDISCOVER packet to the configured DHCP server on its subnet.
4. DHCP server 1 receives the packet and uses the *vpn-id* and *subnet-selection* suboptions to allocate an IP address from the proper VPN address space. It finds the available address 192.168.1.37 in the subnet and VPN, and places it in the *yiaddr* field of the packet (the address offered to the client).
5. The server unicasts a DHCPOFFER packet to the relay agent that is identified by the *giaddr* value.

6. The relay agent removes the *relay-agent-info* option and sends the packet to DHCP client 1.
7. DHCP client 1 broadcasts a DHCPREQUEST message requesting the same IP address that it was offered. The relay agent receives this broadcast message.
8. The relay agent forwards the DHCPREQUEST packet to DHCP server 1, which replies with a unicast DHCPACK packet to the client.
9. For a lease renewal, the client unicasts a DHCPRENEW packet to the IP address found in the *dhcp-server-identifier* option of the DHCPACK message. This is 192.168.1.1, the address of the relay agent. The relay agent unicasts the packet to the DHCP server. The server does its normal renewal processing, without necessarily knowing whether it was the server that gave out the original address in the first place. The server replies in a unicast DHCPACK packet. The relay agent then forwards the DHCPACK packet to the client IP address identified by the *ciaddr* field value.

   If the *server-id-override* suboption of the *relay-agent-info* option (82) exists, the DHCP server uses its value to compare to that of the *dhcp-server-identifier* option in the reply packet. Any packet that the DHCP client unicasts then goes directly to the relay agent and not to the server (which may, in fact, be inaccessible from the client). Both partners in a failover environment can renew a lease if the packet includes the *server-id-override* suboption.

Creating and Editing Virtual Private Networks

To set up the VPN and its index:

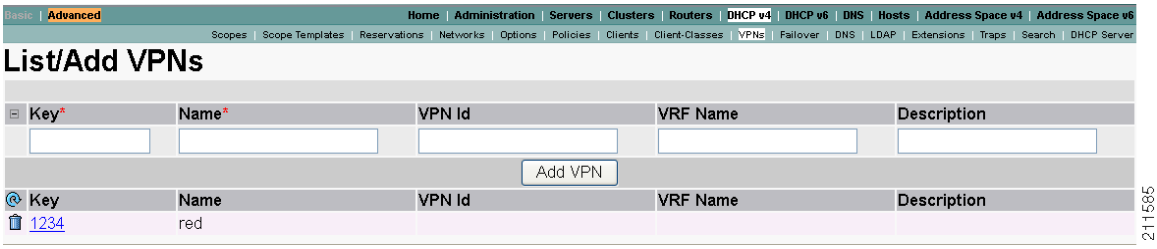
Step 1 Coordinate with the Cisco IOS relay agent administrator that the VPNs are configured either by VPN ID or VRF name on the relay agent. This will determine how to identify the VPN in Network Registrar.

Step 2 Create a VPN to allow provisioning DHCP clients onto the VPN that is configured in the IOS switch or router.

Enter a VPN index, which can be any unique text string except the reserved words **all** or **global**. Its associated ID must also be unique. To add an index at the:

- **Local cluster**—Click **DHCP**, then **VPNs** to open the List/Add VPNs page (see Figure 23-2). Give the VPN a numerical key identifier and a unique name in the cluster.

Figure 23-2 List/Add VPNs Page (Local)



- **Regional cluster**—Add the local cluster containing the VPN (click **Clusters**, then **Cluster List**). Then, click **DHCP Configuration** followed by **VPNs**. This opens the List/Add VPNs page. You can create the VPN on this page or pull the VPN from the local clusters:
  - If creating the VPN, give it a numerical key identifier and a unique name.
  - If pulling the VPN from the local clusters, click **Pull Replica VPNs** on the List/Add VPNs page, then pull a specific VPN or all the VPNs from the selected cluster.

You can also push VPNs to the clusters by clicking **Push VPN** or **Push All VPNs** on the List/Add VPNs page. Then choose the synchronization mode and the clusters to which to push the VPNs on the Push VPN Data to Local Clusters page.

- **In the CLI**—Use **vpn** *name* **create** *key*. For example:

```
nrcmd> vpn blue create 99
```

Step 3 Specify the appropriate VPN identifier, either by VPN ID or VRF name. It is rarely both.

- If you use a VPN ID, set the *vpn-id* attribute value for the VPN. The value is usually in hexadecimal, in the form *oui*:*index*, per IETF RFC 2685. It consists of a three-octet VPN Organizationally Unique Identifier (OUI) that corresponds to the VPN owner or ISP, followed by a colon. It is then followed by a four-octet index number of the VPN itself. Add the VPN ID value to the List/Add VPNs page. In the CLI, set the *vpn-id* attribute. For example:

```
nrcmd> vpn blue set vpn-id=a1:3f6c
```

- If you use a VPN Routing and Forwarding (VRF) instance name, set the *vrf-name* attribute value for the VPN. Cisco routers frequently use VRF names. Add the VRF Name value to the List/Add VPNs page. In the CLI, set the *vrf-name* attribute. For example:

```
nrcmd> vpn blue set vrf-name=framus
```

Step 4 Add a description for the VPN (optional).

Step 5 Click **Add VPN**. You can edit the VPN to change the values on the Edit VPN page.

Step 6 Create a scope for the VPN. Keep the VPN name and scope name as similar as possible for identification purposes:

- Click **DHCP**, then **Scopes** to open the List/Add DHCP Scopes page. Create a scope or edit an existing one. Under the Miscellaneous attributes, look for the *vpn-id* attribute. Choose the VPN from the drop-down list.
- In the CLI, identify to which VPN the scope belongs in one of three ways:
  - Its VPN name, through the *vpn* attribute (which applies the VPN ID to the scope).
  - The VPN ID itself, through the *vpn-id* attribute.
  - The current session VPN name, by omitting the VPN or its ID on the command line.

  You set the default VPN for the current session using **session set current-vpn**. You can then set the usual address range and necessary option properties for the scope. For example:

```
nrcmd> scope blue-1921681 create 192.168.1.0 255.255.255.0 vpn=blue
```

  Or:

```
nrcmd> scope blue-1921681 create 192.168.1.0 255.255.255.0 vpn-id=99
```

  Or:

```
nrcmd> session set current-vpn=blue
nrcmd> scope blue-1921681 create 192.168.1.0 255.255.255.0
```

  Then:

```
nrcmd> scope blue-1921681 addRange 192.168.1.101 192.168.1.200
nrcmd> scope-policy blue-1921681 setOption routers 192.168.1.1
```

Step 7 If in staged scope edit mode, reload the DHCP server after you create all the VPNs and scopes.

VPN Usage

The VPN name is used to qualify many DHCP objects in Network Registrar, such as IP addresses (leases), scopes, and subnets. For example, lease names can have this syntax:

vpn/*ipaddress*

For example, red/192.168.40.0

A VPN can be any unique text string except the reserved words **global** and **all**. You can use **global** and **all** when you export address or lease data. The **global** VPN maps to the [none] VPN; the **all** VPN maps to both the specific VPN and the [none] VPN.

In the CLI, if you omit the VPN or its ID in defining an object, the VPN defaults to the value set by **session set current-vpn**. In the web UI, if the current VPN is not defined, it defaults to the [none] VPN, which includes all addresses outside of any defined VPNs.

These objects have associated VPN properties:

- **Address blocks**—Define the VPN for an address block.
  - Click **Address Space**, then **Address Blocks**. On the List/Add Address Blocks page, choose the VPN from the Select VPN drop-down list.
  - In the CLI, use the **dhcp-address-block** creation and attribute setting commands. For example:

```
nrcmd> dhcp-address-block red create 192.168.50.0/24
nrcmd> dhcp-address-block red set vpn=blue
nrcmd> dhcp-address-block red set vpn-id=99
```

- **Clients and client-classes**—In some cases it is best to provision a VPN inside of Network Registrar instead of externally, where it might have to be configured for every Cisco IOS device. To support this capability, you can specify a VPN for a client or client-class. Two attributes are provided:
  - *default-vpn*—VPN that the packet gets if it does not already have a *vpn-id* or *vrf-name* value in the incoming packet. You can use the attribute with clients and client-classes.
  - *override-vpn*—VPN the packet gets no matter what is provided for a *vpn-id* or *vrf-name* value in the incoming packet. You can use the attribute with clients and client-classes. Note that if you specify an override VPN on the client-class, and a default VPN for the client, the override VPN on the client-class takes precedence over the default VPN on the client.

  At the local cluster—Click **DHCP**, then **Client-Classes**. Create or edit a client-class and enter the *default-vpn* and *override-vpn* attribute values.

  At the regional cluster—Click **DHCP Configuration**, then **Client-Classes**. Create or pull, and then edit a client-class to enter the *default-vpn* and *override-vpn* attribute values.

  In the CLI—Use the **client-class** creation and attribute setting commands. For example:

```
nrcmd> client 1,6,00:d0:ba:d3:bd:3b set default-vpn=blue
nrcmd> client-class CableModem set override-vpn=blue
```

  In a cable modem deployment, for example, you can use the *override-vpn* attribute to provision the cable modems. The client-class would determine the scope for the cable modem, and the scope would determine the VPN for the uBR. User traffic through the cable modem would then have the *vpn-id* suboption set and use the specific VPN. The *override-vpn* value also overrides any *default-vpn* set for the client.

- **Leases**—List leases, show a lease, or get lease attributes.

  In the CLI—To import leases, use **import leases** *filename*. Each lease entry in the file can include the VPN at the end of the line. If it is missing, Network Registrar assigns the [none] VPN. (See also the "Importing and Exporting Lease Data" section on page 22-5.)

  ```
  nrcmd> import leases leaseimport.txt
  ```

  To export the address or lease data to include the VPN, use **export addresses** with the *vpn* attribute, or **export leases** with the **–vpn** option. The VPN value can be the reserved word **global** or **all**:

  - **Global**—Any addresses outside the defined VPNs (the [none] VPN).
  - **All**—All VPNs, including the [none] VPN.

  If you omit the VPN, the export uses the current VPN as set by **session set current-vpn**. If the current VPN is not set, the server uses the [none] VPN.

  ```
  nrcmd> export addresses file=addrexport.txt vpn=red
  nrcmd> export leases -server -vpn red leaseexport.txt
  ```

- **Scopes**—Scopes can include the VPN name or its ID, as described in the "Creating and Editing Virtual Private Networks" section on page 23-16.
  - **At the local cluster**—Click **DHCP**, then **Scopes**. Create or edit a scope and set the Miscellaneous attribute *vpn-id*.
  - **At the regional cluster**—Click **DHCP Configuration**, then **Scope Templates**. Create or pull, and then edit a scope template to set the Miscellaneous attribute *vpn-id*.
  - **In the CLI**—Use the **scope** creation and attribute setting commands. For example:

    ```
    nrcmd> scope examplescope1 set vpn=blue
    nrcmd> scope examplescope1 set vpn-id=99
    ```

- **Subnets**—Listing subnets, showing a subnet, or getting the *vpn* or *vpn-id* attribute for a subnet shows the VPN. See the "Configuring DHCP Subnet Allocation" section on page 23-19.
- **DHCP server**—If the *vpn-communication* attribute is enabled (which it is by default), the DHCP server can communicate with DHCP clients that are on a different VPN from that of the DHCP server by using an enhanced DHCP relay agent capability. This capability is indicated by the *server-id-override* suboption in the relay agent information option (82).

Configuring DHCP Subnet Allocation

The following section provides an example of setting up subnet allocation using the DHCP server. Figure 19-5 on page 19-6 shows a sample subnet allocation configuration with subnets assigned to provisioning devices, along with the conventional DHCP client/server configuration.

Before allocating subnets, the DHCP server first determines what VPN the client is on, in the following order:

1. The server looks for incoming VPN options and uses the value for the VPN.
2. If no VPN options are found, the server uses the relay agent suboption value, then combines the VPN with the subnet address to form the unique identifier.
3. If no relay agent suboption is found, the server looks for client-class information (selection tags).

Step 1 Create a DHCP address block for a subnet, set the initial subnet mask and its increment, and set other subnet allocation request attributes. Also, associate a policy or define an embedded policy.

- If you use VPNs, you can specify a *vpn* or *vpn-id* attribute (see the "Configuring Virtual Private Networks Using DHCP" section on page 23-15).

  Note Unsetting the VPN ID in the CLI reverts the value to the current session VPN.

- The server uses the presence of the *subnet-alloc* DNS option (220) in the request packet to determine that the packet is a subnet allocation request. You can configure the server to use the *subnet-name* suboption (3) as a selection tag if you set the *addr-blocks-use-selection-tags* attribute for the server or VPN.
- You can optionally set a default selection tag by setting the *addr-blocks-default-selection-tags* attribute for the DHCP server or VPN object. This identifies one or more subnets from which to allocate the addresses. If the relay agent sends a VPN string (via a VPN option or relay agent suboption), associated with a subnet, any address block with that string as one of its *addr-blocks-default-selection-tags* values uses that subnet.
- The default behavior on the server and for VPNs is that the DHCP server tries to allocate subnets to clients using address blocks that the clients already used. Disabling the *addr-blocks-use-client-affinity* attribute causes the server to supply subnets from any suitable address block, based on other selection data in the clients' messages.
- If you want to support configurations of multiple address blocks on a single LAN segment (analogous to using primary and secondary scopes), add a *segment-name* attribute string value to the DHCP address block. When the relay agent sends a single subnet selection address, it selects address blocks tagged with that *segment-name* string value. However, you must also explicitly enable the LAN segment capability (*addr-blocks-use-lan-segments*) at the server or VPN level.
- Instead of associating a policy, you can set properties for the address block embedded policy. As in embedded policies for clients, client-classes, and scopes, you can enable, disable, set, unset, get, and show attributes for an address block policy. You can also set, unset, get, and list any DHCP options for it, as well as set, unset, and list vendor options. Note that deleting an address block embedded policy unsets all the embedded policy properties.

Step 2 Note that the server allocates subnets based on the relay agent request. If not requested, the default subnet size is a 28-bit address mask. You can change this default, if necessary, by setting the *default-subnet-size* attribute for the DHCP address block. For example:

```
nrcmd> dhcp-address-block red set default-subnet-size=25
```

Step 3 You can control any of the subnets the DHCP server creates from the address blocks. Identify the subnet in the form *vpn-name*/*netipaddress*/*mask*, with the *vpn-name* optional. Subnet control includes activating and deactivating the subnet as you would a lease. Likewise, you can force a subnet to be available, with the condition that before you do so, that you check that the clients assigned the subnet are no longer using it. First, show any subnets created.

Step 4 If in staged scope edit mode, reload the DHCP server.

VPN and Subnet Allocation Tuning Parameters

Consider these tuning parameters for VPNs and on-demand address pools.

- **Keep orphaned leases that have nonexistent VPNs**—Network Registrar usually maintains leases that do not have an associated VPN in the Network Registrar state database. You can change this by enabling the DHCP attribute *delete-orphaned-leases*. The server maintains a lease state database that associates clients with leases. If a scope modification renders the existing leases invalid, the lease database then has orphaned lease entries. These are typically not removed even after the lease expires, because the server tries to use this data in the future to reassociate a client with a lease. One downside to this is that the lease database may consume excessive disk space. When you enable the *delete-orphaned-leases* attribute, such lease database entries are removed during the next server reload. However, be cautious when enabling this attribute, because rendering leases invalid can result in clients using leases that the server believes to be free. This can compromise network stability.
- **Keep orphaned subnets that have nonexistent VPNs or address blocks**—This is the default behavior, although you can change it by enabling the DHCP attribute *dhcp enable delete-orphaned-subnets*. As the DHCP server starts up, it reads its database of subnets and tries to locate the parent VPN and address block of each subnet. With the attribute enabled, if a subnet refers to a VPN that is no longer configured in the server, or if the server cannot locate a parent address block that contains the subnet, the server permanently deletes the subnet from the state database.
- **Keep the VPN communication open**—This is the default behavior, although you can change it by disabling the DHCP attribute *vpn-communication*. The server can communicate with clients that reside on a different VPN from that of the server by using an enhanced DHCP relay agent capability. This is signaled by the appearance of the *vpn-id* suboption of the *relay-agent-info* option (82). You can disable the *vpn-communication* attribute if the server is not expected to communicate with clients on a different VPN than the server. The motivation is typically to enhance network security by preventing unauthorized DHCP client access.

Setting DHCP Forwarding

The Network Registrar DHCP server supports forwarding DHCP packets to another DHCP server on a per-client basis. For example, you might want to redirect address requests from certain clients, with specific MAC address prefixes, to another DHCP server. This can be useful and important in situations where the server being forwarded to is not one that you manage. This occurs in environments where multiple service providers supply DHCP services for clients on the same virtual LAN.

Enabling DHCP forwarding requires implementing an extension script. The DHCP server intercepts the specified clients and calls its forwarding code, which checks the specified list of forwarded server addresses. It then forwards the requests rather than processing them itself. You attach and detach extensions to and from the DHCP server by using **dhcp attachExtension** and **dhcp detachExtension**.

The DHCP forwarding feature works like this:

1. When DHCP is initialized, the server opens a UDP socket, which it uses to send forwarded packets. To support servers with multiple IP addresses, the socket address pair consists of INADDR_ANY and any port number. This enables clients to use any one of the server IP addresses.
2. When the DHCP server receives a request from a client, it processes these extension point scripts:
   - **post-packet-decode**
   - **pre-client-lookup**
   - **post-client-lookup**

As the DHCP server processes these scripts, it checks the environment dictionary for this string:

```
cnr-forward-dhcp-request
```

3. When it finds that string and it has the value *true* (enabled), the server calls its forwarding code.
4. The forwarding code checks the environment dictionary for a string with this key:

   ```
   cnr-request-forward-address-list
   ```

   It expects a list of comma-separated IP addresses with an optional colon-delimited port number, as in this example:

   ```
   192.168.168.15:1025,192.168.169.20:1027
   ```

   By default, the server forwards to port 67. It sends a copy of the entire client request to each IP address and port in turn. If any element in the list is invalid, the server stops trying to parse the list.
5. After the forwarding code returns, the server stops processing the request. In the post-client-lookup extension point script, however, this action might create an optional log message with client-entry details.

The following example of a portion of a TCL extension script tells the DHCP server to forward a request to another server based on the information in the request. You can use such a script if there are multiple device provisioning systems in the same environment. In this case, you would run the extension script on the DHCP server to which routers forward broadcast requests. The script would determine which (if any) other server or servers should handle the request, and tell the original server to forward the request.

The sample script uses a static mapping of MAC address prefix to send modems from a specific vendor to a specific system:

```
proc postPktDecode {req resp env} {
    set mac [$req get chaddr]
    set addrs ""
    ;# Very simple, static classifier that forwards all requests from devices
    ;# with a vendor-id of 01:0c:10 to the DHCP servers at 10.1.2.3 and 10.2.2.3:
    switch -glob -- $mac {
        01:0c:10* {
            set addrs "10.1.2.3,10.2.2.3"
        }
    }
    ;# If we decide to forward the packet, the $addrs var will have the IP addresses
    ;# where to forward the packet:
    if {$addrs != ""} {
        ;# Tell the DHCP server to forward the packet...
        $env put cnr-forward-dhcp-request true
        ;# ...and where to forward it:
        $env put cnr-request-forward-address-list $addrs
        ;# No more processing is required.
        return
    }
}
```

A more flexible script could use a per-client configuration object, such as the Network Registrar client entry, to indicate which DHCP server should get the request.



CHAPTER 24

Configuring Client-Classes and Clients

Use the Network Registrar client and client-class concepts to provide differentiated services to users across a common network. You can group clients based on administrative criteria, and then ensure that each group receives its appropriate class of service (COS). Without client-class processing, the DHCP server provides client leases based solely on their network location.

See Also

Configuring Client-Classes
Configuring Clients, page 24-9
Subscriber Limitation Using Option 82, page 24-14
Configuring Network Registrar to Use LDAP, page 24-19

Configuring Client-Classes

You can differentiate client services in a number of ways:

- Register clients using the Network Registrar database (this section) or the Lightweight Directory Access Protocol (see the "Configuring Network Registrar to Use LDAP" section on page 24-19).
- Register intermediary devices (such as cable modems) so that you can differentiate their upstream clients by class of service.
- Use the contents of client packets without the foreknowledge of client data:
  - Known DHCP options that can be in the packet, such as the *dhcp-user-class-id* DHCP option (77), or the *radius-attribute* suboption of the *relay-agent-info* DHCP option (82, see the "Processing Client Data Including External Sources" section on page 24-6).
  - Other data in the packet to extract using an expression in the *client-class-lookup-id* DHCP server attribute (see the "Calculating Client-Classes and Creating Keys" section on page 24-16).
- Use a two-stage process of first creating a client-class to assign clients, then set a *client-lookup-id* for certain clients (see the "Expression Processing for Subscriber Limitation" section on page 24-17).

See Also

Client-Class Process, page 24-2
Defining Client-Classes, page 24-2
Setting Selection Tags on Scopes and Prefixes, page 24-4
Defining Client-Class Hostname Properties, page 24-4
Editing Client-Classes and Their Embedded Policies, page 24-5
Processing Client Data Including External Sources, page 24-6
Troubleshooting Client-Classes, page 24-8

Client-Class Process

Enable or disable client-class processing for the DHCP server and apply a set of properties to groups of clients. With client-class enabled, the server assigns the client to an address from a matching DHCPv4 scope or DHCPv6 prefix. The server acts according to the data in the packet. To configure client-classes:

1. Enable client-class processing for the DHCP server.
2. Define client-classes that include or exclude selection tags (criteria).
3. Apply the selection tags to specific scopes or prefixes (or their templates).
4. Assign clients to these classes.

This process is for clients configured through Network Registrar. For processing affected by data from external sources, see the "Processing Client Data Including External Sources" section on page 24-6.

Defining Client-Classes

You enable and define client-classes at the server level.

Local Basic or Advanced Web UI

Step 1 Enable client-classes. In Basic or Advanced mode:

a. Click **DHCP**, then **DHCP Server** to open the Manage DHCP Server page.

b. Click the name of the server.

c. On the Edit DHCP Server page, enable the *client-class* attribute.

d. Click **Modify Server**.

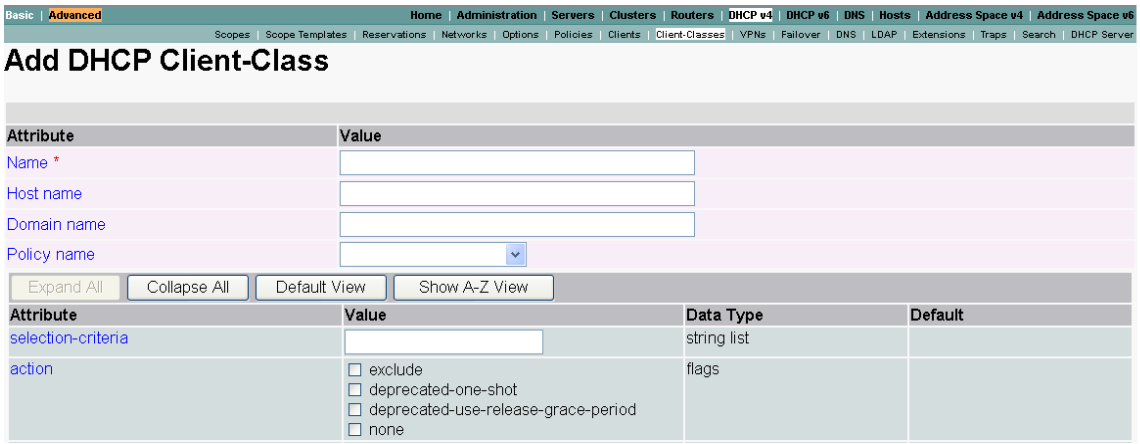
Step 2 In Basic mode, click the submenu **Classes of Service** to open the List DHCP Classes of Service (Client-Classes) page. In Advanced mode, click the submenu **Client-Classes** to open the List DHCP Client-Classes page.

Step 3 Create the client-class. Click **Add Client-Class** to open the Add DHCP Client-Class page (see Figure 24-1 on page 24-3 for a partial view of the Advanced mode page).

Step 4 Enter at least a distinguishing name for the client-class.

Step 5 Set other client-class properties. The hostname and domain name attributes are mainly used for DNS updates if not using a DNS update configuration (see the "Creating DNS Update Configurations" section on page 28-5). The hostname properties are described in the "Defining Client-Class Hostname Properties" section on page 24-4. You can also choose the appropriate policy for the client-class.

Figure 24-1 Add DHCP Client-Class Page (Local Advanced)



Step 6 Define the selection criteria. The critical step in creating a client-class is defining its selection criteria so that you can associate the client-class with a DHCPv4 scope or DHCPv6 prefix. Use the *selection-criteria* attribute. You can enter multiple selection tags by separating them with commas. The values have to match the selection tags set for the desired scope or prefix (see the "Setting Selection Tags on Scopes and Prefixes" section on page 24-4).

Step 7 To add an embedded policy to the client-class, see the "Editing Client-Classes and Their Embedded Policies" section on page 24-5.

Step 8 Click **Add Client-Class**.

Step 9 Debug as needed. To debug client-class errors, set the DHCP log settings to **client-criteria-processing**.

Step 10 To delete a client-class, click the Delete icon ( ) next to its name, and confirm the deletion.

CLI Commands

Enable client-classes by using **dhcp enable client-class**. To create the client-class, use **client-class** *name* **create**. The name should clearly identify its intent. It is not case sensitive; classPC is the same as Classpc.

Set properties of the clients in the client-class by using **client-class** *name* **set** *attribute*=*value*. For example, set the desired policy to associate with the client-class by using **client-class** *name* **set policy-name=***value*. Associate a scope with the client-class by using **client-class** *name* **set selection-criteria**. (See the "Setting Selection Tags on Scopes and Prefixes" section on page 24-4).

Show the properties of a created client-class by using **client-class** *name* [**show**]. You can also list the properties for all the client-classes created, or list just their names. To debug the client-class processing, use **dhcp set log-settings=client-criteria-processing**. To delete the client-class, use **client-class** *name* **delete**.

Setting Selection Tags on Scopes and Prefixes

To assign clients to different address pools, you must define the DHCPv4 scope (or template) or DHCPv6 prefix (or template) with the selection tags that you specified in the selection-criteria for the client-class. All the selection-criteria tags that the client-class has must match the tags the scope or prefix has, even though the scope or prefix might have additional tags. If the client-class omits all selection-criteria, no limitations apply to the scope or prefix selection.

For example:

Scope A has tag1, tag2
Scope B has tag3, tag4

Assuming both scopes are on the same network, a client in a client-class with:

- Tag1, tag2, or both, gets leases from scope A.
- Tag3, tag4, or both, gets leases from scope B.
- One or more tags from both scopes (such as tag1 and tag3) does not get leases from either scope.
- No tags gets leases from either scope.

Local Basic or Advanced Web UI

Create or edit a scope or prefix or its template; on the Add or Edit page for the scope or prefix (or its template), find the Selection Tags attribute and enter a list of comma-separated selection tags created in the *selection-criteria* attribute for the client-class that you want to associate with this scope or prefix (or its template). Then save the changes and reload the DHCP server, if necessary.

CLI Commands

Use **scope** *name* **set selection-tags**. For a scope template, use **scope-template** *name* **set selection-tag-list**. For a prefix, use **prefix** *name* **set selection-tags**. For a prefix template, use **prefix-template** *name* **set selection-tags**.

Defining Client-Class Hostname Properties

You can specify the hostname that each client should adopt, using the Hostname (*host-name*) attribute of the client-class. This can be an absolute, valid DNS value to override the one included in the DHCP client request, or can be any of these:

- ***@host-name-option***—The server uses whatever hostname option the client sent.
- ***@no-host-name-option***—The server ignores the hostname that the client sends. If DNS name generation is in effect, the server uses a generated name, if set up as such for dynamic DNS updating.
- ***@use-macaddress***—The server synthesizes a hostname from the client MAC address, hyphenates the octets, then adds an **x** at the front. For example, if a client MAC address is 1,6:00:d0:ba:d3:bd:3b, the synthesized hostname would be x1-6-00-d0-ba-d3-bd-3b.

If you omit a value, the hostname is unspecified. You can also synthesize hostnames by using a DNS update configuration (see the "Creating DNS Update Configurations" section on page 28-5).

See Also

Editing Client-Classes and Their Embedded Policies, page 24-5
Processing Client Data Including External Sources, page 24-6
Troubleshooting Client-Classes, page 24-8
Subscriber Limitation Using Option 82, page 24-14
Configuring Network Registrar to Use LDAP, page 24-19

Editing Client-Classes and Their Embedded Policies

Editing a client-class involves the same attributes as creating a client-class. You can also add and modify an embedded policy for the client-class so that you can set its policy options. The embedded policy has no properties or DHCP options associated with it until you add them. (See also the "Creating and Editing Embedded Policies" section on page 21-8). The client-class embedded policy setting is the third priority the DHCP server uses in its policy selection, after that set for the client itself (see the "Policy Hierarchy" section on page 21-3).

Local Basic or Advanced Web UI

Step 1 Create the client-class.

Step 2 Click the name of the client-class on the List DHCP Client-Classes page to open the Edit DHCP Client-Class page.

Step 3 Make changes to attribute settings as required.

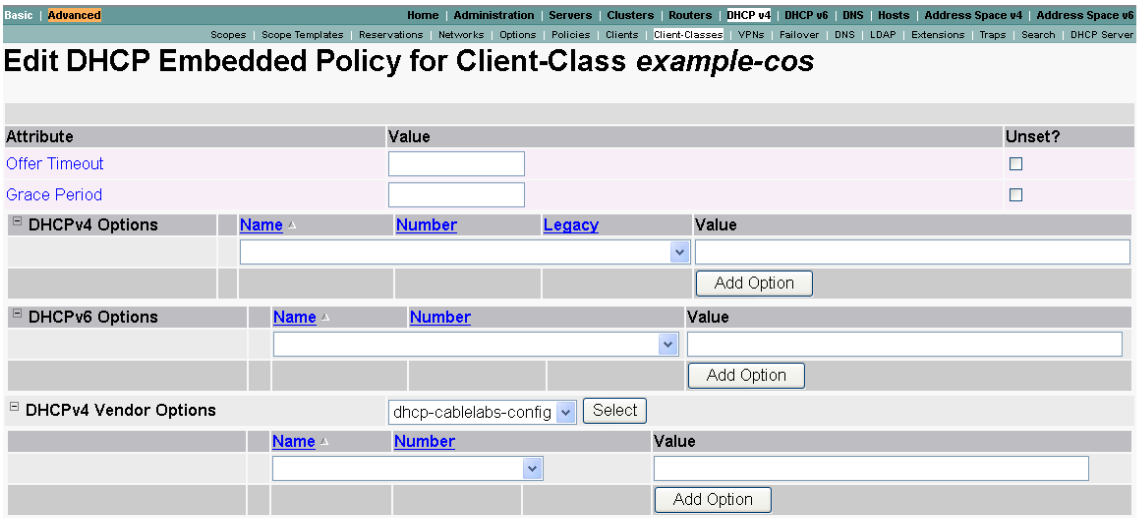
Step 4 To add a new embedded policy for the client-class, click **Create New Embedded Policy**. If there is an existing embedded policy that you want to edit, click **Edit Existing Embedded Policy**. Both actions open the Edit DHCP Embedded Policy for Client-Class page (see Figure 24-2 on page 24-6 for a partial view). (If you want to unset the existing embedded policy, click **Unset** on the Edit DHCP Client-Class page; this resets the button to **Create New Embedded Policy**.)

a. Modify the fields, options, and attributes on this page. For example, under the DHCPv4 Options, set the client lease time by choosing **dhcp-lease-time [51]** from the drop-down list, enter a lease interval value in the Value field, then click **Add Option**. If necessary, unset attribute values.

b. Click **Modify Embedded Policy** at the bottom of the page.

Step 5 Be sure to click **Modify Client-Class** at the bottom of the Edit DHCP Client-Class page.

Figure 24-2 Edit DHCP Embedded Policy for Client-Class Page (Local Advanced)



CLI Commands

To check if there are any embedded policy values already set for a client-class, use **client-class-policy** *client-class-name* **show**. To set the attributes for the embedded policy, use **client-class-policy** *client-class-name* **set** *attribute=value*. To set the DHCP options, use **client-class-policy** *client-class-name* **{setOption | setVendorOption | setV6Option | setV6VendorOption}** *option value*. To set the lease time, use **client-class-policy** *client-class-name* **setLeaseTime** *value*.

Processing Client Data Including External Sources

Information about network hosts running DHCP clients and their users can arrive at the DHCP server from several external sources. The server can use this data as part of client-class processing, and capture it in its lease database to make it available to the Network Registrar management system.

Recently introduced external factors that can influence client definitions are:

- A *subscriber-id* suboption of the *relay-agent-info* DHCP option (82), whereby a network administrator defines a network subscriber or client and sends this data to the DHCP server.
- RADIUS authentication server data, as part of 802.1x protocol deployments where the RADIUS data can be helpful in DHCP decision making. In this case, a device can send the data as part of *radius-attribute* suboption attributes in the *relay-agent-info* DHCP option (82).

Both these external options use DHCP option 82, as described in the "Subscriber Limitation Using Option 82" section on page 24-14. The RADIUS source can send the following attributes:

- Client user or account name (the user attribute)
- Administratively defined class string (the class attribute)
- Vendor-specific data (the vendor-specific attribute)
- Session timeout value (the session-timeout attribute)
- IP address pool to use for the client (the framed-pool attribute)
- IPv6 address pool to use for the client (the framed-ipv6-pool attribute)

Network Registrar provides extension support for the *subscriber-id* suboption and the user, class, and framed-pool attributes of the RADIUS suboption (see the Appendix C, "DHCP Extension Dictionary"), and expression support for all of the suboptions (see Chapter 25, "Using Expressions"). Additionally, the DHCP server now includes attribute settings to configure how the server handles the RADIUS class and framed-pool attributes. Network Registrar can use the server attributes to map the RADIUS attribute value as a selection tag or client-class name, or append the value to the selection tag that it finds in its client database. For example:

```
nrcmd> dhcp set map-radius-class=append-to-tags
```

For client-classes and scope-selection tags determined from external resources such as RADIUS, the processing order is slightly more complex than that described in the "Client-Class Process" section on page 24-2. See the following subsections. Remember that to use the client-class feature, you must enable the DHCP server *client-class* attribute.

Processing Order to Determine Client-Classes

The order in which the DHCP server uses possible sources to determine client-class names is as follows:

1. It uses the client-class name in the extension environment dictionary.
2. If it finds a real client-entry in the database, it uses its *client-class-name*. (To prevent this unnecessary database read, enable the *skip-client-lookup* DHCP server attribute; see the "Skipping Client Entries for Client-Classing" section on page 24-13.)
3. If you map the RADIUS framed-pool value to a client-class (by using **dhcp set map-radius-pool-name=map-as-class**), it uses the framed-pool value.
4. If you map the RADIUS class value to a client-class (by using **dhcp set map-radius-class=map-as-class**), it uses the class value.
5. If you map the *dhcp-user-class-id* DHCP option (77) to a client-class (by using **dhcp set map-user-class-id=map-as-class**), it uses the option value. (Note that you can alternatively use a lookup ID expression instead of this mapping; see the "Client-Class Lookup Expression Processing" section on page 24-16.)
6. If it finds no mapping or user-class ID, it uses the default-client-class-name from the environment dictionary.
7. If it finds no default-client-class-name or client-entry, it uses the client-class-name from the client named **default** (if found).

Processing Order to Determine Selection Tags

The order in which the server uses the possible sources to determine selection tags (it uses the first nonnull source) is as follows:

1. Selection tags in the extension environment dictionary.
2. If it finds a real client-entry in the database, it uses the client-entry *selection-tags*. (To prevent this unnecessary database read, enable the *skip-client-lookup* DHCP server attribute; see the "Skipping Client Entries for Client-Classing" section on page 24-13.)
3. Scope-selection tags in the client-class.
4. If you map an available RADIUS framed-pool value to a tag (by using **dhcp set map-radius-pool-name=map-as-tag**), it uses that tag.
5. If you map an available RADIUS class value to a tag (by using **dhcp set map-radius-class=map-as-tag**), it uses that tag.

6. If you map an available *dhcp-user-class-id* DHCP option (77) to a tag (by using **dhcp set map-user-class-id=map-as-tag**), it uses that tag.

Next, the server could append one of the following to the list of selection tags (if any):

1. If a RADIUS framed-pool value is available and you set the *map-radius-pool* DHCP attribute to append to the tags (by using **dhcp set map-radius-pool=append-to-tags**), the server appends it.
2. If a RADIUS class value is available and you set the *map-radius-class* DHCP attribute to append to the selection tags (by using **dhcp set map-radius-class=append-to-tags**), the server appends it.
3. If a dhcp-user-class-id is available and you set the *map-user-class-id* DHCP attribute to append to the selection tags (by using **dhcp set map-user-class-id=append-to-tags**), the server appends it.

Troubleshooting Client-Classes

To troubleshoot a client-class, enable client-class logging using the *log-settings* attribute on the Edit DHCP Server page of the web UI, or **dhcp set log-settings=***setting* in the CLI, then reload the DHCP server (if in staged scope edit mode). The recommended settings are:

- **client-detail**—Logs a single line at the end of every client-class client lookup operation. This line shows all the data found for the client as well as the data that was found in the client-class.
- **client-criteria-processing**—Logs a message whenever the server examines a scope or prefix to find an available lease or to determine if a lease is still acceptable for a client that already has one.
- **ldap-query-detail**—Logs messages whenever the DHCP server initiates a lease state entry creation to an LDAP server, receives a response from an LDAP server, or retrieves a result or error message from an LDAP server.
- If the problem could be related to your LDAP server, also enable the LDAP *can-query* setting.

These logs will help answer these questions:

- Is the server reading the client entry from the expected database?

  The server can read the client entry from LDAP or MCD (the Network Registrar internal database). The *client-detail* log shows you where the server is reading the client entry.

- Is client-class enabled?

  If enabled, but you get unexpected results, verify the database that your Network Registrar server is reading. Is it reading from LDAP or MCD? The *ldap-query-detail* log tells you if it is reading from LDAP. If not, enable the DHCP *use-ldap-client-data* property.

  Note Using LDAP requires configuring the LDAP server for queries. Enable the LDAP *can-query* attribute. You also must configure the DHCP server to use LDAP for queries.

- Is the server providing clients the right data, but you see the wrong results from that data (for example, clients are not receiving the expected IP addresses)?

  Verify the explicit relationships on your network. The *client-criteria-processing* log shows from which scopes or prefixes the server is getting addresses. If it does not get addresses from the expected sources, explicit relationships might be incorrectly defined. A scope that you thought was a secondary scope might not be defined that way.

- In Expert mode, did you set the include and exclude criteria for selection tags properly?

  If you define a series of selection tags to include, the tags of a scope or prefix must match those of the client. In Expert mode, you can also use a *selection-criteria-excluded* attribute on the client-class to exclude selection tags. If you define a series to exclude, a scope or prefix must have none of these tags defined so that the client can get configuration parameters from it. Avoid complex inclusion and exclusion scenarios as you begin working with selection tags.

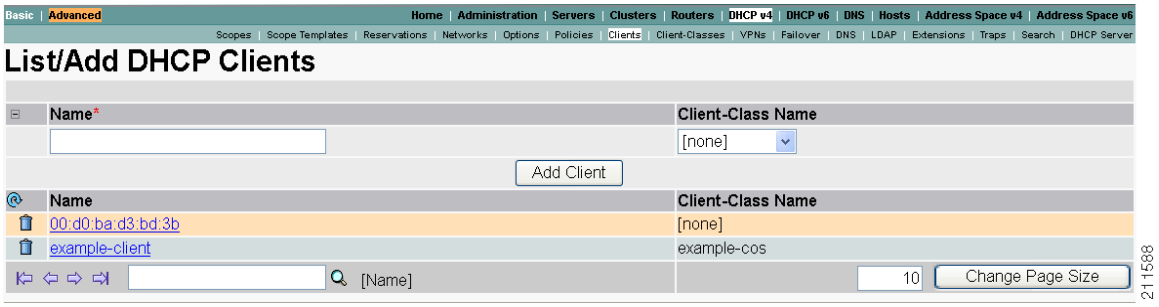
Configuring Clients

DHCP client properties include the participating client-class and associated policy for a client, the action to perform, and the inclusion and exclusion criteria for scope-selection tags. A client inherits the properties from its client-class, which you may choose to override or supplement by specifying different properties for the client.

Local Basic or Advanced Web UI

Step 1 Click **DHCP**, then **Clients** to open the List/Add DHCP Clients page (see Figure 24-3).

Figure 24-3 List/Add DHCP Clients Page (Local Advanced)



Step 2 Enter the client identity, typically a MAC address, but it can also be a DUID or lookup key. (Note that you can set up the DHCP server to validate the client name as a MAC address by enabling the server attribute *validate-client-name-as-mac*.)

You can also create a client named **default** that does not have a specific client configuration. For example, you can have a client always use its MAC address for its hostname.

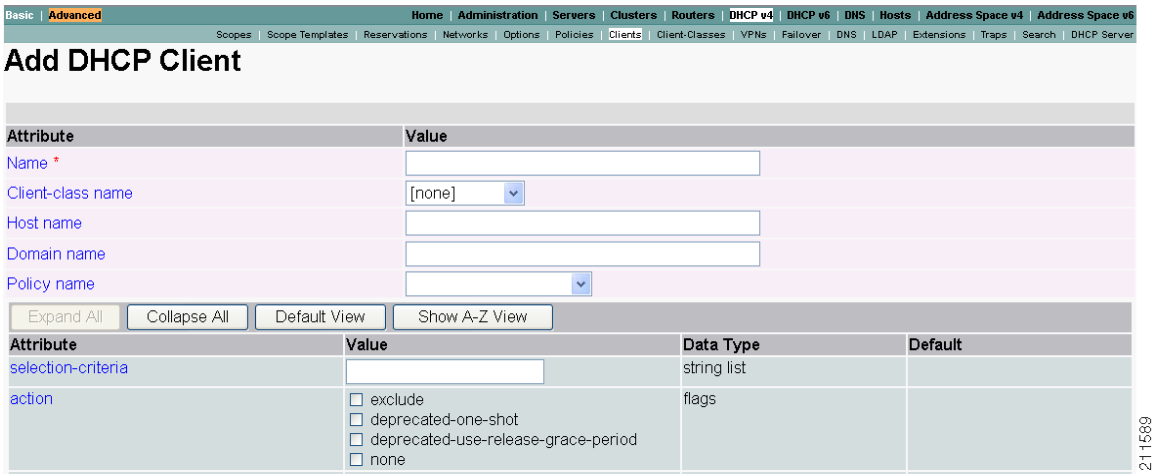
Step 3 Click a client-class name, if desired, from the drop-down list of predefined client-classes.

Step 4 Click **Add Client**. If you did not choose a client-class, this opens the Add DHCP Client page (see Figure 24-4 on page 24-10 for a partial view).

Step 5 The critical step in creating a client is defining its selection criteria so that you can associate the client with a scope or prefix (unless the selection criteria were already set up for the client-class associated with the client). Use the *selection-criteria* attribute under the Attribute list. You can enter multiple selection tags by separating them with commas. The values have to match the selection tags set for the desired scope or prefix (see the "Setting Selection Tags on Scopes and Prefixes" section on page 24-4).

Note If you chose a client-class for the client, this page does not appear, and the client name is listed on the List/Add Client page.

Figure 24-4 Add DHCP Client Page (Local Advanced)



Step 6 Set other attributes as desired. For example:

- Set the *host-name* attribute to *@no-host-name-option* to provide provisional addresses to unknown clients. See the "Allocating Provisional Addresses" section on page 24-12.
- Set the domain name of the zone to use when performing dynamic DNS updates.
- Set the policy and action for the client. With the *exclude* action, the server ignores all communication from this client (no packets are shown); with the *one-shot* action, the server does not renew or re-offer a lease to this client.
- Choose the number of time units (seconds, minutes, hours, days, weeks), or UNIX style date (such as Mar 24 12:00:00 2002) to indicate when the authentication expires, or use **forever**.

Step 7 Click **Add Client** at the bottom of the page.

Step 8 Debug as needed. To debug client errors, set the DHCP log settings to **client-criteria-processing**.

Step 9 To delete a client, click the Delete icon ( ) next to its name, and confirm the deletion.

CLI Commands

To create a client, use **client** *name* **create**. To associate a client-class with the client, use **client** *name* **set client-class-name**=*value*. To set selection criteria for scopes or prefixes, use **client** *name* **set selection-criteria**. To set other attributes, use **client** *name* **set** *attribute*=*value*.

To display client properties, use **client** *name* [**show**]. To display properties for all the clients, use **client list**, or **client listnames** to list just the names. To debug clients, use **dhcp set log-settings=client-detail**. To delete a client, use **client** *name* **delete**.

See Also

Editing Clients and Their Embedded Policies, page 24-11
Setting Windows Client Properties, page 24-11
Allocating Provisional Addresses, page 24-12
Skipping Client Entries for Client-Classing, page 24-13
Limiting Client Authentication, page 24-13
Setting Client Caching Parameters, page 24-14

Editing Clients and Their Embedded Policies

Editing a client involves the same attributes as creating a client. You can also add and modify an embedded policy for the client so that you can set its policy options. The embedded policy has no properties or DHCP options associated with it until you add them. (See also the "Creating and Editing Embedded Policies" section on page 21-8.) The client embedded policy setting is the first priority the DHCP server uses in its policy selection (see the "Policy Hierarchy" section on page 21-3).

Local Basic or Advanced Web UI

Step 1 Create the client.

Step 2 Click the name of the client on the List DHCP Clients page to open the Edit DHCP Client page.

Step 3 Make changes to attribute settings as required.

Step 4 To add a new embedded policy for the client-class, click **Create New Embedded Policy**. If there is an existing embedded policy that you want to edit, click **Edit Existing Embedded Policy**. Both actions open the Edit DHCP Embedded Policy for Client page. (This page is almost identical to the Edit DHCP Embedded Policy for Client-Class page; see Figure 24-2 on page 24-6 for a partial view.)

a. Modify the fields, options, and attributes on the Edit DHCP Embedded Policy for Client page. For example, under the DHCPv4 Options, set the client lease time by choosing **dhcp-lease-time [51]** from the drop-down list, enter a lease interval value in the Value field, then click **Add Option**. If necessary, unset attribute values.

b. Click **Modify Embedded Policy** at the bottom of the page.

If you want to unset the existing embedded policy, click **Unset** on the Edit DHCP Client page; this resets the button to **Create New Embedded Policy**.

Step 5 Be sure to click **Modify Client** at the bottom of the Edit DHCP Client page.

CLI Commands

To see if there are any embedded policy values already set for a client, use **client-policy** *client-name* **show**. To create an embedded policy, use **client-policy** *client-name* **set** *attribute=value*. To set the DHCP options, use **client-policy** *client-name* **{setOption | setVendorOption | setV6Option | setV6VendorOption}** *option value*. To set the lease time, use **client-class-policy** *client-name* **setLeaseTime** *value*.

Setting Windows Client Properties

Windows 2003 and XP clients support class-based provisioning. You can set certain properties that relate to client-class processing. These are:

- Look up the client entry to determine the default client for client-class processing.
- Map the user class ID to the client-class or selection tag.
- Set whether to append the class ID to the selection tag name.

Settings in Windows Clients

On the Windows client host, use **ipconfig /setclassid** to set the class ID. If you plan to map this client ID to a client-class or selection tag, it must have the same name. Then confirm by using **ipconfig /showclassid**. For example:

```
DOS> ipconfig /setclassid adapter engineering
DOS> ipconfig /showclassid adapter
```

Settings in DHCP Servers

You must set Windows client properties in the DHCP server.

Use DHCP server attributes in the local cluster web UI or **dhcp set** command attributes in the CLI to set the Windows client properties for the server. If you set the *skip-client-lookup* attribute to true (the default is false), the DHCP server skips the client entry for client-class processing. (See the "Skipping Client Entries for Client-Classing" section on page 24-13.) Use one of the *map-user-class-id* attribute settings:

- **0**—Ignore the user class ID (the default)
- **1**—Map the user class ID to the scope-selection tag
- **2**—Map the user class ID to the client-class
- **3**—Append the user class ID to the list of scope-selection tags

Allocating Provisional Addresses

You can provide provisional addresses to clients.

Provisional Addresses for Unknown Clients

The DHCP server can allocate provisional addresses to unknown clients for a short time on a one-shot basis. The server gives an address to the unknown client only as long as its lease period (which should be set short), and the client cannot renew the lease. Once the lease expires, the client cannot obtain a new lease until after the grace period expires (this locks the client out of network access). The idea is to give the client a short time to register and prevent it from using the network if it does not register in that time.

Step 1 Create an **unknown** policy, for example (the name is arbitrary).

Step 2 Use the *Grace period* field on the Edit DHCP Policy page, or the **policy unknown create grace-period=***extended-time* setting in the CLI.

Step 3 Use the **default** client to set the *Policy name* value to **unknown**, and the *action* attribute value to **one-shot** on the Edit DHCP Client page, or use **client default create policy-name=unknown action=one-shot** in the CLI, to give provisional addresses to unknown clients.

Note Provisioning unknown clients is not supported in DHCPv6.

Using One-Shot Action

Use the one-shot action to allocate provisional addresses. This is useful when you want a client to have an address for only a short time. Configure the default client (or the client-class that the default client specifies) by setting the *action* attribute to **one-shot**.

The server then gives a lease to an unknown client, but does not allow it to renew the lease. When the lease expires, the server does not respond to that client during the lease grace period, and only responds when a different client uses the lease. The grace period, therefore, is the minimum period during which the client cannot obtain a lease.

You can allow the client a relatively short lease time, such as one day, and specify a long grace period, such as two weeks. This way you can offer an incentive to the client to register with some authority and become a known client, while not re-allocating the lease to another client. After the lease expires, the client cannot get another address for the lease for the two-week grace period.

You can configure the lease and grace period differently for each scope or prefix, so that provisional leases can have different lease and grace periods than nonprovisional ones. Provisional addresses are less restrictive if you use multiple DHCP servers, because each server operates its one-shot capabilities independently. With the approach described and two DHCP servers, an unregistered client can get two days of provisional address use every two weeks.

Skipping Client Entries for Client-Classing

You may not want to honor client entries for client-classing to prevent unnecessary database reads. To accomplish this, enable the *skip-client-lookup* DHCP server attribute (**dhcp enable skip-client-lookup** in the CLI).

Limiting Client Authentication

By default, client entries get unlimited authentication. Using the *authenticate-until* attribute, you can limit authenticating a client entry by specifying an expiration time.

When a client entry is no longer authenticated, the DHCP server uses the *unauthenticated-client-class-name* attribute value for the name of the client-class entry to use in answering this DHCP request. If this attribute is unset, or if there is no client-class entry in it, the DHCP server ignores the request.

The valid client authentication values are:

- ***+num unit***—Time in the future, where *num* is a decimal number and *unit* is *s*, *m*, *h*, *d*, or *w* for seconds, minutes, hours, days or weeks, respectively. For example, “+3w” is three weeks in the future.
- ***date***—Month, day, 24-hour, and 2-or-4-digit-year. For example, “Jun 30 20:00:00 2002.” Enter the local process time. If the server runs in another time zone, disregard the time zone and use local time instead.
- **forever**—Does not expire the authentication for this client.

An example follows of using the *authenticate-until* attribute to distinguish between clients that are authenticated and those that are not authenticated. After the authentication expires and the client requests another address, the DHCP server assigns the client an address from the unauthenticated scope range:

Step 1 Create an authenticated and an unauthenticated client-class. Set the selection criteria for each as appropriate.

Step 2 Create the client and include the authenticate-until expiration time. Set the *client-class-name* and *unauthenticated-client-class-name* attributes as appropriate.

Step 3 Create the authenticated and unauthenticated scopes, define their address ranges, and tie them to their respective selection tags.

Step 4 Enable client-class processing for the server.

Step 5 If necessary, reload the DHCP server.

Setting Client Caching Parameters

The initial request from a client for an address from a DHCP server often goes through a DHCPDISCOVER-DHCPOFFER-DHCPREQUEST-DHCPACK cycle. This process requires that the server must consult the database twice per request for client data. If the client caching parameters are set, the DHCP server caches client data in memory so that it only needs to consult the database once. Client caching can provide a noticeable performance improvement in systems that store client information in LDAP. Client caching is enabled by default unless you unset the applicable attributes.

You can adjust the maximum cache count and time-to-live (TTL) parameters based on the expected rate of client requests. If you expect an onslaught of requests, you might want to increase the cache count, up to a limit based on your available memory. If you expect a longer request cycle, you might want to increase the TTL. The aim is to have the server consult the client cache once during the request cycle.

To set the limit on the number of entries that the server keeps in the client cache, use the *client-cache-count* attribute on the Edit DHCP Server page, or **dhcp set client-cache-count** in the CLI. By default, the maximum number to cache is 1000 clients. To disable the cache, set the attribute to 0.

The client cache is usually valid for only ten seconds, called the cache TTL. After the TTL expires, the server reads the client information from the database, if necessary. You can adjust the TTL using the *client-cache-ttl* attribute on the Edit DHCP Server page, or **dhcp set client-cache-ttl** in the CLI.

When the client cache count reaches the specified maximum, the server cannot cache any more clients until a client-entry TTL expires, after which it reads from the database and begins caching again.

Subscriber Limitation Using Option 82

In many situations, service providers want to limit the number of IP addresses the DHCP server should give out to devices on customer premises. They want these devices to have “real” addresses that the DHCP server provides, but limit their number. One way is to use the client-class to register (or provision) each customer device so that the server issues IP addresses only to devices that are registered in the client-entry database. The major drawback to this approach is that it requires registering every customer device, which involves knowing its MAC address. Service providers often do not want to know about each device, but simply that there are not too many of them per customer.

Another approach is to limit customer devices on a per-subscriber basis on values in the *relay-agent-info* DHCP option (option 82, as described in RFC 3046) that the DHCP relay agent sends in a DHCPDISCOVER message. This option includes data about the port on a switch over which the customer device is attached. In a cable modem scenario, one of the option 82 suboptions usually contains the MAC address of the cable modem when the DHCP request comes from a device attached beyond the cable modem. In general, many devices that generate option 82 data place some values in its suboptions such that the value varies per subscriber on the same upstream device. In some cases, this value is unique across all possible subscribers (such as the MAC address of the cable modem). In others, it can be a port on a switch and thus unique across the other subscribers attached to that switch. However, it might not be unique across all subscribers on the switch.

Using this approach, the network administrator can configure limitations on subscriber use of the DHCP-allocated addresses without seriously impacting other DHCP server capabilities. In many environments, network administrators might want to use option 82 limitation for some class of devices and not others. A key aspect of this support is to allow network administrators to separate the devices for which they want to use option 82 limitation from those for which they do not.

See Also

General Approach to Subscriber Limitation
Typical Limitation Scenario, page 24-16
Calculating Client-Classes and Creating Keys, page 24-16
Client-Class Lookup Expression Processing, page 24-16
Limitation Processing, page 24-16
Expression Processing for Subscriber Limitation, page 24-17
Configuring Option 82 Limitation, page 24-17
Lease Renewal Processing for Option 82 Limitation, page 24-18
Administering Option 82 Limitation, page 24-18
Troubleshooting Option 82 Limitation, page 24-19
Expression Examples, page 24-19

General Approach to Subscriber Limitation

The current approach to client processing is to look up every client in the client-entry database. One of the goals of option 82 limitation is to remove the need explicitly to register (provision) every customer device in the client-entry database (either in the MCD database or LDAP). However, there is still a requirement that the specific number to which a subscriber is limited should be configurable and override the default number given to all unregistered subscribers.

Note Limitation processing is not currently available for DHCPv6 clients.

At a high level, you can configure subscriber limitation by creating an *expression* that the server evaluates for each incoming packet and returns the name of the client-class where you want the client to go. (See Chapter 25, "Using Expressions," for details on the use of expressions.) Each client-class allows specification of a limitation identifier (ID), a key the server determines from the incoming packet and uses in later processing to actually limit the number of devices. The server considers all devices with the same limitation ID (the *limitation-id* property) to come from the same subscriber.

Typical Limitation Scenario

For example, an incoming packet might be evaluated such that:

1. If the *remote-id* suboption of option 82 matches the client hardware address (*chaddr*), the subscriber is a cable modem and should be in the **cm-client-class**.
2. If the first six bytes in the *dhcp-class-identifier* option value match the string **docsis**, then the subscriber is a DOCSIS modem and should be in the **docsis-cm-client-class**.
3. If the *user-class* option value matches the string **alternative-class**, then the subscriber should be in the **alternative-cm-client-class**.

Calculating Client-Classes and Creating Keys

You set the expression that determines the client-class for the *client-class-lookup-id* attribute of the DHCP server, or **dhcp set client-class-lookup-id=***expression* in the CLI. Include simple expressions in the attribute definition or more complex ones in a file referenced in the attribute definition (see the "Using Expressions" section on page 25-1).

Clients and client-classes also allow specifying a *limitation-id* value for the client or client-class. The server uses this identifier (ID) value to set the address limit on the number of devices with the identical ID on the same network or LAN segment. If a requesting client oversteps the limit of available addresses for that ID, the server assigns it to an *over-limit-client-class-name* (if set); otherwise, it drops the packet. The *limitation-id*, in effect, defines a subscriber.

Client-Class Lookup Expression Processing

The initial client-class lookup is to allow you to decide whether the client should participate in some sort of limitation. Configure an expression server-wide with the *client-class-lookup-id* attribute. The server executes this expression on every incoming packet with the goal of determining the client-class of the packet.

The expression should return a string that is the client-class name for the packet, or the distinguishing string <none> indicating that no client-class value was considered for the client request. Returning the <none> string is equivalent to not configuring a *client-class-lookup-id* value and that no client-class processing should occur. If the expression returns null or there is an error evaluating the *client-class-lookup-id*, the server drops the packet (with an accompanying log message).

Limitation Processing

The DHCP server limits the number of IP addresses allocated to DHCP clients with the same *limitation-id* value in the same network or LAN segment. In cases where the server finds that allocating another address to the client would go over the limit, it places the client packet in the *overflow-client-class* (if any is specified). This allows special handling for clients that are over the configured limit. Handling these clients in some self-provisioning way is one of the benefits of using limitation on the DHCP server instead of in the hardware (should it even be supported).

If there is no over-limit client-class, the server drops a packet where allocating an address for that packet would exceed the allowed *limitation-count* for that *limitation-id*. Note that the server enforces the limitation only in a single network or LAN segment. This is hardly a restriction, because network managers tend to see a single subscriber connecting only over one LAN segment at a time.

Configure the *limitation-count* with an identical *limitation-id* in a DHCP policy. The limitation code searches up the policy hierarchy for the *limitation-count* just as it does for any other policy item. This means that you can configure the *limitation-count* in a client-class embedded or named policy, a scope embedded or named policy, or the system *system_default_policy*.

When you configure a *limitation-id* on a client-class, you thereby signal to pursue limitation processing for the client-class. When you do not configure a *limitation-id*, you thereby signal not to pursue it. When executing the expression to determine the *limitation-id*, if the expression returns null, this signals that limitation processing should occur and to use the *limitation-id* saved in the lease state database.

Expression Processing for Subscriber Limitation

Expressions exist in several places in the limitation processing. Each expression evaluates to null or a string (typically to determine a client-class name when looking up a client-class), or to a series of bytes (a blob) when creating a *limitation-id*. You can use expressions in these places:

- Looking up a client-class
- Creating the key to limit clients of the same subscriber (the *limitation-id*)
- Creating the key to look up in the client-entry database (the *client-lookup-id*)

Configuring Option 82 Limitation

Step 1 If you do not register clients explicitly, do not enable client-class as a DHCP server property when using option 82 data.

Step 2 Determine if you want to limit some clients and not others. If you want to limit some clients:

a. Find some method to distinguish these clients from the others, based on some values contained in the DHCP requests from each class of clients.

b. Determine the names of the client-classes into which you want to put the clients that are not limited, and the selection tag and scope or scopes you want to use for these unlimited clients.

Step 3 Decide if you want to put clients that are over-limit into a different client-class or just drop their packets. If you want to put them in an over-limit client-class, determine the client-class name and the selection tag and scope or scopes into which you want to put the over-limit clients.

Step 4 Determine the client-class into which you want to put clients that you intend to limit and the selection tags and scope or scopes you want to use for these clients.

Step 5 Create all these selection tags, client-classes, and scopes.

Step 6 Configure the *limitation-count* in a policy, probably the named policy associated with the client-class for the clients to limit.

Step 7 Write the expression to separate the incoming clients into those to be limited and those not to be limited. Configure it on the DHCP server by setting the *client-class-lookup-id* attribute.

Step 8 Write the expression to determine the limitation ID for the devices to limit, and configure it on the client-class for clients to limit by setting the *limitation-id*.

Lease Renewal Processing for Option 82 Limitation

Only packets that the DHCP client broadcasts arrive at the server with option 82 data attached. The BOOTP or DHCP relay agent adds the option 82 data in the first upstream router from the client device. A DHCPRENEW packet is unicast to the server and arrives without option 82 data. This can pose a problem when trying to configure the server for subscriber limitation.

There are generally two approaches to take when dealing with renewals:

- Place all packets that do not have option 82 data in a client-class with no associated selection tags. This is equivalent to a wildcard selection and means that any packet with no option 82 data is accepted.
- Place a DHCPRENEW in the same client-class as you would place a packet that has option 82 data, and have its *limitation-id* evaluate to null. This is a signal that when checking for limitation, the DHCP server should use a previously stored *limitation-id* instead of one from the packet.

Both approaches work. The second one can be more secure, but in practice, it is not much better than the first. This is because you have to use an IP address for the DHCP server to respond to a DHCPRENEW, and most clients would not do this unless the server loses some of its state. In this case, you would want it to give the client the address. In the case of a malicious client, it would still have to use the address to get the server to give the address to the client, thereby limiting the exposure for this case.

Administering Option 82 Limitation

Whenever a client is involved in limitation because of its inclusion in a client-class with a *limitation-id*, the limitation ID used appears in the DHCP log file whenever client data logging occurs as "... LID: *nnn*:*nnn*:*nnn*...." The data is logged only for clients with active leases that are currently occupying one of the *limitation-count* counts.

You can determine all the clients using a *limitation-id* in a subnet. On the Manage DHCP Server page, click the Run icon ( ) in the Commands column to open the DHCP Server Commands page. Enter at least the IP address of the currently active lease in the IP Address field, then click the Run icon. You can also enter the *limitation-id* itself in the form *nn***:***nn***:***nn* or as a string (**"***nnnn***"**), in which case the IP address becomes the network in which to search. In the CLI, use **dhcp limitationList**:

```
nrcmd> dhcp limitationList ipaddr [limitation-id] show
```

If you specify both the *ipaddr* and *limitation-id*, the *ipaddr* value, the server uses it just like a *giaddr* to determine the subnet. You can use any IP address that could appear in any scope (primary or secondary) for the network to specify a subnet. If you specify only the *ipaddr*, it must be an address that the DHCP server serves, and the command returns all the clients and corresponding leases they use.

If a client is denied service due to a *limitation-count* overflow, a message such as this appears in the DHCP server log file:

```
Warning Server 0 05646 Could not add Client MAC: '1,6,01:02:03:04:0c:03'  with
limitation-id: 01:02:03 using Lease: 10.0.0.23, already 3 Clients with that id.
No over-limit client class specified! Dropping packet!
```

You can determine which clients are extended beyond the *limitation-count*, thus causing a denial of service for any new client, by using **dhcp limitationList**. The *ipaddr* value in the command should be the "using Lease:" value, and the limitation-id should be the "limitation-id:" value, in the log file. Using the log file example, the command would be:

```
nrcmd> dhcp limitationList 10.0.0.23 01:02:03 show
```

Troubleshooting Option 82 Limitation

There are several ways that you can debug limitation support. First, you might want to turn on packet tracing by setting the DHCP server debug value to **VX=1** (or using **dhcp setDebug VX=1**). (The **VX=0** debug value disables packet tracing.) Then, you probably want to enable client-class debugging by adding *client-criteria-processing* and *client-detail* to your log settings.

There is also a server-wide expression trace level, *expression-trace-level*, that you can set to various levels. Setting it to 6 gives you a details trace of every expression evaluation. This can take a bit of space in the log, and slows down the server considerably as well, but is invaluable in the process of getting familiar with expression evaluation. See the "Debugging Expressions" section on page 25-24.

When things seem to be going strangely, or when submitting log files to report a problem, it is important to enable some additional tracing by setting the DHCP server debug value to **QR57=9** (or using **dhcp setDebug QR57=9**. (The **QR57=0** debug value disables this tracing). Note that the Q and R are both uppercase. The Q is client-class debugging and the R is response debugging (required to get the flow of control clear in the log). The 5 is expression processing and the 7 is *client-class-lookup* processing. This generates a page or so of output for each packet, which will help you understand what is going on inside the server.

Expression Examples

See the "Expression Examples" section on page 25-21.

Configuring Network Registrar to Use LDAP

The Lightweight Directory Access Protocol (LDAP) provides directory services to integrate Network Registrar client and lease information. By building on your existing standard schema for objects stored in LDAP directories, you can handle information about DHCP client entries. Thus, instead of maintaining client information in the DHCP server database, you can ask the Network Registrar DHCP server to issue queries to one or more LDAP servers for data in response to DHCP client requests, or write lease data to an LDAP server.

Network Registrar on Windows and Solaris uses the iPlanet LDAP Software Development Kit (SDK) version 5.0 (previous releases used SDK version 3.0). Linux uses their OpenLDAP client distribution.

See Also

About LDAP Directory Servers
Adding and Editing LDAP Remote Servers
Configuring DHCP Client Queries in LDAP, page 24-21
Configuring DHCP LDAP Update and Create Services, page 24-24
Troubleshooting LDAP, page 24-30

About LDAP Directory Servers

LDAP directory servers provide a way to name, manage, and access collections of attribute/value pairs. You can enter information into your LDAP server in any number of ways, because Network Registrar does not depend on specific LDAP object classes or schema:

- You can store DHCP client information in unused attributes. For example, you could use the *givenname* attribute to hold the DHCP *client-class name* value.
- You can add new attributes to an object class without altering your LDAP schema if you disable LDAP schema checking. For example, you could add the *client-class-name* attribute to the organizational person object class.
- You can create a new object class and define the appropriate attributes. For example, you can create the DHCP client object class and define the client attributes that you want to use.

When you configure the DHCP server to read from LDAP, a query dictionary tells the server which LDAP attributes to query for. The server converts the resulting data into DHCP client data attributes.

Tip You can configure Network Registrar to generate SNMP traps when an LDAP server stops responding or resumes responding to requests from the DHCP server.

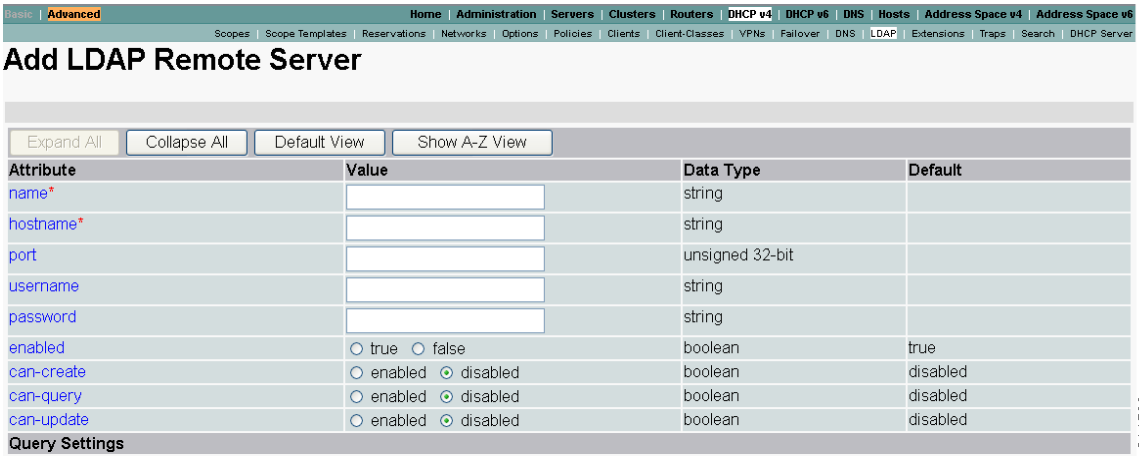
Adding and Editing LDAP Remote Servers

You must add a remote LDAP server so that you can begin using the LDAP services.

Local Advanced Web UI

Click **DHCP**, then **LDAP** to open the List LDAP Remote Servers page. Click **Add LDAP Remote Server** to open the Add LDAP Remote Server page (Figure 24-5 for a partial view). To edit the remote server, click its name to open the Edit LDAP Remote Server page.

Figure 24-5 Add LDAP Remote Server Page (Local Advanced)



On this page, provide at least the name and fully qualified domain name of the LDAP server. The username and password are required for successful operation.

CLI Commands

Use **ldap** *name* **create** *domain-name*. For example:

```
nrcmd> ldap ldap-1 create ldap.example.com
```

Configuring DHCP Client Queries in LDAP

You can configure and unprovision DHCP client queries, and configure embedded policies, in an LDAP client entry.

Configuring DHCP-Server-to-LDAP Client Queries

To enable the DHCP server to query your LDAP server for client data, perform the following steps. Like local client entries, LDAP client entries are keyed by the client MAC address.

Note When connecting to an LDAP server, use the *distinguished name* (*dn*) of the user. It uniquely identifies an object in the LDAP schema, and is like a unique key in a database or a fully qualified path name for a file. For example, a dn for a person might be dn: cn=Beth Jones, ou=Marketing, o=Example Corporation. In this company, there may be many people named Beth and many people named Jones, but no one else named Beth Jones works in Marketing at Example Corporation.

Step 1 Supply a hostname for the LDAP server. On the Add LDAP Remote Server page (see Figure 24-5 on page 24-20), enter a value in the name field. In the local CLI, use this command:

```
nrcmd> ldap ldap-1 create ldap.example.com
```

Later, if you need to delete the server, use **ldap** *server* **delete**.

Step 2 Configure the connection credentials. Use the distinguished name (*dn*) for the user. Enter a value in the username field. In the CLI, use this command, for example:

```
nrcmd> ldap ldap=1 set username="cn=joe,o=Example Corp,c=US" password=access
```

Step 3 Set the search path (and, if necessary, the search scope). The path is a point in the directory from which to start searches. If the search scope is:

- SUBTREE, the server searches all the children of the search path.
- ONELEVEL, the server searches only the immediate children of the base object.
- BASE, the server searches only the base object itself.

This example sets the base of the search to be the organization Example Corp and the country US, with a subtree search scope. Enter a value in the search-path field. In the CLI, use this command, for example:

```
nrcmd> ldap ldap-1 set search-path="o=Example Corp,c=US" search-scope=SUBTREE
```

Step 4 Set the search filter to be the attribute for which DHCP will substitute the clients' MAC addresses. In this example, the attribute is the common name (*cn*). Enter a value in the search-filter field. In the CLI, use this command, for example:

```
nrcmd> ldap ldap-1 set search-filter=(cn=%s)
```

Step 5 Configure a query dictionary that contains all the LDAP-to-DHCP mappings. Use **ldap** *servername* **setEntry** to set these mappings.

a. Retrieve the DHCP surname from the *sn* LDAP attribute:

```
nrcmd> ldap ldap-1 setEntry query-dictionary sn=host-name
```

b. Retrieve the client-class name from the first *givenname* LDAP attribute:

```
nrcmd> ldap ldap-1 setEntry query-dictionary givenname=client-class-name
```

c. Retrieve the domain name from the *localityname* LDAP attribute:

```
nrcmd> ldap ldap-1 setEntry query-dictionary localityname=domain-name
```

d. If you need to unset any of the entries, use **ldap** *server* **unsetEntry** *attribute key.* You can also check any of the settings using **ldap** *server* **getEntry** *attribute key.*

Step 6 Enable queries for the LDAP server. This example enables queries for *myserver.* Set the *can-query* attribute to enabled. In the CLI, use this command:

```
nrcmd> ldap ldap-1 enable can-query
```

Step 7 Enable client-class processing for the DHCP server. On the Edit DHCP Server page, set the *client-class* attribute to enabled. In the CLI, use this command:

```
nrcmd> dhcp enable client-class
```

Step 8 Enable the DHCP server to use LDAP for client entry queries. On the Manage DHCP Server page, set the *client-class* attribute to enabled. In the CLI, use this command:

```
nrcmd> dhcp enable use-ldap-client-data
```

Step 9 If you have more than one LDAP server configured, you can also set them to operate in round-robin or failover mode:

- **round-robin**—The LDAP servers' preference values are ignored and all servers that are configured to handle client queries and accept lease state updates are treated equally.
- **failover**—The DHCP server uses the active LDAP server with the highest preference (lowest preference number). If the preferred server loses its connection or fails, the DHCP server uses the next LDAP server of lower preference (increasing preference number). If the preference values are the same (or not set), the DHCP reverts to round-robin mode with these servers.

Set the LDAP server mode by setting the *ldap-mode* on the Edit DHCP Server page. LDAP failover mode actually performs preferential load balancing. The DHCP server assesses the LDAP connection and error states and how fast the LDAP server responds. In an optimal state, the DHCP server uses the LDAP server with the highest assigned preference (lowest preference number). In a less-than-optimal state, the DHCP server uses the next LDAP server of lower preference (increasing preference number). If the preference values are the same (or unset), the DHCP server reverts to round-robin mode.

In the CLI, use **dhcp set ldap-mode** to set the mode, and **ldap** *server* **set preference** to set the server preferences; for example:

```
nrcmd> dhcp set ldap-mode=failover
nrcmd> ldap ldap-1 set preference=1
nrcmd> ldap ldap-2 set preference=2
```

Note also that, depending on how many threads you have open, as set by using the *connections* attribute (see the "Recommended Values for LDAP" section on page 24-31) between the DHCP and LDAP servers, the DHCP server opens only as many threads as it can before the *query-timeout* expires. The LDAP server might be processing these threads, but it is not servicing the request, because the failover server has now taken over.

Step 10 Show or list the LDAP configuration. Go to the List LDAP Remote Servers page. In the CLI, use:

```
nrcmd> ldap ldap-1
nrcmd> ldap list
nrcmd> ldap listnames
```

Step 11 If in staged scope edit mode, reload the DHCP server.

Unprovisioning Client Entries

You can unprovision LDAP client entries so that the client information remains in LDAP, but the DHCP server treats the client as if that information does not exist. The DHCP server then supplies the client with the default behavior. Configure the search filter set in Step 4 of the preceding section so that the LDAP server does not return a client entry containing a specified attribute with a value.

If you want to unprovision the LDAP entry *givenname*, configure the search filter accordingly. For example:

```
nrcmd> ldap ldap-1 set search-filter=(&(cn=%s)(!(givenname=unprovision)))
```

Whenever the *givenname* attribute in the LDAP client entry is set to the "unprovision" string, the LDAP server does not return the client entry to the DHCP server. In other words, the DHCP server treats the client as if it has no LDAP client entry. This procedure has no measurable performance impact on either the DHCP or the LDAP server.

Configuring Embedded Policies in LDAP

Step 1 Configure an LDAP server for the DHCP server, naming it myserver, for example.

Step 2 Map the LDAP attribute that you want the DHCP server to interpret as the embedded policy to the internal *embedded-policy* property. This example maps the businessCategory LDAP attribute:

```
nrcmd> ldap myserver setEntry query-dictionary businessCategory=embedded-policy
```

Step 3 Add a string to the LDAP attribute that the DHCP server can interpret as an embedded policy. The most practical way to determine what this string should look like is to create a dummy client in the Network Registrar database and extract data from the client embedded policy setup. Note that this dummy client will never be used, because you are using LDAP, and you can subsequently delete it. Have the embedded policy include the option data types that you need.

a. For example, create an embedded client policy for dummy client 1,6,00:d0:ba:d3:bd:3b. Add some reply options and a multivalue option (routers) with an IP address data type:

```
nrcmd> client 1,6,00:d0:ba:d3:bd:3b create
nrcmd> client-policy 1,6,00:d0:ba:d3:bd:3b set v4-reply-options=routers
nrcmd> client-policy 1,6,00:d0:ba:d3:bd:3b setOption routers 1.2.3.4,5.6.7.8
nrcmd> save
```

b. Get the client embedded policy data so that you can display the values:

```
nrcmd> client 1,6,00:d0:ba:d3:bd:3b get embedded-policy
100 Ok
embedded-policy="((ClassName Policy)(name client-policy:00:d0:ba:d3:bd:3b)(option-list
[((ClassName Option)(number 3)(option-definition-set-name dhcp-config)(value
01:02:03:04:05:06:07:08))])(v4-reply-options [routers ])"
```

c. Copy what is between the quotes in the client output in the previous substep and paste it in for the definition of the *businessCategory* LDAP attribute:

```
businessCategory:((ClassName Policy)(name client-policy:00:d0:ba:d3:bd:3b)(option-list
[((ClassName Option)(number 3)(option-definition-set-name dhcp-config)(value
01:02:03:04:05:06:07:08))])(v4-reply-options [routers ])
```

The option values are translated into hexadecimal field syntax, including multiple values that were originally comma-separated.

d. Use the syntax as a model for each new embedded policy entry in LDAP. To see how other option data types appear in the LDAP string, add these options to the client or create further dummy clients with them. Once you extract the data, you can delete the dummy client:

```
nrcmd> client 1,6,00:d0:ba:d3:bd:3b delete
nrcmd> save
```

Here is another example with multiple option definitions:

1. Create a dummy client 1,6,00:d0:ba:d3:bd:3b and an embedded policy attached to that client, with the following options and values:

```
3 routers 10.1.1.1,10.2.1.1
66 tftp-server tftp-server.com
67 bootfile device-boot-file.txt
```

2. Save the changes to the embedded policy, save the client, then extract the following output string into an LDAP client configuration:

```
nrcmd> client 1,6,00:d0:ba:d3:bd:3b get embedded-policy
100 Ok
embedded-policy="((ClassName Policy)(name client-policy:00:d0:ba:d3:bd:3b)(option-list
[((ClassName Option)(number 3)(option-definition-set-name dhcp-config)(value
0a:01:01:01:0a:02:01:01))((ClassName Option)(number 66)(option-definition-set-name
dhcp-config)(value 74:66:74:70:2d:73:65:72:76:65:72:2e:63:6f:6d))((ClassName
Option)(number 67)(option-definition-set-name dhcp-config)(value
64:65:76:69:63:65:2d:62:6f:6f:74:2d:66:69:6c:65:2e:74:78:74))])"
```

Configuring DHCP LDAP Update and Create Services

You can configure the Network Registrar DHCP server to write lease and client data to an LDAP server. The DHCP server can use the client data when responding to DHCP client requests, through the use of the *query* configuration. You can configure the DHCP LDAP service to copy lease state data to attributes on client objects in the LDAP server. The DHCP server converts the lease state data to string form, and uses an update dictionary to map the DHCP data values to the LDAP attributes.

Each time the lease state changes, the DHCP server writes the change to the LDAP server that you configured to store the data. The lease data that the DHCP server writes to LDAP is “write-only” in that it is a copy of the authoritative data in the lease state database.

See Also

Lease State Attributes
Configuring DHCP to Write Lease States to LDAP, page 24-26
Using LDAP Updates, page 24-27
Configuring LDAP State Updates, page 24-27
Configuring LDAP Entry Creation, page 24-29

Lease State Attributes

You can store any of these attributes about the lease state information in your LDAP server:

- ***address***—IP address of this lease.
- ***client-dns-name***—Name the DHCP server attempted to enter into the DNS server for this client.
- ***client-domain-name***—Domain into which to put the client name.
- ***client-flags***—A variety of flags relating to the client.
- ***client-host-name***—DNS name that the client requested the DHCP server to place in the DNS server.
- ***client-id***—Client-id specified by the client, or one synthesized by the DHCP server for this client.
- ***client-mac-addr***—MAC address that the client presented to the DHCP server.



Note Although the MAC addresses in LDAP have to be formatted exactly the way they are formatted by Network Registrar when it creates local client-entries, they are separate instances and thus unique to lease data.

- ***expiration***—The time at which the lease expires.
- ***flags***—Flags for the lease (reserved or deactivated).
- ***lease-renewal-time***—The earliest time in which the client is expected to issue a lease renewal. You can have Network Registrar save this as part of the lease state by using **dhcp enable save-lease-renewal-time** (it is not saved by default).
- ***start-time-of-state***—The time at which the state last changed to its current value.
- ***state***—The lease state can be:
  - Available (1)
  - Deferred (2)
  - Leased (3)
  - Expired (4)
  - Unavailable (5)
  - Released (6)
  - Other_available (7)
  - Disconnected (8)
  - Deleted (9)
- ***vendor-class-identifier***—The name of the vendor, used by clients and servers to exchange vendor-specific information.

Not every lease has all these attributes. The *client-mac-addr* and *client-id* lease state attribute are not present if a client releases its lease or is forced available through Network Registrar. In addition, the *lease-renewal-time* attribute may not be present if the *save-lease-renewal-time* property is disabled through DHCP. Similarly, the *vendor-class-identifier* property may not be present if the *save-vendor-class-id* property is disabled through DHCP, using the CLI.

Configuring DHCP to Write Lease States to LDAP

To have DHCP write lease state updates to LDAP:

Step 1 Choose the LDAP lease state update scheme.

Step 2 Add entries to the directory or modify existing entries to store the lease state information. You may need to extend entries through the addition of attributes or custom object classes.

Step 3 Configure Network Registrar to perform the updates.

Given the flexibility of directories, there are many different ways in which you could choose to store a copy of lease state attributes in a directory. For example, you could choose to store the lease state data as part of an existing entry, or you could store the lease state data independently.

Storing Lease State Data as Part of Existing Entries

You can store lease state data as part of an existing entry. It is even possible to store the client entry, lease state, and employee data in the same entry. As part of the setup for this method, you must decide how you want to store the lease data attributes. You can store data attributes using these methods:

- Map attributes from the entry
- Add attributes to the entry
- Extend the entry by creating a new object class

The advantage is that lease data is stored directly with other client information. The disadvantage is that there are scenarios, albeit unlikely, related to client-class and reservations that could result in stale data being in the directory for a short period of time when the server moves a client off a lease.

Note If the lease whose state is being updated does not have a client, it will not have an associated MAC address. This situation occurs when a client gets a lease, and then is moved off that lease by client-class processing. It can also occur when a client has a pre-existing lease and a reservation for a different lease in the same LAN segment. When the reserved lease is available, the server moves the client off its existing lease and onto the reservation. Both of these transfers result in an LDAP update for the old lease without a client MAC address. This is generally not a problem, because the update for the new lease (which has an associated MAC address) should come through.

Also, this method requires two LDAP interactions to write the lease information. When updating lease state information, the DHCP LDAP service contacts the directory twice because when updating an entry it is not enough just to know how to find the entry. You must specifically know the *dn* of the entry.

The DHCP LDAP service first finds the appropriate entry in the directory by using one of the lease state attributes that you chose (preferably the MAC address) as the search criteria. This is necessary because none of the lease state attributes is part of the *dn* of the entry. When the DHCP LDAP service locates the

entry, the *dn* is returned. The DHCP LDAP service then updates that same entry with the appropriate information. For an example how to use this method, see the "Configuring LDAP State Updates" section on page 24-27.

Storing Lease State Data Independently

You can store lease state data by IP address in its own entries. This method results in a copy of the server lease database in a directory, and is the most straightforward way to configure the database. As part of the setup for this method, create new entries for each IP address that the server can serve. The advantage to this method is that there are no scenarios in which the lease state data in the directory will be stale. The disadvantage is that lease data is not stored directly with other associated client information.

To update the lease state information, the DHCP LDAP service contacts the directory service once. When performing the update, the service uses the IP address to construct the *dn*.

Using LDAP Updates

There are two ways you can use the LDAP update feature:

- Keep track of clients that use LDAP client entry information and to associate some of the attributes of that LDAP host with lease state attributes.
- Create and update objects that can be located by their IP address. When Network Registrar creates these objects, it can make a level of LDAP objects that matches (or is) the DHCP server lease state.

When using Network Registrar, you should be aware that:

- The DHCP server only reads from a single object and writes to a single object. You can use separate objects to hold the client entry data read and the lease state date written, but Network Registrar cannot read some attributes from one object and some from another.
- The performance of LDAP queries, like all database access, depends on indexed attributes. If you did not index the attributes that you configure to use in query filters, you will experience poor performance.
- When configured for update, Network Registrar does not create new objects in the directory. The DHCP server only writes to existing objects. However, if you configure for update and create, it does create new objects (if no object exists).

Configuring LDAP State Updates

There are two options available for performing a lease state update to an LDAP server:

- ***update-search-path***—The DHCP server first queries to locate the *dn* for an update.
- ***dn-format***—The server is provided with the *dn* for an update. In other words, the DHCP performs a direct update without having to query before an update.

Option 1: Using the update-search-path Option

The following example illustrates the first option, *update-search-path*. It shows what to do when the distinguished name (*dn*) of an LDAP object cannot be constructed from data that is available in the lease state. The DHCP server creates an LDAP query based on the *update-search-xxx* information, locates the LDAP object, and uses its *dn* to issue an LDAP update.

The example shown in Table 24-1 on page 24-28 assumes that you are using the standard LDAP organizational person object class attributes to hold lease update data.

Table 24-1 LDAP-to-DHCP Mapping Example

Attribute	DHCP Lease Entry Mapping
uid	address (IP address)
carlicense	state (lease state)

Step 1 Tell DHCP about the LDAP server by supplying the server hostname in the LDAP configuration.

Step 2 Configure the credentials to use when connecting to the LDAP server. This CLI example sets the administrator to joe and his password to access. Use the distinguished name (*dn*) for the user:

```
nrcmd> ldap myserver set username="cn=joe,o=Example Corporation,c=US" password=access
```

Step 3 Configure the *update-search-path* attribute, which is the starting point in the directory for the objects that the DHCP server will update. You can also set the update search scope. This CLI example sets the search path to begin at the organizational unit (ou) IT, the organization Example Corporation, and country US. The update search scope is set to SUBTREE:

```
nrcmd> ldap myserver set update-search-path="ou=IT,o=Example Corp,c=US"
       update-search-scope=SUBTREE
```

Step 4 Set the ID of the attribute you want to use to search for the LDAP object that will be updated. This CLI example sets the search attribute to be the client MAC address:

```
nrcmd> ldap myserver set update-search-attribute=client-mac-addr
```

Step 5 Configure a filter expression into which the *update-search-attribute* attribute should be formatted. This expression must contain a "%s," which indicates where the search attribute data should be substituted. Here is a CLI example:

```
nrcmd> ldap myserver set update-search-filter=(cn=%s)
```

Step 6 Configure the *update-dictionary* attribute, which allows you to identify the LDAP attributes that you want set with the values of the corresponding lease state attributes. This example specifies that the LDAP UID should be updated to contain the IP address, and that the *carlicense* attribute should be updated to contain the DHCP lease state information. Using the CLI:

```
nrcmd> ldap myserver setEntry update-dictionary uid=address carlicense=state
```

Step 7 Enable updates for the new LDAP server. Here is a CLI example:

```
nrcmd> ldap myserver enable can-update
```

Step 8 If in staged scope edit mode, reload the DHCP server.

Option 2: Using the dn-format Option

This example illustrates using the second option, *dn-format*:

Step 1 Tell DHCP about the LDAP server by supplying the server hostname in the LDAP configuration.

Step 2 Configure the credentials to use when connecting to the LDAP server. This CLI example sets the administrator to joe and his password to access. Use the *dn* for the user:

```
nrcmd> ldap myserver_option2 set username="cn=joe,o=Example Corporation,c=US"
       password=access
```

Step 3 Use the *dn-format* string to specify where in the LDAP server database hierarchy you want to begin searching for the update. Here is a CLI example:

```
nrcmd> ldap myserver_option2 set dn-format="cn=\"%s\",ou=IT,o=Example Corp,c=US"
```

Step 4 Set the *dn-attribute* attribute to which you want the *dn-format* string to refer. This CLI example sets the *dn-attribute* to be the client MAC address:

```
nrcmd> ldap myserver_option2 set dn-attribute=client-mac-addr
```

Step 5 Specify the entries to be updated. Using the CLI:

```
nrcmd> ldap myserver_option2 setEntry update-dictionary uid=address carlicense=state
```

Step 6 Enable the *can-update* attribute. Here is a CLI example:

```
nrcmd> ldap myserver_option2 enable can-update
```

Step 7 If in staged scope edit mode, reload the DHCP server.

Configuring LDAP Entry Creation

This section explains how to create LDAP entries. LDAP entry creation provides the ability to locate entries and update them with current lease information. Entries are created only if a state update operation fails because it cannot locate an entry.

After performing the steps in the previous example, follow these steps in the CLI:

Step 1 Set the *dn-attribute* property for the LDAP server for the lease object attribute, such as the *client-mac-addr* field, and set the *dn-format* string. Here is a CLI example:

```
nrcmd> ldap myserver set dn-attribute=client-mac-addr
       dn-format="cn=\"%s\",ou=IT,o=Example Corp,c=US"
```

This step is required only if you configure the lease state updates using the *update-search-path* option. (See "Option 1: Using the update-search-path Option" section on page 24-27). Skip this step if you configure lease state updates using the *dn-format* string. (See "Option 2: Using the dn-format Option" section on page 24-29.)

Step 2 Specify the *dn* of the entry to be created when combined with the existing *dn-attribute* property. Here is a CLI example:

```
nrcmd> ldap myserver set dn-create-format="cn=\"%s\",ou=IT,o=Example Corp,c=US"
```

The Network Registrar *client-mac-addr* field uses the form **1,6:*xx*:*xx*:*xx*:*xx*:*xx*:*xx***. Since the comma character is a special separator in LDAP, you must use the **\"** characters to quote the *dn*.

Step 3 Using the *create-dictionary* property, establish mappings between LDAP attributes and lease state attributes by entering a series of name=value pairs. The LDAP attributes indicate the entry attributes set to the value of their corresponding lease state attributes. In the CLI:

```
nrcmd> ldap myserver setEntry create-dictionary sn=client-host-name
nrcmd> ldap myserver setEntry create-dictionary givenname=client-class-name
nrcmd> ldap myserver setEntry create-dictionary localityname=client-domain-name
```

Step 4 Using the *create-object-classes* property, specify the object classes to be used when creating the entry. Here is a CLI example:

```
nrcmd> ldap myserver set create-object-classes=
       "top,person,organizationalPerson,inetorgperson"
```

Step 5 Enable entry creation for the LDAP server myserver. Here is a CLI example:

```
nrcmd> ldap myserver enable can-create
```

Note Enable the *can-update* attribute before you enable the *can-create* attribute. For an example, see the "Configuring LDAP State Updates" section on page 24-27.

Step 6 If in staged scope edit mode, reload the DHCP server.

Step 7 To see if creation, queries, and updates were successful, view the LDAP log settings.

Troubleshooting LDAP

The following sections include some advice on fine-tuning and detecting failures of the LDAP server.

See Also

LDAP Connection Optimization, page 24-30
Recommended Values for LDAP, page 24-31

LDAP Connection Optimization

You can optimize LDAP connections by using separately tunable read and write objects. This CLI example tunes write (create and update) operations, which require longer server processing:

```
nrcmd> ldap LDAP-Write create csrc-ldap password=changeme port=389 preference=1
nrcmd> ldap LDAP-Write setEntry query-dictionary csrcclientclasas=client-class-name
nrcmd> ldap LDAP-Write set
       search-filter=(&(macaddress=%s)(|(crscclassname=Computer)(csrcclassname=Modem)))
nrcmd> ldap LDAP-Write set search-path=csrcprogramname=csrc,o=NetscapeRoot
nrcmd> ldap LDAP-Write set
       username=uid=admin,ou=Administrators,ou=TopologyManagement,o=NetscapeRoot
nrcmd> ldap LDAP-Write disable can-query
nrcmd> ldap LDAP-Write enable can-create
nrcmd> ldap LDAP-Write enable can-update
nrcmd> ldap LDAP-Write enable limit-requests
nrcmd> ldap LDAP-Write set connections=2 max-requests=8 timeout=10s
```

This CLI example tunes read (query) operations, which require shorter server processing:

```
nrcmd> ldap LDAP-Read create csrc-ldap password=changeme port=389 preference=1
nrcmd> ldap LDAP-Read setEntry query-dictionary csrcclientclasas=client-class-name
nrcmd> ldap LDAP-Read set
       search-filter=(&(macaddress=%s)(|(crscclassname=Computer)(csrcclassname=Modem)))
nrcmd> ldap LDAP-Read set search-path=csrcprogramname=csrc,o=NetscapeRoot
nrcmd> ldap LDAP-Read set
       username=uid=admin,ou=Administrators,ou=TopologyManagement,o=NetscapeRoot
nrcmd> ldap LDAP-Read enable can-query
nrcmd> ldap LDAP-Read disable can-create
nrcmd> ldap LDAP-Read disable can-update
nrcmd> ldap LDAP-Read enable limit-requests
nrcmd> ldap LDAP-Read set connections=3 max-requests=12 timeout=4s
```

Recommended Values for LDAP

Table 24-2 shows recommended values for some key LDAP attributes.

Table 24-2 Recommended Values for LDAP Attributes

Attribute and Value	Description
connections=5 to 25	Number of connections that the server should make to an LDAP server. This is primarily a performance tuning parameter. The default is one connection. In some cases, more than one connection can improve overall throughput. The amount depends on the load on the LDAP server. With many applications using LDAP, five connections would be appropriate; with just Network Registrar using LDAP, 25 would be appropriate.
threadwaittime=2	Interval (in milliseconds) at which each LDAP client connection polls for results, if it has outstanding queries or updates.
query-timeout=3	Network Registrar DHCP servers fail over at the *query-timeout* interval if *failover* and *can-query* are set. The default setting is 3 seconds and is recommended (in that it is less than the default 4-second *drop-old-packets* value for the DHCP server, after which time the connection is considered inactive and the LDAP server as “unhealthy”).
timeout=10	Number of seconds an LDAP request remains in a connection queue before being declared stale and timing out. Any response the DHCP client receives after the client timeout period is stale. The default is 10 seconds, which is recommended. Network Registrar DHCP servers fail over at the *timeout* interval if *failover* and *can-update* or *can-create* are enabled.



CHAPTER 25

Using Expressions

Network Registrar provides enhanced client-class support. You can now place a request into a client-class based on the contents of the request, without having to register the client in the client database. Also, you can now place requests in a client-class based on the number of the active leases of a subscriber, allowing limitations on the level of service offered to various subscribers. This is possible through the special DHCP options processing using expressions.

You can set the limitation on subscriber addresses based on values in the DHCP *relay-agent-info* option (option 82, as described in RFC 3046). These values do not need to reveal any sensitive addresses. You can create values that relate an individual to a subscriber by creating an expression that evaluates the incoming DHCPDISCOVER request packets against option 82 suboptions (*remote-id* or *circuit-id*) or other DHCP options. The expression is a series of *if* statements that return different values depending on what is evaluated in the packet. This, in effect, calculates the client-class in which the subscriber belongs, and limits address assignment to the scope of that client-class.

Note Expressions are not the same as DHCP extensions. Expressions are commonly used to create client identities or look up clients. Extensions (see Chapter 29, "Using Extension Points") are used to modify request or response packets. The expressions described here are also not the same as regex.

See Also

Using Expressions
Entering Expressions, page 25-2
Creating Expressions, page 25-3
Expression Examples, page 25-21
Debugging Expressions, page 25-24

Using Expressions

Expression processing is used in several places:

- **Calculating a client-class**—*client-class-lookup-id*. This expression determines the client-class based on the contents of the incoming packet.
- **Creating the key to look up in the client-entry database**—*client-lookup-id*. This accesses the client-entry database with the key resulting from the expression evaluation.
- **Creating the ID to use to limit clients of the same subscriber**—*limitation-id*. This is the ID to use to check if any other clients are associated with this subscriber.

This kind of processing results in this scenario:

1. The DHCP server tries to get a client-class based on a *client-class-lookup-id* expression. If it cannot calculate the client-class, it uses the usual MAC address method to look up the client.
2. If the server can calculate the client-class, it determines if it needs to do a client-entry lookup, based on evaluating a *client-lookup-id* expression that returns a *client-lookup-id*. If it has such an ID, it uses it to look up the client. If it does not have such an ID, it uses the calculated client-class value to assign addresses.
3. If the server uses the *client-lookup-id* and finds a client-entry, it uses the data for the client. If it cannot find a client-entry, it uses the calculated or default client-class data.

You set the upper limit on assigned addresses to clients on a network or LAN segment having an identical *limitation-id* value on the policy level. Set this upper limit as a positive integer using the *limitation-count* attribute for the policy.

The values to set for limiting IP addresses to subscribers are:

- For a policy, set the *limitation-count* attribute to a positive integer.
- For a client-class, set the *limitation-id* and *client-lookup-id* attributes to an expression, and set the *over-limit-client-class-name* attribute to a client-class.
- For a client, set the *over-limit-client-class-name* attribute to a client-class.

The expressions to use are described in the "Creating Expressions" section on page 25-3.

Entering Expressions

You can include simple expressions as such in the attribute definition, or include more complex ones in an expression file and reference the file in the attribute definition. Either way, the maximum allowable characters is 16 KB.

Simple expressions must adhere to these rules when you enter them in the CLI:

- They must be limited to a single command line.
- The entire expression must be enclosed in double quotes (" ").
- Embedded double quotes must be escaped with a backslash (\).

Here is an example of a simple expression to set the *client-class-lookup-id*:

```
"\"limit\""
```

Here is a slightly more extensive example to set the client-class *limitation-id*:

```
"(request option 82 \"circuit-id\")"
```

You must enter any more complex expressions, that are not limited to one line or that you want to format for comprehension, in a file and reference it in the attribute definition prefixed by the "at" symbol (@):

```
@cclookup.txt
```

The syntax of the expression in the file does not have the extra requirements (as to spacing and escaping of characters) of the simple expression. It can also include comment lines, prefixed by the pound sign (#), double-slash (//), or a semicolon (;), and terminated at the end of line.

For example, in the cclookup.txt file:

```
// Expression to calculate client-class based on remote-id
(try (if (equal (request option "relay-agent-info" "remote-id") (request chaddr))
        "cm-client-class"
        "cpe-client-class")
        "<none>")
```

The IPv6 version of the previous example (using option numbers) is:

```
// Expression to calculate client-class based on DOCSIS 3.0 cm-mac-address
(try (if (equal (request option 17 enterprise-id 4491 36)
    (or (request relay option 17 enterprise-id 4491 1026) "none"))
        "v6-cm-client-class"
        "v6-cpe-client-class")
        "<none>")
```

You can also write the previous expression by substituting option names in place of numbers:

```
// Expression to calculate client-class based on DOCSIS 3.0 cm-mac-address
(try (if (equal
    (request option "vendor-opts" enterprise-id "dhcp6-cablelabs-config" "device-id")
    (or (request relay option "vendor-opts" enterprise-id "dhcp6-cablelabs-config"
    "cm-mac-address") "none"))
        "v6-cm-client-class"
        "v6-cpe-client-class")
        "<none>")
```

The **or** function in the example ensures that if the packet was not relayed or if the relay agent did not add the option, then the server assumes the client to be a CPE and not a cable modem (CM).

Creating Expressions

Using DHCP expressions, you can retrieve, process, and make decisions based on data in incoming DHCP packets. You can use them for determining the client-class of an incoming packet, and create the equivalence key for option 82 limitation support. They provide a way to get information out of a packet and individual options, a variety of conditional functions to allow decisions based on information in the packet, and data synthesis capabilities where you can create a client-class name or key.

The expression to include in an expression file that would describe the example in the "Typical Limitation Scenario" section on page 24-16 would be:

```
// Begins the try function
(try
    (or (if (equal (request option "relay-agent-info" "remote-id") (request chaddr))
        "cm-client-class")
        (if (equal (substring (option "dhcp-class-identifier") 0 6) "docsis")
            "docsis-cm-client-class")
        (if (equal (request option "user-class") "alternative-class")
            "alternative-cm-client-class")
    )
    <none>
)
// Ends the try function
```

The expression uses the **or** function and evaluates three **if** functions. In a simpler form, you can calculate a client-class and include this expression in the cclookup.txt file.

```
// Expression to calculate client-class based on remote-id
(try (if (equal (request option "relay-agent-info" "remote-id") (request chaddr))
        "cm-client-class"
        "cpe-client-class")
        "<none>")
```

Refer to this file to use the expression to set the client-class lookup ID for the server:

```
nrcmd> dhcp set client-class-lookup-id=@cclookup.txt
```

You can generate a limitation key by trying to get the *remote-id* suboption from option 82, and if unable, to use a standard MAC blob key. Include an expression in a file and set the limitation ID to it in the **cclimit.txt** file:

```
// Expression to use remote-id or standard MAC
(try (request option "relay-agent-info" "remote-id") 00:d0:ba:d3:bd:3b)
```

```
nrcmd> client-class name limitation-id=@cclimit.txt
```

Expression Syntax

Expressions consist solely of functions and literals. Its syntax is similar to that of Lisp. It follows many of the same rules and uses Lisp functions names where possible. The basic syntax is:

```
(function argument-0 ... argument-n)
```

A more useful example is:

```
(try (if (equal (request option "relay-agent-info" "remote-id") (request chaddr))
        "cm-client-class"
        "cpe-client-class")
        "<none>")
```

This example compares the *remote-id* suboption of the *relay-agent-info* option (option 82) with the MAC address in the packet, and if they are the same, returns “cm-client-class,” and if they are different, returns “cpe-client-class.” (If the expression cannot evaluate the data, the **try** function returns a “<none>” value—see the “Expressions Can Fail” section on page 25-6.) The intent is to determine if the device is a cable modem (where, presumably, the *remote-id* equals the MAC address) and, if so, put it into a separate client-class than the customer premise equipment or PC. Note that both functions and literals are expressions. The previous example shows a function as an expression. For literals, see the “Literals in Expressions” section on page 25-5.

Expression Datatypes

The datatypes that expressions support are:

- **Blob**—Counted series of bytes, with a minimum supported length of 1 KB.
- **String**—Counted series of NVT ASCII characters, not terminated by a zero byte, with a minimum supported length of 1 KB.
- **Signed integer**—32-bit signed integer.
- **Unsigned integer**—32-bit unsigned integer.

Note that there is no IP address datatype; an IP address is a 4-byte blob. All numbers are in network byte order. See the "Datatype Conversions" section on page 25-19.

Literals in Expressions

A variety of literals are included in the expression capability:

- **Signed integers**—Normal numbers that must fit in 32 bits.
- **Unsigned integers**—Normal unsigned numbers that must fit in 32 bits.
- **Blobs**—Hex bytes separated by colons. For example, 01:02:03:04:05:06 is a 6-byte blob with the bytes 1 through 6 in it. This is distinct from "01:02:03:04:05:06" (a 17-byte string). The string is related to the blob by being the text representation of the blob. For example, the expression `(to-blob "01:02:03")` returns the blob 01:02:03. Note that you cannot create a literal representation of a one-byte blob, as 01 will turn into an integer. To get a one-byte blob containing a 1, you would use the expression `(substring (to-blob 1) 3 1)`. The 3 indicates the offset to extract the fourth byte of the 4-byte integer (00:00:00:01), with the 1 being the number of bytes extracted, with a result of "01."
- **String**—Characters enclosed in double quotes. For example, "example.com" is a string, as is "01:02:03:04:05:06." To place a quote in a literal string, escape it with a backslash (\), for example:

  ```
  "this has one \"quote"
  ```

Integer literals (signed and unsigned) are assumed to be in base10. If they start with a 0, they are considered octal; if they start with 0x, they are considered hexadecimal. Some examples of literals:

- "hello world" is a string literal (and a perfectly valid expression).
- 1 is an unsigned integer literal (also a perfectly valid expression). It contains 4 bytes, the first three of which are zero, and the last of which contains a 1 in the least significant bit.
- 01:02:03 is a blob literal containing three bytes, 01, 02, and 03.
- –10 is a signed integer literal containing four bytes with the twos-complement representation of decimal -10.

Expressions Return Typed Values

With few exceptions, the point of an expression is to return a value. The expression configured to determine a client-class is configured in the DHCP server property *client-class-lookup-id*. When this expression is evaluated, the DHCP server expects it to return a string containing the name of a client-class, or the string **<none>**.

Every function returns a value. The datatype of the value may depend on the datatype of the argument or arguments. Some expressions only accept arguments of a certain datatype; for example:

```
(+ argument0 argument1)
```

In most cases, a function that requires a certain datatype for a particular argument tries to convert the argument that it gets to the proper datatype. For example, `(+ "1" 2)` returns 3, because it successfully converts the string literal "1" into a numeric 1. However, `(+ "one" 2)` causes an error, because "one" does not convert successfully into a number. In general, the expression evaluator tries to do the right thing as much as possible when making datatype conversion decisions.

Expressions Can Fail

While some of the functions that make up an expression operate correctly on any datatype or value, many do not. In the previous section, the **+** function would not convert the string literal "one" into a valid number, so the evaluation of that function failed. When a function fails to evaluate, its calling function also fails, and so on, until the entire expression fails. A failed expression evaluation has different consequences depending on the expression involved. In some cases, it can cause the packet to be dropped, while in others it only generates a warning message.

You can prevent the evaluation from failing by using the **(try** *expression failure-expression***)** function. The **try** function evaluates the expression and, if successful, the value of the function is the value of the *expression*. If the evaluation fails (for whatever reason), the value of the function is the value of the *failure-expression*. The only situation where a **try** function itself fails is if the *failure-expression* evaluation fails. Thus, you should be careful what expression you define as a *failure-expression*. A string literal is a safe bet. Thus, protecting the evaluation of the *client-class-lookup-id* with a **try** function is a good idea. The previously cited example shows how this can work:

```
(try (if equal (request option "relay-agent-info" "remote-id") (request chaddr))
        "cm-client-class"
        "cpe-client-class")
    "<none>")
```

If evaluating the **if** function fails in this case, the value of the *client-class-lookup-id* expression is <none>. It could have been a client-class name instead, of course.

Expression Functions

Table 25-1 lists the expression functions. Expressions must be enclosed in parentheses.

Table 25-1 Expression Functions

Function Description	Example
(and *arg1 ... argn***)**	**(and "hello" "world")** returns "world" **(and (request option 82 1) (request option 82 2))** returns option-82 sub-option 2 if both option-82 sub-option 1 and sub-option 2 are present in the request
The **and** function returns a value that is the datatype of *argn* or null. It evaluates its arguments in order from left to right (the arguments can evaluate to a datatype). If any argument evaluates to null, it stops evaluating the arguments and returns null. Otherwise, it returns the value of the last argument, *argn*.	
(as-blob *expr***)**	**(as-blob "hello world")** returns the blob 68:65:6c:6c:6f:20:77:6f:72:6c:64
The **as-blob** function treats *expr* as if it were a blob. If *expr* evaluates to a string, the bytes that make up the string become the bytes of the blob that is returned. If *expr* evaluates to a blob, that blob is returned unmodified. If *expr* evaluates to either kind of integer, a 4-byte blob containing the bytes of the integer is returned. (See Table 25-2 on page 25-19.)	
(as-sint *expr***)**	**(as-sint ff:ff:ff:ff)** returns -1 **(as-sint 2147483648)** returns an error
The **as-sint** function treats *expr* as if it were a signed integer. If *expr* evaluates to a string or blob of 4 bytes or less, the function returns a signed integer constructed out of those bytes (if longer than 4 bytes, it returns an error). If *expr* evaluates to a signed integer, it returns the value unchanged; if an unsigned integer, it returns a signed integer with the same bit value. (See Table 25-2 on page 25-19.)	

Table 25-1 Expression Functions (continued)

Function Description	Example
(**as-string** *expr*)	**(as-string 97)** returns "a" **(as-string 68:65:6c:6c:6f:20:77:6f:72:6c:64)** returns "hello world" **(as-string 0)** returns an error.
The **as-string** function treats *expr* as if it were a string. If *expr* evaluates to a string, it returns that string. If *expr* evaluates to a blob, it returns a string constructed from the bytes in the blob, unless they are nonprintable ASCII values, which returns an error. If *expr* evaluates to an integer, it considers its value to be the ASCII value for a single character and returns a string consisting of that one character, unless it is nonprintable, which returns an error. (See Table 25-2 on page 25-19.)	
(**as-uint** *expr*)	**(as-uint -2147483648)** returns the unsigned integer 2147483648 **(as-uint -1)** returns the unsigned integer 4294967295 **(as-uint ff:ff:ff:ff)** returns the unsigned integer 4294967295
The **as-uint** function treats *expr* as if it were an integer. If *expr* evaluates to a string or blob of 4 bytes or less, it returns an unsigned integer constructed from those bytes; if longer than 4 bytes, it returns an error. If the result is an unsigned integer, it returns the argument unchanged; if a signed integer, it returns an unsigned integer with the same bit value (see Table 25-2 on page 25-19).	
(**ash** *expr shift*)	**(ash 00:01:00 1)** returns the blob 00:02:00 **(ash 00:01:00 -1)** returns the blob 00:00:80 **(ash 1)** returns the unsigned integer 2
The **ash** function returns an integer or blob with the bits shifted by the *shift* amount. The *expr* can evaluate to an integer, blob or string. If *expr* evaluates to a string, this function tries to convert it to a signed integer, and if that fails, to a blob. If both fail, it returns an error. The *shift* must evaluate to something that is convertible to a signed integer. If *shift* is positive, the shift is to the left; if negative, the shift is to the right. If *expr* results in a signed integer, the right shift is with sign extension. If *expr* results in an unsigned integer or blob, a right shift shifts zero bits in on the most significant bits.	
(**base32encode** *arg*) The **base32encode** function returns a base-32 encoded string of bytes in the *arg*.	
(**bit-and** *arg1 arg2*) (**bit-or** *arg1 arg2*) (**bit-xor** *arg1 arg2*) (**bit-eqv** *arg1 arg2*) (**bit-andc1** *arg1 arg2*) (**bit-andc2** *arg1 arg2*) (**bit-orc1** *arg1 arg2*) (**bit-orc2** *arg1 arg2*)	**(bit-and 00:20 00:ff)** returns 00:20 **(bit-or 00:20 00:ff)** returns 00:ff **(bit-xor 00:20 00:ff)** returns 00:df **(bit-andc1 00:20 00:ff)** returns 00:df
These **bit** functions return the result of a bit-wise boolean operation on the two arguments. The data type of the result is a signed integer if both arguments result in either kind of integer, otherwise the result is a blob. The *arg1* and *arg2* arguments must evaluate to two integers, two blobs of equal length, or one integer and one blob of length 4. If either argument evaluates to a string, the function tries to convert the string to a signed integer, and if that fails, to a blob. After this conversion, the results must match the criteria mentioned above. If these conditions are not met, it returns an error. Operations with **c1** and **c2** indicate that the first and second arguments, respectively, are complemented before the operation.	

Table 25-1 Expression Functions (continued)

Function	Example
Description	
(**bit-not** *expr*)	**(bit-not ff:ff)** returns 00:00 **(bit-not 1)** returns 4294967295 **(bit-not "hello world")** returns an error
The **bit-not** function returns a value that is the bit-by-bit complement of *expr*. The datatype of the result is the same as the result of evaluating *expr* and any subsequent conversions, if the result was a string. The expression must evaluate to an integer of either type, or a blob. If it evaluates to a string, the function tries to convert it to a signed integer; if that fails, to a blob, and if that fails, returns an error.	
(**byte** *arg1*)	**(byte 150)** returns 0x96 **(byte 0x96)** returns 0x96
The **byte** function eases creation of one-byte blobs. It returns this blob depending on the data type: • **sint**, **uint**—Returns a low-order byte of type integer. • **blob**—Returns the last byte in the blob. • **string**—Returns the last bye in the string.	
(**comment** *comment expr1… exprn*)	**(comment "this is a comment that won't get lost" (request option 82 1))**
The **comment** function inserts a comment string into an expression and returns the value of the last expression (*exprn*).	
(**concat** *arg1 … argn*)	**(concat "hello " "world")** returns "hello world" **(concat -1 "world")** returns an error **(concat -1 00:01:02)** returns the blob ff:ff:ff:ff:00:01:02
The **concat** function concatenates the values of the arguments into a string or blob (ignoring null arguments). The first argument (*arg1*) must evaluate to a string or a blob; if it evaluates to an integer, the function converts it to a blob. The datatype of *arg1* (after any conversion) determines the datatype of the result. The function converts all subsequent arguments to the datatype of the result, and if this conversion fails, returns an error.	
(**create-prefix** *template prefix*)	**(create-prefix "cm-prefix" (create-prefix-range 32 0x1))**
Use the **create-prefix** function in DHCPv6 link templates to create a prefix based on a predefined prefix template name and the prefix, including the link VPN. The *prefix* argument can be the prefix name, but can also be the **create-prefix-addr** or **create-prefix-range** function. (Note that you can use the **list** function to combine multiple operations.)	
(**create-prefix-addr** *prefix interface-id*)	
The **create-prefix-addr** function creates an IPv6 address string based on the prefix name and interface ID, which is the lower 64-bit address in the prefix (which need not be contained in the parent prefix). Used in the *prefix-expr* and *options-expr* attributes in DHCPv6 link templates, and the *range-expr* and *options-expr* attributes in prefix templates.	

Table 25-1 Expression Functions (continued)

Function	Example
Description	
(**create-prefix-range** *prefix-length-delta value-to-insert*)	`(create-prefix-range 32 0x1)`
Use the **create-prefix-range** function in DHCPv6 link templates and prefix templates to create a range value from a prefix by increasing the prefix length, thereby creating a child prefix. The *prefix-length-delta* is how many bits longer to make the prefix and must be a value from 1 through 32. The *value-to-insert* is the number of bits to insert in the new prefix length determined by the *prefix-length-delta*; it is the *n*th occurrence of the child prefix. Its value can be 0, but is limited to less than 2 to the power of the *prefix-length-delta*. The prefix value that the function is based on is either the template root prefix if applying a link template to a link, or the prefix address if applying a prefix template to a prefix.	
(**create-v6-option** *opt val*)	
Use the **create-v6-option** function in the *options-expr* attribute in DHCPv6 link templates and prefix templates to create a DHCPv6 option. The *opt* can be the literal string or integer identifying the option. The *val* is the string representation of the option value, as defined by the option TLV value.	
(**decodeRFC1035** *blob-string*)	
The **decodeRFC1035** function returns the string for the DNS wire-encoded domain name. See also the **encodeRFC1035** function.	
(**dotimes** (*var count-expr* [*result-expr*]) *exp1 … expn*)	`(let (x y) (setq x 01:02:03) (dotimes (i (length x)) (setq y (concat (substring x i 1) y)))) returns null, but after the dotimes y is the reverse of x` `(dotimes (i 10) (setq i 1)) loops forever!`
The **dotimes** function creates an environment with a single local integer variable, *var*, which is initially set to zero, and evaluates *exp1* through *expn*. It then increments *var* by one, and if it is less than *count-expr*, evaluates *exp1* through *expn* again. When *var* is equal to or greater than *count-expr*, the function evaluates *result-expr* and returns it as the result of the entire **dotimes**. If there is no *result-expr*, the function returns null. The *var* defines a local variable, and must be an alphabetic name. The *count-expr* must evaluate to an integer or be convertible to one. The *exp1* through *expn* are expressions that can evaluate to any data type. The *result-expr* is optional, and if it appears, it can evaluate to any data type. When the function evaluates *count-expr*, *var* is not bound and cannot appear in *count-expr*. Alternatively, *var* is bound for the evaluation of *result-expr* and has the value of *count-expr*. If *result-expr* is omitted, the function returns null. **Note** Be careful changing the value of *var* in *exp1* through *expn*, because you can easily create an infinite loop (see the example).	
(**encodeRFC1035** *string*)	
The **encodeRFC1035** function returns the blob-formatted DNS wire-encoded data for *string*. See also the **decodeRFC1035** function.	

Table 25-1 Expression Functions (continued)

Function Description	Example
(environmentdictionary {**get** | **put** *val* | **delete**} *attr***)**	`nrcmd> dhcp set initial-environment-dictionary=first=one,second=2` **(environmentdictionary get "first")** returns "one" **(environmentdictionary get "second")** returns "2" (note string 2) **(environmentdictionary put "two" "second")** returns "second" **(environmentdictionary delete "first")** returns null
The **environmentdictionary** function gets, puts, or deletes a DHCP extension environment dictionary attribute value. The *val* is the value of the attribute and *attr* is the attribute name. Both are converted to a string regardless of their initial datatype. The initial environment dictionary cannot be changed, but it can be shadowed (you can redefine something that is in the initial dictionary, but if you remove it, then the original initial value is still there). Note that the **get** keyword is not optional for a "get."	
(equal *expr1 expr2 expr3***)** **(equali** *expr1 expr2 expr3***)**	**(equal (request option "dhcp-class-identifier") "docsis")** returns the string "docsis" if the value of the option dhcp-class-identifier is a string identical to "docsis" **(equali "abc" "ABC")** returns "ABC" **(equal "abc" "def")** returns null **(equal "ab" (as-string 61:62)) "this is true")** returns "this is true" **(equal "ab" 61:62 "this is not true")** returns null **(equal 01:02:03 01:02:03)** returns 01:02:03 **(equal (as-blob "ab") 61:62)** returns null **(equal 1 (to-blob 1))** returns null **(equal (null) (request option 20))** returns "*T*" if there is no option 20 in the packet
The **equal** function evaluates the equivalency of the result of evaluating *expr1* and *expr2*. If they are equal, it returns: 1. The value of *expr3*, if specified, else 2. The value (and datatype, after possible string conversion) of *expr2*, as long as *expr2* is not null, else 3. The string "*T*" (since returning null would incorrectly indicate a failed comparison). If *expr1* and *expr2* are not equal, the function returns null. The arguments can be any datatype. If different, the function converts them to strings (which cannot fail) before comparing them. Note that any string conversion is performed using the equivalent of **(to-string** ...**)**. Thus, the blob 61:62 is not equal to the "ab" string. Note also that a one-byte blob 01 is not equal to a literal integer 1 (both are converted to strings, and the "01" and "1" strings are not equal). The **equali** function is identical to the **equal** function, except that if the comparison is for strings (either because string arguments were used or because the arguments were converted to strings), a case insensitive comparison is used.	
(error) The **error** function returns a "no recovery" error that causes the entire expression evaluation to fail unless there is a **try** function above the **error** function evaluation.	

Table 25-1 Expression Functions (continued)

Function Description	Example
(**if** *cond* [*then else*])	**(if (equali (substring (request option "dhcp-class-identifier") 0 6) "docsis") (request option 82 1))** returns sub-option 1 of option 82 if the first six characters of the dhcp-class-identifier are "docsis" in any case; otherwise returns null
The **if** function evaluates the condition expression *cond* in an *if-then-else* sense. If *cond* evaluates to a value that is nonnull, it returns the result of evaluating the *then* argument; otherwise it returns the result of evaluating the *else* argument. Both *then* and *else* are optional arguments. If you omit the *then* and *else* arguments, the function simply returns the results of evaluating the *cond* argument. If you omit the *else* argument and *cond* evaluates to null, the function returns null. There are no restrictions on the data types of any of the three arguments.	
(**ip-string** *blob*)	**(ip-string 01:02:03:04)** returns "1.2.3.4" **(ip-string -1)** returns "255.255.255.255" **(ip-string (as-blob "hello world")** returns "104.101.108.108"
The **ip-string** function returns the string representation of the four-byte IP address *blob* in the form *"a.b.c.d*". The single argument *blob* must evaluate to a blob or be convertible into one. If the blob exceeds four bytes, the function uses only the first four to create the IP address string. If the blob has fewer bytes, the function considers the right-most bytes as zero when it creates the IP address string.	
(**ip6-string** *blob*)	**(ip-string (as-blob "hello world")** returns "6865:6c6c:6f20:776f:726c:6400::"
The **ip6-string** function returns the string representation of a 16-byte IPv6 address *blob* in the form *"a*:*b*:*c*:*d*:*e*:*f*:*g*:*h*". The single argument *blob* must evaluate to a blob or be convertible into one. If the blob exceeds 16 bytes, the function uses only the first 16 to create the IPv6 address string. If the blob has fewer bytes, the function considers the right-most bytes as zero when it creates the IPv6 string.	
(**is-string** *expr*)	**(is-string 01:02:03:04)** returns null **(is-string "hello world")** returns "hello world" **(is-string 68:65:6c:6c:6f:20:77:6f:72:6c:64)** returns the blob
The **is-string** function returns the value of *expr*, if the result of evaluating *expr* is a string or can be used as a string, this function, otherwise it returns null. That is, if **as-string** does not return an error, then **is-string** returns the value of *expr.*	
(**length** *expr*)	**(length 1)** returns 4 **(length 01:02:03)** returns 3 **(length "hello world")** returns 11
The **length** function returns an integer whose value is the length in bytes of the value of *expr.* The argument *expr* can evaluate to any datatype. Integers always have length 4. The length of a string does not include any zero byte that may terminate the string.	
(**let** (*var1 ... varn*) *expr1 ... expn*)	**(let (x) (setq x (substring (request option "dhcp-class-identifier") 0 6)) (if (equali x "docsis") "client-class-1") (if (equali x "something else") "client-class-2"))**
The **let** function creates an environment with local variables *var1* through *varn*, which are initialized to a null value (you can give them other values by using the **setq** function). Once the local variables are initialized to null, the function evaluates expressions *expr1* through *exprn* in order. It then returns the value of its last expression, *exprn*. The benefit of this function is that you can use it to calculate a value once, assign it to a local variable, then reuse that value in other expressions without having to recalculate it. Variables are case sensitive.	

Table 25-1 Expression Functions (continued)

Function	Example
Description	
(list *oper1 ... opern***)**	**(list (create-prefix "cm-prefix" (create-prefix-range 32 0x1)) (create-prefix "cpe-address-prefix" (create-prefix-range 32 0x2)) (create-prefix "cpe-pd-prefix" (create-prefix-range 16 0x1)))**
The **list** function combines multiple operations, such as **create-v6-option** and **create-prefix-range** operations for DHCPv6. Nesting is not supported.	
(log *severity expr***)**	
The **log** function logs the result of converting *expr* to a string. The *severity* and *expr* must be a string and are converted to one if they do not evaluate to one. The *severity* can also be null; if a string, it must have one of these values: "debug" "activity" (the default if severity is null) "info" "warning" "error" **Note** Logging consumes considerable server resources, so limit the number of **log** function evaluations you put in an expression. Even if "error" severity is logged, the log function does not return an error. This only tags the log message with an error indication. See the **error** function to return an error as part of a function evaluation.	
(mapcharacters *string original-chars replacement-chars***)**	**(mapcharacters "Testing in progress" "Testing" "Work")** yields "Work in progress"
The **mapcharacters** function maps one set of characters to another. The function replaces each of the characters in *original-chars* found in *string* with the *replacement-chars* in the corresponding position in the string. If *replacement-chars* is shorter then *original-chars*, the replacement removes the extra characters. To remove all the *original-chars*, use a null string for *replacement-chars*. The processing of characters is from left to right, so that the mapping occurs at most once.	
(mask-blob *mask-size length***)**	**(mask-blob 1 4)** yields 80:00:00:00 **(mask-blob 4 2)** yields f0:00 **(mask-blob 31 4)** yields ff:ff:ff:fe
The **mask-blob** function returns a blob that contains the mask of length *mask-size* starting from the high-order bit of the blob, with a blob length of *length*. The *mask-size* is an expression that evaluates to an integer or must be convertible to one. Likewise the *length*, which cannot be smaller than the *mask-size*, but has no fixed limit except that it must be zero or positive. If *mask-size* is less than zero, it denotes a mask length calculated from the right end of the blob.	
(mask-int *mask-size***)**	**(mask-int 1)** yields 0x80000000 **(mask-int 4)** yields 0xf0000000 **(mask-int 31)** yields 0xfffffffe **(mask-int -1)** yields 0x00000001
The **mask-int** function returns an integer mask of length *mask-size* bits starting from the high-order bit of the integer. The *mask-size* is an expression that evaluates to an integer or must be convertible to one. Any number over 32 is meaningless and is treated as though a value of 32 was used. If *mask-size* is less than zero, it denotes a mask length calculated from the right end of the integer.	

Table 25-1 Expression Functions (continued)

Function	Example
Description	
(**not** *expr*)	`(not "hello world")` return null
The **not** function evaluates a string, blob, or integer expression to nonnull if it is null, and null if it is nonnull. The nonnull value returned when the value of *expr* is null is not guaranteed to remain the same over two calls.	
(**null** [*expr1 ... exprn*])	
The **null** function returns null and does not evaluate any of its arguments.	
(**or** *arg1 ... argn*) (**pick-first-value** *arg1 ... argn*)	`(or (request option 82 1) (request option 82 2) 01:02:03:04)` returns the value of sub-option 1 in option 82, and if that does not exist, returns the value of sub-option 2, and if that does not exist, returns 01:02:03:04
The **or** or **pick-first-value** function evaluates the arguments sequentially. When evaluating an *arg* returns a nonnull value, it returns the value of that first nonnull argument. Otherwise, it returns the value of the last argument, *argn*. The datatypes need not be the same.	
(**progn** *arg ... argn*)	`(progn (log (null) "I was here") (request option 82 1))`
The **progn** function evaluates arguments sequentially and returns the value of the last argument, *argn*.	
(**request** [**get** | **get-blob**] [**relay** [*n*]] **option** *opt* [{**enterprise-id** *n*} | {**vendor** *string*}] [**instance** *n*] [[*subopt* | {**option** *opt*}] [{**enterprise-id** *n*} | {**vendor** *string*}] [**instance** *n*] [[*sub-subopt* | {**option** *opt*}] [{**enterprise-id** *n*} | {**vendor** *string*}] [**instance** *n*]] [**instance-count** | {**index** *n*} | **count**])	`(request option 82)` returns the relay-agent-info option as a blob `(request option 82 1)` returns just the circuit-id (1) suboption `(request option 82 "circuit-id")` is the equivalent `(request option "domain-name-servers")` returns the first IP address from the domain-name-servers option `(request option 6 index 0)` is the equivalent `(request option 6 count)` returns the number of IP addresses `(request get-blob option "dhcp-class-identifier")` returns the value as a blob, not a string `(request option "IA-NA" instance 2 option "IAADDR" instance 3)` returns the third instance of the IA-NA option, and the fourth instance of the IAADDR option encapsulated in the IA-NA option `(request get-blob option "vendor-opts" enterprise-id 1234)` returns a blob of the option data for enterprise-id 1234 `(request option "vendor-opts" enterprise-id 1234 3)` returns suboption 3 from the requested vendor option data

Table 25-1 Expression Functions (continued)

Function	Example
Description	

The **request option** function returns the value of the option from the packet. The keywords are:

- **get**—Optional and assumed if omitted.
- **get-blob**—Returns the data as a blob, providing direct access to the option bytes.
- **relay**—Applies to IPv6 packets only, otherwise returns an error. Requests a relay option instead of a client option. The *n* indicates the *n*th closest relay agent to the client; if omitted, 0 (the relay agent nearest to the client) is assumed.
- **option**—Options are specified with the *opt* argument, which must evaluate to an integer or a string. If it does not evaluate to one of these, the function does not convert it and returns an error. Valid string values for the *opt* specifier are the same as those used for extensions.
- **enterprise-id**—After an option or suboption and for DHCPv4 and DHCPv6, returns only the data bytes after the given enterprise-id in the packet, instead of the entire data of an option.
- **vendor**—After an option or suboption, requests that the vendor custom option definition be used for decoding the data in the option. Does not apply to DHCPv6 options. Note that if no definition exists for the specified vendor string, no error is issued and the standard definition of an option is used (or, if none, it is assumed to be a blob).
- **instance**—Selects the (*n*+1)th instance of the preceding option or suboption. Instances start at 0. (You cannot use the instance and instance-count together in a single request function.)
- **instance-count**—Returns the number of instances of the preceding option or suboption, and is usually used to loop through all instances of it.
- **index**—Selects the (*n*+1)th value in an option that contains multiple values. For example, **index 0** returns the first value and **index 1** returns the second value.
- **count**—Returns the number of relevant data items in the preceding option, and is usually used with the **index** keyword to loop through all data values for an option or suboption.

The only string-valued suboption names defined for the *subopt* (suboption) specifier are for the relay-agent-info option (82) and are:

1—"circuit-id"
2—"remote-id"
4—"device-class"
5—"subnet-selection"
6—"subscriber-id"
7—"radius-attributes" (which includes the following encapsulated attributes that can be specified as subsuboptions: 1—"radius-user", 6—"radius-class", 88—"radius-framed-pool-name", 26—"radius-vendor-specific", 27—"radius-session-timeout", 100—“radius-framed-ipv6-pool”)
8—"authentication"
9—"v-i-vendor-class"
150—"cisco-subnet-selection"
151—"cisco-vpn-id"
152—"cisco-server-id-override"
181—"vpn-id"
182—"server-id-override"

Table 25-1 Expression Functions (continued)

Function	Example
Description	
The **request option** function returns a value with a datatype depending on the option requested. This shows how the datatypes in the table correspond to the datatypes returned by the **request** function: blob —> blob IP address —> 4-byte blob string —> string 8-bit unsigned integer —> uint 16-bit unsigned integer —> uint 32-bit unsigned integer —> uint integer —> sint byte-valued boolean —> sint=1 if true, null if false	
(request [**get** | **get-blob**] [**relay** [*number*]] *packetfield***)**	
	(request get ciaddr) returns the ciaddr if it exists, otherwise returns null **(request ciaddr)** is the same as **(request get ciaddr)** **(request giaddr)**
Valid values for *packetfield* are: **op** (blob 1) **htype** (blob 1) **hlen** (blob 1) **hops** (blob 1) **xid** (uint) **secs** (uint) **flags** (uint) **ciaddr** (blob 4) **yiaddr** (blob 4) **siaddr** (blob 4) **giaddr** (blob 4) **chaddr** (blob *hlen*) **sname** (string) **file** (string)	The **request** *packetfield* function returns the value of the named field from the request packet. DHCP request packets contain named fields as well as options in an option area. This form of the request function is used to retrieve specific named fields from the request packet. The **relay** keyword is described in the earlier, more general **request** function. The *packetfield* values defined in RFC 2131 are listed at the left. There are several *packetfield* values that can be requested which do not appear in exactly these ways in the raw DHCP packet. These take data that appears in the packet and combine it in commonly used ways. In these explanations, the packet contents assumed are: *hlen* = 1 *htype* = 6 *chaddr* = 01:02:03:04:05:06 **macaddress-string** (string)—Returns the MAC address in *hlen*,*htype*,*chaddr* format (for example, “1,6,01:02:03:04:05:06”) **macaddress-blob** (blob)—Returns the MAC address in *hlen*:*htype*:*chaddr* format (for example, 01:06:01:02:03:04:05:06) **macaddress-clientid** (blob)—Returns a client-id created from the MAC address in the Microsoft *htype*:*chaddr* client-id format (for example, 01:01:02:03:04:05:06)

Table 25-1 Expression Functions (continued)

Function Description	Example
Valid values for the DHCPv6 *packetfield* are: **msg-type** (uint) **msg-type-name** (string) **xid** (uint) **relay-count** (uint) **hop-count** (uint) **link-address** (blob 16) **peer-address** (blob 16)	The **msg-type** packet field for DHCPv6 describes the current relay or client message type, and has the values: 1=SOLICIT, 2=ADVERTISE, 3=REQUEST, 4=CONFIRM, 5=RENEW, 6=REBIND, 8=RELEASE, 9=-DECLINE, 11=INFORMATION-REQUEST, 12=RELAY-FORWARD The **msg-type-name** packet field returns a string of the message type name. The string value is always uppercase; for example, SOLICIT. The **xid** is the 24-bit client transaction ID, and the **relay-count** is the number of relay messages in the request. If a DHCPv6 packet field is requested from a DHCPv4 packet, an error is returned. The inverse is also true.
(request dump) The **request dump** function dumps the current request packet to the log file after the function evaluates the expression. Note that not all expression evaluations support the **dump** keyword, but when it is not supported, it is ignored.	
(search *arg1 arg2 fromend***)**	**(search "test" "this is a test")** returns 9 **(search "test" "this test test test" "true")** returns 15 The **search** function searches *arg1* for a subsequence in *arg2* that exactly matches. If found, it returns the index of the element in *arg2* where the subsequence begins (unless you set the *fromend* argument to "true" or some other arbitrary value); otherwise it returns null. (If *arg1* is null, it returns 0; if *arg2* is null, it returns null.) The function does an implicit **as-blob** conversion on both arguments. Thus, it compares the actual byte sequences of strings and blobs, and sints and uints become 4-byte blobs for the purpose of comparison. A nonnull *fromend* argument returns the index of the leftmost element of the rightmost matching subsequence.
(setq *var expr***)**	see the **let** function for examples The **setq** function sets *var* to the value of *expr.* You must precede it with the **let** function.
(sha256 *arg bits***)** The **sha256** function returns the specified *bits* from the SHA-256 encoded string of bytes in the *arg*.	

Table 25-1 Expression Functions (continued)

Function	Example
Description	
(starts-with *expr prefix-expr***)**	**(starts-with "abcdefghijklmnop" "abc")** returns "abcdefghijklmnop" **(starts-with "abcdefgji" "bcd")** returns null **(starts-with 01:02:03:04:05:06 01:02:03)** returns 01:02:03:04:05:06 **(starts-with "abcd" (as-string 61:62))** returns "abcd" **(starts-with "abcd" 61:62)** returns null **(starts-with "abcd" (to-string 61:62))** returns null
The **starts-with** function returns the value of *expr* if the *prefix-expr* value matches the beginning of *expr*, otherwise null. If *prefix-expr* is longer than *expr*, it returns null. The function returns an error if *prefix-expr* cannot be converted to the same datatype as *expr* (string or blob), or if *expr* evaluates to an integer. (See Table 25-2 on page 25-19.)	
(substring *expr offset len***)**	**(substring "abcdefg" 0 6)** returns bcdefg **(substring 01:02:03:04:05:06 3 2)** returns 04:05
The **substring** function returns *len* bytes of expression *expr*, starting at *offset*. The *expr* can be a string or blob; if an integer, converts to a blob. The result is a string or a blob, or null if any argument evaluates to null. If *offset* is greater than the length *len*, the result is null. If *offset* plus *len* is data beyond the end of *expr*, the function returns the rest of the data in *expr*. If *offset* is less than zero, the offset is from the end of the data (the last character is index –1, because –0=0, which references the first character). If this references data beyond the beginning of data, the offset is considered to be zero.	
(to-blob *expr***)**	**(to-blob 1)** returns 00:00:00:01 **(to-blob "01:02")** returns 01:02 **(to-blob 02:03)** returns 02:03 **(to-blob "this is not in blob format")** return an error
The **to-blob** function converts an expression to a blob. If *expr* evaluates to a string it must be in "*nn*:*nn*:*nn*" format. This function returns a blob that is the result of converting the string to a blob. If the function cannot convert the string to a blob, it returns an error. If *expr* evaluates to a blob, it returns that blob. If *expr* evaluates to an integer, it returns a four-byte blob representing the bytes of the integer in network order. (See Table 25-2 on page 25-19.)	
(to-lower *expr***)**	
The **to-lower** function takes a string and produces a lowercase string from it. When using the *client-lookup-id* attribute to calculate a client-specifier to look up a client-entry in the MCD local store (as opposed to LDAP), the resulting string must be lowercase. Use this function to easily make the result of the *client-lookup-id* a lowercase string. You may or may not want to use this function when accessing LDAP using the *client-lookup-id*.	

Table 25-1 Expression Functions (continued)

Function Description	Example
(to-sint *expr*)	**(to-sint "1")** returns 1 **(to-sint -1)** returns -1 **(to-sint 00:02)** returns 2 **(to-sint "00:02")** returns an error **(to-sint "4294967295")** returns an error
The **to-sint** function converts an expression to a signed integer. If *expr* evaluates to a string, it must be in a format that can be converted into a signed integer, else the function returns an error. If *expr* evaluates to a blob of one to four bytes, the function returns it as a signed integer. If *expr* evaluates to a blob of more than 4 bytes in length, it returns an error. If *expr* evaluates to an unsigned integer, it returns a signed integer with the same value, unless the value of the unsigned integer was greater than the largest positive signed integer, in which case it returns an error. If *expr* evaluates to a signed integer, it returns that value. (See Table 25-2 on page 25-19.)	
(to-uint *expr*)	**(to-uint "1")** returns 1 **(to-uint 00:02)** returns 2 **(to-uint "4294967295")** returns 4294967295 **(to-uint "00:02")** returns an error **(to-uint -1)** returns an error
The **to-uint** function converts an expression to an unsigned integer. If *expr* evaluates to a string, it must be in a format that can be converted into an unsigned integer, else the function returns an error. If *expr* evaluates to a blob of one to four bytes, it returns it as an unsigned integer. If *expr* evaluates to a blob of more than 4 bytes in length, it returns an error. If *expr* evaluates to a signed integer, it returns an unsigned integer with the same value, unless the value of the signed integer less than zero, in which case it returns an error. If *expr* evaluates to an unsigned integer, the function returns that value. (See Table 25-2 on page 25-19.)	
(to-string *expr*)	**(to-string "hello world")** returns "hello world" **(to-string -1)** returns "-1" **(to-string 02:04:06)** returns "02:04:06"
The **to-string** function converts an expression to a string. If *expr* evaluates to a string, it returns it; if a blob or integer, it returns its printable representation. It never returns an error if *expr* itself evaluates without error, because every value has a printable representation. (See Table 25-2 on page 25-19.)	
(try *expr failure-expr*)	**(try (try (***expr***) (***complex-failure-expr***)) "***string-constant***"** ensures that the outer try never returns an error (because evaluating "*string-constant*" cannot fail). **(try (error) 01:02:03)** always returns 01:02:03 **(try 1 01:02:03)** always returns 1 **(try (request option 82) "failure")** never returns "failure" because (request option 82) turns null if there is no option-82 in the packet and does not return an error **(try (request option "junk") "failure")** returns "failure" because "junk" is not a valid option-name.
The **try** function evaluates *expr* and returns the result of that evaluation if there were no errors encountered during the evaluation. If an error occurs while evaluating *expr* then: • If there is a *failure-expr* and it evaluates without error, it returns the result of that evaluation as the result of the **try** function. • If there is a *failure-expr* and the function encounters an error while evaluating *failure-expr*, it returns that error. • If there is no *failure-expr*, the **try** returns null.	

Table 25-1 Expression Functions (continued)

Function Description	Example
(**+** *arg1 ... argn*) (**–** *arg1 ... argn*) (***** *arg1 ... argn*) (**/** *arg1 ... argn*) (**%** *arg1 arg2*)	**(+ 1 2 3 4)** returns 10 **(- 10 5 2)** returns 3 **(* 3 4 5)** returns 60 **(/ 20 2 5)** returns 2 **(/ 20 0)** returns an error **(% 12 7)** returns 5 (12/7=1*7+5)

These functions do arithmetic operations on a signed integer or an expression convertible to a signed integer. Any argument that cannot convert to a signed integer (and is not null) returns an error. Any argument that evaluates to null is ignored (except that the first argument for – and / must not evaluate to null). These functions always return signed integers (note that overflow and underflow are currently not caught):

- **+** sums the arguments; if no arguments, the result is 0.
- **–** negates the value of a single argument or, if multiple arguments, successively subtracts the values of the remaining ones from the first one; for example, **(– 3 4 5)** becomes –6.
- ***** takes the product of the argument values; if no arguments, the result is 1.
- **/** successively divides the first argument by all of the others; for example, **(/ 100 4 5)** becomes 5. If any argument other than the first equals 0, an error is returned.
- **%** is the modulo arithmetic operator to determine the remainder of the result of the first argument divided by the second one; for example, **(% 12 7)** becomes 5 (12 / 7 = 1 * 7 + 5).

Datatype Conversions

When a function needs an argument of a particular datatype, it tries to convert a value into that datatype. Sometimes this can fail, often causing the entire function to fail. Datatype conversion is also performed by the **to-string**, **to-blob**, **to-sint**, and **to-uint** functions. Whenever a function needs an argument in a specific datatype, it calls the internal version of these externally available functions.

There are also **as-string**, **as-blob**, **as-sint**, and **as-uint** conversion functions, where the data in a value are simply relabeled as the desired datatype, although some checking does go on. The conversion matrix for both function sets appears in Table 25-2.

Table 25-2 Datatype Conversion Functions

Argument Type: Function:	String	Blob	Signed Integer	Unsigned Integer
to-string	—	Cannot fail	Cannot fail	Cannot fail
as-string	—	Relabels as string bytes, if printable characters	Converts to a 4-byte blob, then processes it as a blob (which fails except for a few special integers)	Converts to a 4-byte blob, then processes as a blob (which fails except for a few special integers)
to-blob	Must be in the form “01:02:03”	—	Cannot fail; produces a 4-byte blob from the 4 bytes of the integer.	Cannot fail; produces a 4-byte blob from the 4 bytes of the integer.

Table 25-2 Datatype Conversion Functions (continued)

Argument Type: Function:	String	Blob	Signed Integer	Unsigned Integer
as-blob	Cannot fail; relabels ASCII characters as blob bytes.	—	Cannot fail; produces 4-byte blob from the 4 bytes of the integer.	Cannot fail; produces 4-byte blob from the 4 bytes of the integer.
to-sint	Must be in the form *n* or *–n*.	1-, 2-, 3-, or 4-byte blobs only.	—	Converts only if it is not too big to fit into a signed integer.
as-sint	Not usually useful; converts a 1-, 2-, 3-, or 4-byte string to a blob and then packs it up into a signed integer.	Not usually useful; converts only 1-, 2-, 3-, or 4-byte blobs.	—	Cannot fail; converts to a signed integer, negative if a larger unsigned integer would fit into a positive signed integer.
to-uint	Must be in the form *n*.	1-, 2-, 3-, or 4-byte blobs only.	Nonnegative only.	—
as-uint	Not usually useful; converts a 1-, 2-, 3-, or 4-byte string to a blob and then a signed integer.	Not usually useful; converts only 1-, 2-, 3-, or 4-byte blobs.	Cannot fail; converts to an unsigned integer, and a negative signed integer becomes a large unsigned integer.	—

Expressions in the CLI

You must include the expression in a text file if you want to include it with a CLI attribute setting. The default path of this file is the current working directory. Do not enclose the expression in quotes. You can add comment lines prefixed by **#**, **//**, or **;**. For example:

```
// Expression to set client-class based on remote-id
(if (equal (request option "relay-agent-info" "remote-id") (request chaddr))
           "no-limit" "limit")
```

The server reads the file when processing the command. An example of a command to include this file is:

```
nrcmd> dhcp set client-class-lookup-id=@expressionfile1.txt
```

Note You must set the attribute to a file in the CLI. You cannot enter the expression directly in the equation.

Expression Examples

These examples provide the maximum support for option 82 processing. They set up clients to limit, those not to limit, and those that exceed configuration limits and should be assigned to an over-limit client-class. There are separate scopes and scope-selection tags for each of the three classes of clients:

- **Client-classes**—limit, no-limit, and over-limit.
- **Scopes**—10.0.1.0 (primary), 10.0.2.0 and 10.0.3.0 (secondaries), named for their subnets.
- **Scope-selection tags**—limit-tag, no-limit-tag, and over-limit-tag. The scopes are named for the address pools that they represent. The selection tags are allocated to the scopes with 10.0.1.0 getting limit-tag, 10.0.2.0 getting no-limit-tag, and 10.0.3.0 getting over-limit-tag.

See Also

Limitation Example 1: DOCSIS Cable Modem
Limitation Example 2: Extended DOCSIS Cable Modem, page 25-22
Limitation Example 3: DSL Over Asynchronous Transfer Mode, page 25-23

Limitation Example 1: DOCSIS Cable Modem

The test is to determine whether the device is considered a DOCSIS cable modem, and limit the number of customer devices behind every cable modem. The limitation ID for the limit client-class is the cable modem MAC address, included in the *remote-id* suboption of the *relay-agent-info* option.

The expression for the *client-class-lookup-id* attribute on the server is:

```
// Expression to set client-class to no-limit or limit based on remote-id
(if (equal (request option "relay-agent-info" "remote-id")
    (request chaddr))
    "no-limit"
    "limit")
```

The above expression indicates that if the contents of the *remote-id* suboption (2) of the *relay-agent-info* option is the same as the *chaddr* of the packet, then the client-class is *no-limit*, otherwise *limit*.

The *limitation-id* expression for the *limit* client-class is:

```
(request option "relay-agent-info" "remote-id")
```

Use this expression in the following steps:

Step 1 Define the scope-selection tags. (You do not need to do this in Network Registrar 6.0 or 6.1.)

Step 2 Define the client-classes.

Step 3 Define the scopes, their ranges and tags, and if they are primary or secondary. Note the host range for each scope, which is less likely to be misread than if they all have the same host number.

Step 4 Define the limitation count. It can go in the *default* policy; if the request does not show a limitation ID, the count is not checked.

Step 5 Add an expression in an expression file, cclookup1.txt, for the purpose:

```
// Expression to set limitation count based on remote-id
(if (equal (request option "relay-agent-info" "remote-id")
           (request chaddr)) "no-limit" "limit")
```

Step 6 Refer to the expression file when setting the *client-class lookup-id* attribute on the server level.

Step 7 Add another expression for the limitation ID for the client in a cclimit1.txt file:

```
// Expression to set limitation ID based on remote-id
(request option "relay-agent-info" "remote-id")
```

Step 8 Refer to this expression file when setting the *limitation-id* attribute for the client-class.

Step 9 If in staged scope edit mode, reload the server.

The result of doing this for a previously unused configuration would be to put the first two DHCP clients with a common *remote-id* option 82 suboption value in the *limit* client-class. The third client with the same value would go in the *over-limit* client-class. There are no limits to the number of devices a subscriber can have in the *no-limit* client-class, because it has no configured limitation ID. Any device with a MAC address equal to the value of the *remote-id* suboption is ignored for the purposes of limitation, and goes in the *no-limit* client class, for which there is no limitation ID configured.

Limitation Example 2: Extended DOCSIS Cable Modem

This example is an extension to the example described in the "Limitation Example 1: DOCSIS Cable Modem" section on page 25-21. In the latter example, all of the cable modems allowed only two client devices beyond them, since a limitation count of two was defined for the *default* policy. In this example, specific cable-modems are configured to allow a different number of devices to be granted IP addresses from the scopes that use the *limit-tag* scope-selection tag.

In this case, you need to explicitly configure any cable modem with more than two addresses behind it in the client-class database. This requires enabling client-class processing server-wide, so that you can look up the client entry for a cable modem in the Network Registrar or LDAP database. Not finding the cable modem limits the number of devices to two; finding it uses the limitation count from the policy configured for the cable modem.

This example requires just one additional policy, *five*, which allows five devices.

Step 1 Enable client-class processing server-wide.

Step 2 Create the *five* policy with a limitation count of five devices.

Step 3 As in the previous example, use an expression to set a limitation ID for the *limit* client-class. Put the limitation ID in a cclimit2.txt file, and the lookup ID in a cclookup2.txt file:

```
cclimit2.txt file:
// Expression to set limitation ID
(request option "relay-agent-info" "remote-id")

cclookup2.txt file:
// Expression to set client-class lookup ID
(concat "1,6," (to-string (request option "relay-agent-info" "remote-id")))
```

Step 4 Refer to these files when setting the appropriate attributes.

Step 5 Define some cable modem clients and apply the *five* policy to them.

Step 6 If in staged scope edit mode, reload the server.

Limitation Example 3: DSL Over Asynchronous Transfer Mode

This example shows how to use expressions to configure Digital Subscriber Line (DSL) access for a subscriber to a service provider using asynchronous transfer mode (ATM) routed bridge encapsulation (RBE). Service providers are increasingly using ATM RBE to configure a DSL subscriber. The DHCP Option 82 support for routed bridge encapsulation feature as of Cisco IOS Release 12.2(2)T enables those service providers to use DHCP to assign IP addresses and option 82 to implement security and IP address assignment policies.

In this scenario, DSL subscribers are identified as individual ATM subinterfaces on a Cisco 7401ASR router. Each customer has their own subinterface in the router and each subinterface has its own virtual channel identifier (VCI) and virtual path identifier (VPI) to identify the next destination of an ATM cell as it passes through ATM switches. The 7401ASR router routes up to a Cisco 7206 gateway router.

Step 1 Set up the DHCP server and interfaces for the router using IOS. This is a typical IOS configuration:

```
Router#ip dhcp-server 170.16.1.2
Router#interface Loopback0
Loopback0(config)#ip address 11.1.1.129 255.255.255.192
Loopback0(config)#exit
Router#interface ATM4/0
ATM4/0(config)#no ip address
ATM4/0(config)#exit
Router#interface ATM4/0.1 point-to-point
ATM4/0.1(config)#ip unnumbered Loopback0
ATM4/0.1(config)#ip helper-address 170.16.1.2
ATM4/0.1(config)#atm route-bridged ip
ATM4/0.1(config)#pvc 88/800
ATM4/0.1(config)#encapsulation aal5snap
ATM4/0.1(config)#exit
Router#interface Ethernet5/1
Ethernet5/1(config)#ip address 170.16.1.1 255.255.0.0
Ethernet5/1(config)#exit
Router#router eigrp 100
eigrp(config)#network 11.0.0.0
eigrp(config)#network 170.16.0.0
eigrp(config)#exit
```

Step 2 In IOS, enable the system to insert the DHCP option 82 data in forwarded BOOTREQUEST messages to a Cisco IOS DHCP server:

```
Router#ip dhcp relay information option
```

Step 3 In IOS, specify the IP address of the loopback interface on the DHCP relay agent that is sent to the DHCP server using the option 82 *remote-id* suboption (2):

```
Router#rbe nasip Loopback0
```

Step 4 In Network Registrar, enable client-class processing server-wide.

Step 5 Create the *one* policy with a limitation count of one device.

Step 6 Put the packets in the right client-class. All the packets should be in the *limit* client-class. Create a lookup file containing just the value *limit*, then set the client-class lookup ID. In the cclookup3.txt file:

```
// Sets client-class to limit
"limit"
```

Step 7 Use an expression to ensure that those packets that are limited have the right limitation ID. Put the expression in a file and refer to that file to set the limitation ID. The *substring* function gets the VPI/VCI by extracting bytes 10 through 12 of the option 82 suboption 2 (*remote-id*) data field. In the cclimit3.txt file:

```
// Sets limitation ID
(substring (request option 82 2) 9 3)
```

Step 8 If in staged scope edit mode, reload the server.

Debugging Expressions

If you are having trouble with expressions, examine the DHCP log file at server startup. The expression is printed in such a way as to clarify the nesting of functions, and can help in confirming your intentions. Pay special attention to the **equal** function and any datatype conversions of arguments. If the arguments are not the same datatype, they are converted to strings using code similar to the **to-string** function.

You can set various debug levels for expressions by using the *expression-trace-level* attribute for the DHCP server. All executed expressions are traced to the degree set by the attribute. The highest trace level is 10. If you set the level to at least 2, any nonworking expression is retried again at level 10.

The trace levels for *expression-trace-level* are (use the number value):

- **0**—No tracing
- **1**—Failures, including those protected by **(try** ...**)**
- **2**—Total failure retries (with trace level = 6 for retry)
- **3**—Function calls and returns
- **4**—Function arguments evaluated
- **5**—Print function arguments
- **6**—Datatype conversions (everything)

The trace levels for *expression-configuration-trace-level* are (use the number value):

- **0**—No additional tracing
- **1**—No additional tracing
- **2**—Failure retry (the default)
- **3**—Function definitions
- **4**—Function arguments
- **5**—Variable lookups and literal details
- **6**—Everything

To trace expressions you have trouble configuring, there is also an *expression-configuration-trace-level* attribute that you can set to any level from 1 through 10. If you set the level to at least a 2, any expression that does not configure is retried again with the level set to 6. Gaps in the numbering are to accommodate future level additions.



CHAPTER 26

Managing DHCPv6 Addresses

Network Registrar supports the following IPv6 addressing for DHCP (DHCPv6):

- **Stateless autoconfiguration (RFC 3736)**—The DHCPv6 server does not assign addresses, but instead provides configuration parameters, such as DNS server data, to clients.
- **Stateful autoconfiguration (RFC 3315)**—The DHCPv6 server assigns nontemporary or temporary addresses and provides configuration parameters to clients.
- **Prefix Delegation (RFC 3633)**—The DHCPv6 server delegates prefixes to clients (routers).

The DHCPv6 service provides these capabilities:

- **Links and prefixes**—Similar to DHCPv4 networks and scopes that define the network topology. Each link can have one or more prefixes.
- **Policies and options**—You can assign attributes and options to links, prefixes, and clients.
- **VPN support**—Provides multiple address spaces (virtual private networks).
- **Client-classing**—You can classify clients and select prefixes based on known clients or packet-based expressions.
- **Static reservations**—Clients can receive predetermined addresses.
- **Extensions**—Extend the DHCP server processing by using C/C++ and Tcl extensions.
- **DNS Updates**—DNS server updates of DHCP activity (over IPv4).
- **SNMP traps**—Generate traps for events, such as if the number of leases in a prefix exceeds a certain limit (or drops below a certain limit) or if the server detects duplicate addresses.
- **Statistics collection and logging**—Provides server activity monitoring.

The DHCPv6 service requires that the server operating system support IPv6 and that you configure at least one interface on the system for IPv6.

See Also

DHCPv6 Concepts, page 26-2
DHCPv6 Configuration, page 26-8
DNS Update for DHCPv6, page 26-29

DHCPv6 Concepts

The following subsections describe the concepts related to DHCPv6 operation:

- IPv6 Addressing, page 26-2
- Links and Prefixes, page 26-2
- DHCPv6 Clients and Leases, page 26-4
- DHCPv6 Policy Hierarchy, page 26-7
- DHCPv6 Options, page 26-8

IPv6 Addressing

IPv6 addresses are 128 bits long and are represented as a series of 16-bit hexadecimal fields separated by colons (**:**). The A, B, C, D, E, and F in hexadecimal are case insensitive. For example:

```
2001:db8:0000:0000:0000:0000:0000:0000
```

A few shortcuts to this addressing are:

- Leading zeros in a field are optional, so that you can write 09c0 as 9c0, and 0000 as 0.
- You can represent successive fields of zeros (any number of them) by a double colon (::), but only once in an address (because, if used more than once, the address parser has no way of identifying the size of each block of zeros). This reduces the length of addresses; for example, 2001:db8:0000:0000:0000:0000:0000:0000 can be written:

  ```
  2001:db8::
  ```

Link-local addresses have a scope limited to the link, and use the prefix fe80::/10. Loopback addresses have the address ::1. Multicast addresses have the prefix ff00::/8 (there are no broadcast addresses in IPv6).

The IPv4-compatible addresses in IPv6 are the IPv4 decimal quad addresses prefixed by **::**. For example, you can write the IPv4 address interpreted as **::c0a8:1e01** in the form **::192.168.30.1**.

Links and Prefixes

The explicit DHCPv6 configuration objects are links and prefixes:

- **Link**—Network segment that can have one or more prefixes, and adds an additional layer at which policies can be applied for DHCPv6 clients.
- **Prefix**—Equates to a scope in IPv4. The link associated with a prefix is similar to a primary scope, except that it names a link and not another prefix.

Just as with scopes, you can create multiple prefix objects for the same IPv6 prefix. However, rather than supporting multiple ranges with explicit start and end addresses, prefixes support only a single range that must be an IPv6 prefix with a length the same as, or longer than, the prefix object. For example, if you define a 2001::/64 prefix with a 2001::/96 range, the server can assign addresses from 2001:0:0:0:0:0:0:0 through 2001:0:0:0:0:0:ffff:ffff only. The range:

- Is limited to powers of 2.
- Must be unique (cannot be duplicated by any other range, except in a different VPN).
- Cannot be contained in, or contain, another range.

If you omit a range, the server uses the full prefix as the range.

You create a link only if more than one prefix object with a different IPv6 prefix exists on a link. When the server loads the configuration, if a prefix has no explicit link, the server searches for or creates an implicit link with the name Link-[*vpn.name*/]*prefix*. All prefix objects with the same IPv6 prefix must either not specify a link or explicitly specify the same link.

The DHCPv6-enabled server supports VPNs (namespaces) for DHCPv6. However there is presently no means to make use of anything other than the default global VPN (there is no VPN option). Both the link and prefix objects have a *vpn-id* attribute, because prefixes do not require links, but all prefixes on a link must use the same VPN ID.

See Also

Determining Links and Prefixes
Generating Addresses, page 26-3
Generating Delegated Prefixes, page 26-4

Determining Links and Prefixes

When the DHCPv6 server receives a DHCPv6 message, it determines the links and prefixes it uses to service the request. The server:

1. Finds the source address:
   a. If the client message was relayed, the server sets the source address to the first nonzero link-address field starting with the Relay-Forward message closest to the client (working outwards). If the server finds a source address, it proceeds to step 2.
   b. Otherwise, if the message source address is a link-local address, the server sets the source address to the first address for the interface on which it received the message for which a prefix exists (or 0 if it finds no prefix for any address). It then proceeds to step 2.
   c. Otherwise, the server sets the source address to the message source address.
2. Locates the prefix for the source address. If the server cannot find a prefix for the source address, it cannot service the client and drops the request.
3. Locates the link for the prefix. This always exists and is either an explicitly configured link or the implicitly created link based on the prefix address.

Now that the server can determine the client link, it can process the client request. Depending on whether the client request is stateful or prefix-delegated, and on the selection criteria and other factors, the server might use one or more prefixes for the link to service the client request.

This is one area of difference between DHCPv4 and DHCPv6. In DHCPv4, the server selects only one of the scopes from the network to service the client request. In DHCPv6, the server can use all the prefixes for the link. Thus, the server might assign a client an address, or delegate a prefix, from multiple prefixes for the link (subject to selection criteria and other conditions).

Generating Addresses

IPv6 addresses are 128-bit addresses (as compared to 32-bit addresses for IPv4). In most cases, DHCPv6 servers assign 64 of those bits, the interface-identifier (EUI-64) portion (see RFC 4291). You can generate addresses by using the client 64-bit interface-identifier or a random number generator. The interface-identifier emulates how stateless autoconfiguration assigns addresses to clients. Unfortunately, there are privacy concerns regarding its use, and it is limited to one address per prefix for the client.

By default, Network Registrar generates an address using an algorithm similar to that described in RFC 4941 to generate a random interface identifier. These random interface identifiers have a zero value for the universal/local bit to distinguish them from EUI-64-based identifiers. The server also skips randomly generated interface identifiers from ::0 to ::ff so that you can use identifiers for infrastructure devices (such as routers). You can configure whether to assign the interface-identifier (if available) first for each prefix (through the interface-identifier flag of the prefix *allocation-algorithm*s attribute). (See the "Creating and Editing Prefixes" section on page 26-18.) If you specify use of the interface-identifier, the server might still use randomly generated addresses if the address is not available to the client, or the client requests multiple addresses on a prefix.

The server generates addresses based on the prefix-configured range (or the prefix address if there is no range). If the range prefix length is shorter than 64, the server supplies only 64 bits and places them in the address interface-identifier field. If the prefix length is longer than 64, the server supplies only the remaining bits of the address. Thus, a /96 range uses 96 bits from the specified range followed by 32 bits of either the client interface-identifier or a randomly generated value. If the resulting address is not available (such as if it is already leased to another client, or to the same client, but on a different binding), the server tries to generate another address. It repeats this process up to at most 500 times.

Note The DHCP server tests only the randomly generated interface identifier for values from ::0 to ::ff, not the resulting address. Thus, a randomly generated address may end up using an *xxxx*:*xxxx*:*xxxx*:*xxxx*::0 through *xxxx*:*xxxx*:*xxxx*:*xxxx*::ff address if the length of the prefix is longer than /64 and the prefix bits that extend beyond the /64 boundary are all zero.

Tip You can also choose from additional address generation algorithms for a prefix and prefix template; see the "Creating and Editing Prefix Templates" section on page 26-14.

Generating Delegated Prefixes

The DHCPv6 server uses a first-fit algorithm when generating delegated prefixes. The server allocates the first available prefix of the length configured or requested.

DHCPv6 Clients and Leases

The DCHPv6 server supports clients and leases that are similar to those for DHCPv4. The key differences are:

- The server identifies DHCPv6 clients by their DHCP Unique Identifier (DUID), which is the DHCPv4 concept of hardware addresses and client IDs consolidated into one unique client identifier.
- DHCPv6 clients can have multiple leases. This means that if multiple prefixes are on a single link, the server assigns the client an address from each prefix that it is allowed to use, not just from one scope, as in DHCPv4.
- The server first creates a DHCPv6 client when it associates the first lease with it, and deletes the client when it no longer has any leases associated with it. This is identical to DHCPv4 behavior, except that a DHCPv4 client can only have a single lease.
- DHCPv6 leases are dynamically created. The server does not create all leases that it can potentially use at configuration time, because there potentially could be billions of these leases.

Leases can be for:

- **Nontemporary addresses**—Standard IPv6 unicast addresses with likely long (and renewable) lifetimes.
- **Temporary addresses**—Standard IPv6 unicast addresses, but with very limited (and nonrenewable) lifetimes. Temporary addresses solve a privacy issue with IPv6 (see RFC 3041).
- **Delegated prefixes**—Used for prefix delegation (see RFC 3633).

Leases have both a preferred and valid lifetime:

- **Preferred lifetime**—Primarily for the use of the client, the length of time that a valid address is preferred. When the preferred lifetime expires, the address becomes deprecated.
- **Valid lifetime**—Used by both client and server, it is the length of time an address remains in the valid state. The valid lifetime must be greater then or equal to the preferred lifetime. When the valid lifetime expires, the address becomes invalid. A lease is eligible to be deleted once the valid lifetime expires. This is essentially the same as the DHCPv4 lease time.

See Also

DHCPv6 Bindings
Lease Affinity
Lease Life Cycle, page 26-6
DHCPv6 Reservations, page 26-7
Searching for Leases, page 26-7
Querying Leases for DHCPv6, page 26-7

DHCPv6 Bindings

Bindings are new to DHCPv6 and allow multiple groups of addresses to be allocated to a client. A client binding consists of one of three types:

- Nontemporary (IA_NA)
- Temporary (IA_TA)
- Prefix delegation (IA_PD)

A binding also consists of a unique Identity Association Identifier (IAID). Leases always exist under a binding. Clients, therefore, have one or more bindings, and bindings have one or more leases. The server creates bindings when it first adds the lease, and removes the binding when it has no more leases. The server creates clients when adding the first binding, and removes them when it has no more bindings.

Lease Affinity

For DHCPv4, when a lease expires or the server releases it, the server remembers the client for an address as long as it is not assigned to another client. For DHCPv6, because of the large IPv6 address space and depending on the address generation technique, eons could pass before an address needs reassignment to another client. Therefore, Network Registrar provides an *affinity-period* attribute so that the client can get the same address even if not requesting a renewal before expiration.

The affinity period is desirable in some environments, but not in others where the affinity time would be zero or very small. During the affinity period, the lease is in the AVAILABLE state and still associated with the client that last leased it. If the client requests a lease during this period, the server grants it the same lease (or, if renewals are inhibited, the client explicitly does not get that lease).

Lease Life Cycle

Leases have a life cycle controlled by states. A lease only exists while it is associated with a client and the server deletes it once it is no longer associated with that client. The life cycle and state transitions are:

1. A lease is born and associated with an address when the server:
   a. Creates a reservation for a lease, which puts the lease in the AVAILABLE state and marks it as RESERVED. No timer is associated with this state and the server does not delete the lease as long as it is RESERVED.
   b. Sends an ADVERTISE message to a client, which puts the lease in OFFERED state. The lease transitions to DELETED state after the offer timeout.
   c. Sends a REPLY message to a client (for a REQUEST, RENEW, or REBIND), which puts the lease in LEASED state. The lease transitions to EXPIRED state after the valid lifetime for the lease elapses.
2. An OFFERED lease transitions to:
   a. LEASED state when the server receives a REQUEST message, and then transitions to EXPIRED state after the valid lifetime for the lease elapses.
   b. DELETED state if the offered-time expires.
3. A LEASED lease:
   a. Is renewed when the server receives a REQUEST, RENEW, or REBIND message. The lease transitions to EXPIRED state after the new valid lifetime for the lease elapses (note that the new valid lifetime could be 0).
   b. Transitions to RELEASED state when the server receives a RELEASE message. The lease transitions to AVAILABLE state after the release-grace-period elapses.
   c. Transitions to UNAVAILABLE state when the server receives a DECLINE message. The server deletes the lease after the unavailable timeout period elapses.
4. An EXPIRED lease transitions to AVAILABLE state after the grace-period. The server deletes the lease after the affinity-period elapses.
5. An AVAILABLE lease:
   a. Transitions to DELETED state and the server deletes it from memory and the lease database after the affinity-period elapses.
   b. Cannot be deleted if it is RESERVED, and it remains AVAILABLE.
6. The server can reoffer a LEASED, EXPIRED, RELEASED, or AVAILABLE lease to a client, but it remains in its current state, although the server extends the timeout to at least the *offer-timeout*.
7. A LEASED lease can also transition to REVOKED state if the server needs to revoke the lease. A revoked lease was previously valid but became invalid because of configuration or selection tag changes. The server can revoke a lease when the client attempts to renew, if the lease is reserved for a different client or the prefix is no longer usable. The lease transitions to AVAILABLE again only after its valid lifetime expires or the client sends a SOLICIT for a new lease.

DHCPv6 Reservations

Reservations apply to nontemporary addresses and delegated prefixes only. They are stored under the prefix in the configuration and must always be for an address (or prefix) under the prefix. The reservation can be outside the prefix object range, provided that it is not in another prefix object range. This restriction has implications when adding new prefix objects, because existing reservations could violate this rule.

Searching for Leases

Network Registrar 7.0 introduces a new mechanism to search for leases in the configured DHCPv6 network. For details, see the "Searching Server-Wide for Leases" section on page 22-9.

Querying Leases for DHCPv6

For details on the DHCPLEASEQUERY implementation for DHCPv6, see the "Leasequery for DHCPv6" section on page 22-31.

DHCPv6 Policy Hierarchy

DHCPv6 uses the existing policy objects, with additional DHCPv6 specific attributes (that are mostly analogous to those in DHCPv4). For DHCPv6, the hierarchy is:

1. Client embedded policy
2. Client named policy
3. Client-class embedded policy
4. Client-class named policy
5. Prefix embedded policy
6. Prefix named policy
7. Link embedded policy
8. Link named policy
9. system_default_policy

For attributes, the default value for the most local policy applies. This hierarchy is the same as for DHCPv4, except for the additional link policies and the fact that the prefix policies replace the scope policies. (For a comparison with the DHCPv4 policy hierarchy, see the "Policy Hierarchy" section on page 21-3.)

The hierarchy applies to most policy attributes, which the server processes in the context of a single prefix. However, the server processes a few attributes (specifically *allow-rapid-commit*, *reconfigure*, *v6-reply-option*, *v6-options*, and *v6-vendor-options*) in the context of multiple prefixes. In these cases, the processing at the prefix levels (steps 5 and 6) is a bit different:

- For the *reconfigure* attribute that controls whether the server requires, allows, or disallows client reconfiguration, the server checks the embedded and named policies of all prefixes on the link that the client is allowed to use (based on selection tags). If any of the prefix policies have the *reconfigure* attribute set to **disallow** or **require**, the server uses that setting. Otherwise, if at least one policy has it set to **allow**, Reconfigure is allowed. Otherwise, the server checks the remaining policies in the hierarchy. (See the "Reconfigure Support" section on page 26-27 for details.)

- If the client requests Rapid Commit (see the "Editing DHCPv6 Server Attributes" section on page 26-25), the server checks the embedded and named policies of all prefixes on the link that the client is allowed to use (based on selection tags). If one of these policies has *allow-rapid-commit* disabled, the server processes the client request as if Rapid Commit were not part of the request. If at least one policy has *allow-rapid-commit* enabled, the client can use Rapid Commit. If no prefix policy has the attribute configured, processing continues at step 7.
- For the options-related attributes (see the "Setting DHCPv6 Options" section on page 26-27), the server also does special handling at steps 5 and 6. The server checks the embedded and then named policy of each prefix on the link. It then uses the first one with the configured *v6-reply-option* attribute, or the first one with the configured value for the *v6-options* or *v6-vendor-options*.
- The server checks the prefixes in case-insensitive alphabetical order.



Tip In configurations with multiple prefixes on a link, avoid setting the Rapid Commit and option properties for the prefix policy, but rather set them on the link policy or other policy instead.

DHCPv6 Options

DHCPv6 options do not use DHCPv4 options; they are unique and separate. There are currently about 46 DHCPv6 options (see Table B-8 on page B-11). Most of these options are the DHCPv6 protocol infrastructure options and are not user-definable. They use a 16-bit option code and 16-bit length (DHCPv4 uses only 8 bits for both of these). Configuring options and the behavior of configured options in policies are similar to those for DHCPv4. See the "Setting DHCPv6 Options" section on page 26-27 for details about client processing as it relates to the policy hierarchy.

DHCPv6 Configuration

The following sections describe how to configure DHCPv6 in Network Registrar:

- Viewing IPv6 Address Space, page 26-9
- Configuring Links, page 26-9
- Configuring Prefixes, page 26-14
- Viewing DHCPv6 Networks, page 26-24
- Editing DHCPv6 Server Attributes, page 26-25
- Configuring DHCPv6 Policies, page 26-25
- Configuring DHCPv6 Client-Classes, page 26-26
- Configuring DHCPv6 Clients, page 26-26
- Setting DHCPv6 Options, page 26-27
- Reconfigure Support, page 26-27

Viewing IPv6 Address Space

When you click **Address Space v6**, then **Address Space** in the local advanced or regional web UI, you open the View Unified v6 Address Space page. This page is like the View Unified Address Space page for IPv4 (see the "Viewing Address Space" section on page 9-2). On the View Unified v6 Address Space page you can:

- Set a VPN for the address space.
- Add a prefix by adding its name and address and choosing a DHCP type and possible template. Click **Add Prefix** to open the Add Prefix page (see the "Creating and Editing Prefixes" section on page 26-18).
- Edit a prefix by clicking its name. This opens the Edit Prefix page (see the "Creating and Editing Prefixes" section on page 26-18).
- View the current usage of the prefix space (see the "Viewing Address Utilization for Prefixes" section on page 26-22).

Configuring Links

You can configure DHCPv6 links directly, or you can create link templates for them first. See the following subsections:

- Creating and Editing Link Templates
- Creating and Editing Links, page 26-12

Creating and Editing Link Templates

You can create links from predefined templates. The attributes you can set for a link template are as follows (for the expression syntax, see the "Using Expressions in Link Templates" section on page 26-11):

- ***name***—User-assigned name for the link template.
- ***description***—Description of the link template itself.
- ***policy***—Shared policy used when replying to clients, as applied to the link.
- ***link-name-expr***—Expression to define the name of the link once the template is applied.
- ***link-description-expr***—Expression to define the description on the link once applied.
- ***prefix-expr***—Expression to create the list of associated prefixes once the template is applied. For example, you can specify creating prefixes based on defining *prefix-expr* as **@link-prefix-expr.txt** to point to the file that contains this expression (and assuming that the cm-prefix, cpe-address-prefix, and cpe-pd-prefix templates exist):

```
(list
(create-prefix "cm-prefix" (create-prefix-range 32 0x1))
(create-prefix "cpe-address-prefix" (create-prefix-range 32 0x2))
(create-prefix "cpe-pd-prefix" (create-prefix-range 16 0x1))
)
```

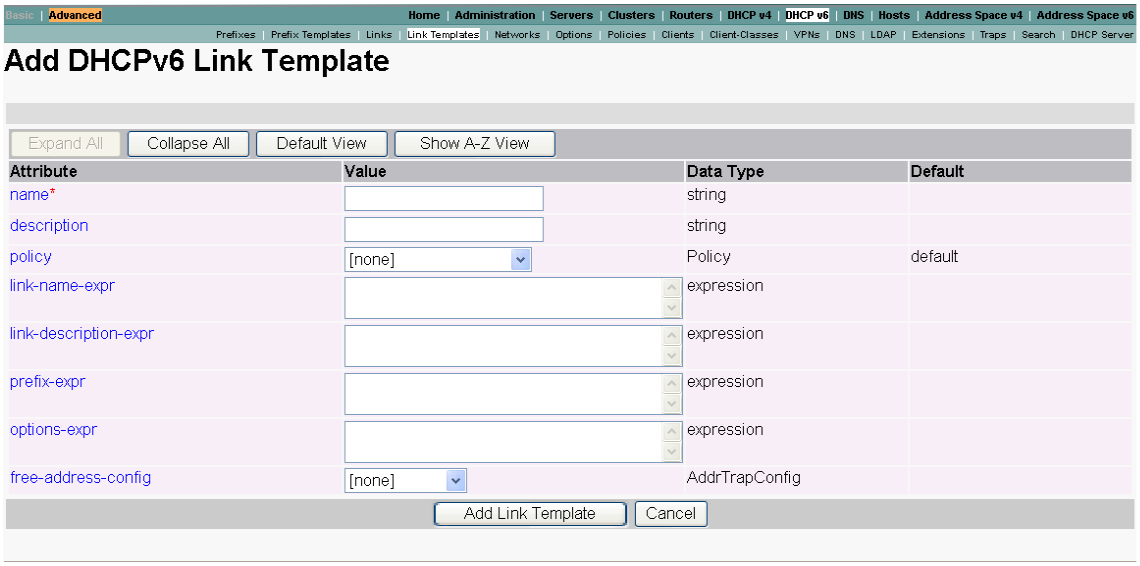
- ***options-expr***—Expression to define the list of embedded policy options to create with the link.

Local Advanced and Regional Web UI

Step 1 Click **DHCP v6**, then **Link Templates**. The List DHCPv6 Link Templates page shows the existing templates.

Step 2 Click **Add Link Template** to open the Add DHCPv6 Link Template page (see Figure 26-1 for the local version of the page).

Figure 26-1 Add DHCPv6 Link Template Page (Local Advanced)



Step 3 Enter a link template name, optional description, and optionally choose a preconfigured policy from the drop-down list.

Step 4 Add expressions for the *link-name-expr*, *link-description-expr*, *prefix-expr*, or *options-expr* field attributes (see the "Using Expressions in Link Templates" section on page 26-11).

Step 5 Click **Add Link Template**.

Step 6 In the regional web UI, you can pull replica link templates or push templates to local clusters:

- Click **Pull Replica Link Template** to open the Select DHCPv6 Link Template Data to Pull page. Choose a pull mode for the cluster (ensure, replace, or exact), then click **Pull All Link Templates**. On the Report Pull DHCPv6 Link Template page, click **OK**.
- Click **Push Link Template** for a specific template (or **Push All Link Templates**) to open the Push DHCPv6 Link Template Data to Local Cluster page. Choose a data synchronization mode (ensure, replace, or exact), move the desired cluster or clusters to the Selected table, then click **Push Data to Clusters**.

CLI Commands

To create the link template, use **link-template** *name* **create**. For example:

```
nrcmd> link-template example-link-template create [attribute=value]
```

You can set and enable the aforementioned expression setting attributes in the usual way, and you can show and list link templates. For example, to set a prefix expression for the link template, use the following file definition and pointer to the file (and assuming that the cm-prefix, cpe-address-prefix, and cpe-pd-prefix templates exist):

```
> type link-prefix-expr.txt
(list (create-prefix "cm-prefix" (create-prefix-range 32 0x1))
(create-prefix "cpe-address-prefix" (create-prefix-range 32 0x2))
(create-prefix "cpe-pd-prefix" (create-prefix-range 16 0x1)) )

nrcmd> link-template example-link-template set prefix-expr=@link-prefix-expr.txt
```

In addition:

- To clone a link template, use **link-template** *name* **create clone=***name*.
- To apply a template to one or more links, use **link-template** *name* **apply-to** {**all** | *link*[**,***link***,**...]}. You can create prefixes by using **link-template** *name* **apply-to** *link* [*prefix*], but only with one link specified.

Using Expressions in Link Templates

You can specify expressions in a link template to dynamically create prefix names, IP address ranges, and embedded options when creating a link. Expressions can include context variables and operations.

Note Expressions are not the same as DHCP extensions. Expressions are commonly used to create client identities or look up clients. Extensions (see Chapter 29, "Using Extension Points") are used to modify request or response packets.

Table 26-1 lists the link template predefined variables and Table 26-2 lists the operators. Note that these variables and operators are not case-sensitive.

Table 26-1 Link Template Expression Predefined Variables

Predefined Variable	Description
mask-length	Number of prefix mask bits (with a *template-root-prefix* defined).
prefix	Network number and length (with a *template-root-prefix* defined).
prefix-addr	Address portion of the prefix (with a *template-root-prefix* defined).
prefix-length	Number of prefix address bits (with a *template-root-prefix* defined).
template.*attribute*	Attribute of the link template.
vpn	VPN of the link.

Table 26-2 Link Template Expression Operators

Expression Operator	Description
Arithmetic Operators (unsigned integer arguments only)	
(**+** *arg1 arg2*)	Adds the two argument values, such as (+ 2 3).
(**–** *arg1 arg2*)	Subtracts the second argument value from the first one.
(***** *arg1 arg2*)	Multiplies the values of two arguments.

Table 26-2 Link Template Expression Operators (continued)

Expression Operator	Description
(*/ arg1 arg2*)	Divides the value of the first argument by that of the second one (which cannot be zero).
(*% arg1 arg2*)	Modulo arithmetic operator to determine the remainder of the result of the first argument divided by the second one.
Concatenation Operator	
(**concat** *arg1 ... argn*)	Concatenates the arguments into a string.
List Operator	
(**list** *oper1 ... opern*)	Creates an options list or list of prefixes. Required if needing more than one option for a link or prefix or more than one prefix for a link. Arguments must all be **create-v6-option** operation. Nesting is not supported.
Create Prefix Operator	
(**create-prefix** *template prefix*)	Creates a prefix based on a predefined prefix template name and the prefix, including the link VPN (assuming that a *template-root-prefix* is defined). The *prefix* argument can be the **create-prefix-addr** and **create-prefix-range** functions.
Create IP Operator	
(**create-prefix-addr** *prefix interface-id*)	Creates an IPv6 address string (assuming that a *template-root-prefix* is defined) based on the prefix name and interface ID (an IPv6 address that you can specify as a string), which is the lower 64-bit address in the prefix (which need not be contained in the parent prefix). Used in the *prefix-expr* and *options-expr* attributes.
Create Range Operator	
(**create-prefix-range** *size n*)	Creates an address range for the prefix, used in the *prefix-expr* attribute (assuming that a *template-root-prefix* is defined). The *size* is the number of bits to increase the prefix length. The *n* is the *n*th occurrence of the child prefix. The *size* and *n* must be greater than zero, the *n* must be less than or equal to the *size*, and the *size* must by less than the parent prefix length.
Create Option Operator	
(**create-v6-option** *opt val*)	Creates a DHCPv6 option, used in the *options-expr* attribute. The *opt* can be the literal string or integer identifying the option. The *val* is the string representation of the option value, as defined by the option TLV value. For example: `(list (create-v6-option "dns-servers"` `(create-prefix-addr prefix "::2"))` `(create-v6-option "domain-list" "sales.example.com,example.com"))`

Creating and Editing Links

You can create links directly. The attributes you can set for the link are:

- ***name***—User-assigned name for the link.
- ***vpn-id***—VPN that contains the link.
- ***description***—Descriptive text for the link.

- ***policy***—Shared policy used when replying to clients.
- ***free-address-config***—Identifies which trap captures unexpected free address events on this prefix. If not configured, the server looks at its *v6-default-free-address-config* attribute.

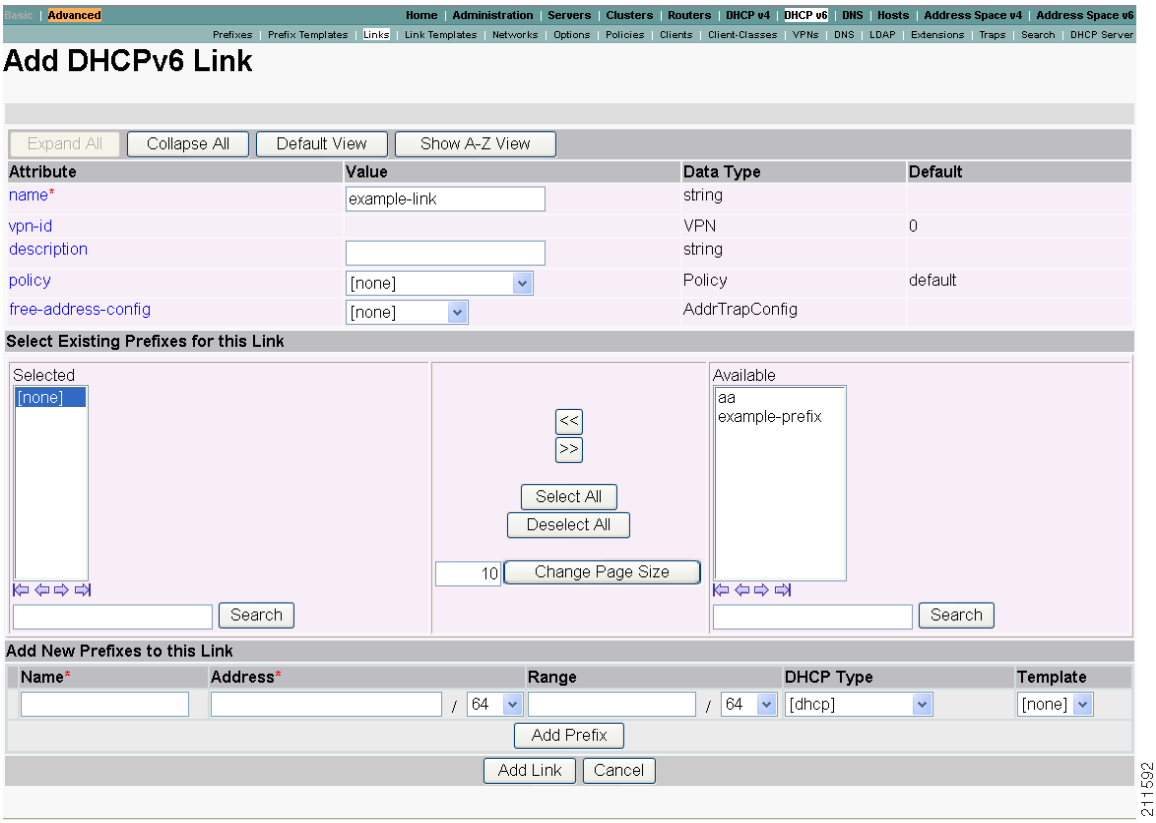
Local Advanced and Regional Web UI

Step 1 Click **DHCP v6**, then **Links**. The List DHCPv6 Links page shows the existing links.

Step 2 To add a link, click **Add Link**.

Step 3 On the Add DHCPv6 Link page (see Figure 26-2 for the local version), enter at least the name you want to give the link. You can also set the aforementioned attributes.

Figure 26-2 Add DHCPv6 Link Page (Local Advanced)



Step 4 Choose the predefined prefixes for the link by moving them to the Selected field.

Step 5 To add new prefixes for the link, enter each prefix name and its address at the bottom of the page, indicate a range, choose the DHCP type and template (if needed), then click **Add Prefix** for each one.

Step 6 Click **Add Link**.

Step 7 In the regional web UI, you can push links and their associated prefixes to local clusters. Click **Push Link** for a specific link (or **Push All Links**) to open the Push DHCPv6 Link Data to Local Cluster page. Choose a data synchronization mode (ensure, replace, or exact), move the desired cluster or clusters to the Selected table, then click **Push Data to Clusters**.

CLI Commands

Use **link** *name* **create**. (The **link** command is a synonym for the **dhcp-link** command from previous releases.) For example:

```
nrcmd> link example-link create [attribute=value]
```

To apply a link template during link creation, use **link** *name* **create template=***name* [**template-root-prefix=***address*], with the *template-root-prefix* specified if the template could create more than one prefix. To apply a template to an existing link definition, use **link** *name* **applyTemplate** *template-name* [*template-root-prefix*].

You can set and enable the aforementioned attributes in the usual way, and you can show and list links. To list prefixes or prefix names associated with a link, use **link** *name* **listPrefixes** or **link** *name* **listPrefixNames**.

Configuring Prefixes

You can configure DHCPv6 prefixes directly, or you can create prefix templates for them first. See the following subsections:

- Creating and Editing Prefix Templates
- Creating and Editing Prefixes, page 26-18

Creating and Editing Prefix Templates

You can create prefixes from predefined templates. The attributes you can set for a prefix template are the following (for the expression syntax, see the "Using Expressions in Prefix Templates" section on page 26-17):

- ***name***—User-assigned name for the prefix template.
- ***description***—Descriptive text for the prefix template.
- ***dhcp-type***—Defines how DHCP manages address assignment for a prefix:
  - **dhcp** (preset value)—Uses the prefix for stateful address assignment.
  - **stateless**—Uses the prefix for stateless option configuration.
  - **prefix-delegation**—Uses the prefix for prefix delegation.
  - **infrastructure**—Uses the prefix to map a client address to a link, when the prefix does not have an address pool.
  - **parent**—Do not have DHCP use the prefix, but use it as a container object to group child prefixes.
- ***policy***—Shared policy to use when replying to clients.
- ***prefix-name-expr***—Expression that evaluates to an AT_STRING value to use for the name of the prefix created. For example, you can have the prefix name prepended by **CM–** if you define *prefix-name-expr* as **(concat "CM-" prefix)**. In the CLI, you would include the expression in a file and point to that file:

```
> type prefix-name.txt
(concat "CM-" prefix)

nrcmd> prefix-template ex-template create prefix-name-expr=@prefix-name.txt
```

- ***prefix-description-expr***—Expression that evaluates to an AT_STRING value to apply to the description on the prefix created when using the template.
- ***range-expr***—Expression that evaluates to an AT_PREFIX value to create an address range. In the CLI, you must use a file reference. For example:

```
> type subprefix-expr.txt
(create-prefix-range 1 0x1)

nrcmd> prefix-template ex-template set range-expr=@subprefix-expr.txt
```

- ***options-expr***—Expression that evaluates to embedded policy options to create. (Use the **list** function to create multiple options.)
- ***allocation-algorithms***—One or more algorithms the server uses to select a new address or prefix to lease to a client. The available algorithms are:
  - **client-request** (preset to off)—Controls whether the server uses a client-requested lease.
  - **reservation** (preset to on)—Controls whether the server uses an available reservation for the client.
  - **extension** (preset to on)—Controls whether the server calls extensions attached at the **generate-lease** extension point to generate an address or prefix for the client. For details on extensions, see Chapter 29, "Using Extension Points."
  - **interface-identifier** (preset to off)—Controls whether the server uses the interface-identifier from the client (link-local) address to generate an address; ignored for temporary addresses and prefix delegation.
  - **random** (preset to on)—Controls whether the server generates an address using an RFC 3041 algorithm; ignored for prefix delegation.
  - **best-fit** (preset to on)—Controls whether the server delegates the first, best-fit available prefix; ignored for addresses.

  When the server needs an address to assign to a client, it processes the flags in the following order until it finds a usable address: client-request, reservation, extension, interface-identifier, and random. When the server needs to delegate a prefix to a client, it processes the flags in the following order until it finds a usable prefix: client-request, reservation, extension, and best-fit.
- ***max-leases***—Maximum number of nonreserved leases allowed on the prefix. When a new lease needs to be created, the server does so only if the limit is not exceeded. When the limit is exceeded, the server cannot create or offer new leases to clients. If you also enable SNMP traps, the *max-leases* value also calculates the percentage of used and available addresses.



Note Be sure to set the *max-leases* value to the expected maximum so that the SNMP address traps can return meaningful results.

- ***ignore-declines***—Controls whether the server responds to a DHCPv6 DECLINE message that refers to an IPv6 address or a delegated prefix from this prefix. If enabled, the server ignores all declines for leases in this prefix. If disabled (the preset value) or unset, the server sets to UNAVAILABLE every address or delegated prefix requested in a DECLINE message if it is leased to the client.
- ***deactivated***—Controls whether a prefix extends leases to clients. A deactivated prefix does not extend leases to any clients and treats all addresses in its ranges as if they were individually deactivated. The preset value is false (activated).

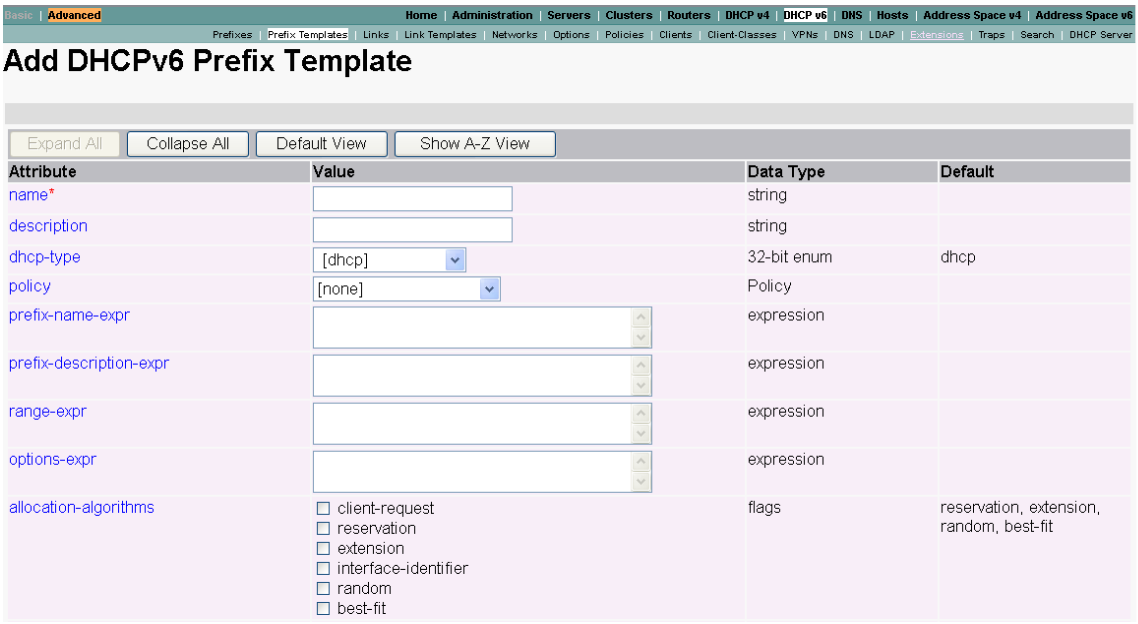
- ***expiration-time***—Time and date at which a prefix expires. After this date and time, the server neither grants new leases nor renews existing leases from this prefix. Once the *expiration-time* passes, the prefix is no longer used (although old leases and leases with grace or affinity periods continue to exist until those periods elapse). Enter a value in the format **"**[*weekday*] *month day hh*:*mm*[:*ss*] *year***"**; for example, **"Dec 31 23:59 2006"**.
- ***reverse-zone-prefix-length***—Prefix length of the reverse zone for ip6.arpa updates. (See the "Determining Reverse Zones for DNS Updates" section on page 28-4 for details.)
- ***selection-tags***—List of selection tags associated with the prefix.

Local Advanced and Regional Web UI

Step 1 Click **DHCP v6**, then **Prefix Templates**. The List DHCP Prefix Templates page shows the existing templates.

Step 2 Click **Add Prefix Template** to open the Add DHCPv6 Prefix Template page (see Figure 26-3 for a partial view of the local version).

Figure 26-3 Add Prefix Template Page (Local Advanced)



Step 3 Set the attributes and add expressions for those requiring expressions (see the "Using Expressions in Prefix Templates" section on page 26-17).

Step 4 Click **Add Prefix Template**.

Step 5 To edit a prefix template, click its name on the List DHCPv6 Prefix Template page. On the Edit DHCPv6 Prefix Template page, edit the template attributes, such as adding a selection tag, then click **Modify Prefix Template**.

Step 6 In the regional web UI, you can pull replica prefix templates or push templates to local clusters:

- Click **Pull Replica Prefix Template** to open the Select DHCPv6 Prefix Template Data to Pull page. Choose a pull mode for the cluster (ensure, replace, or exact), then click **Pull All Prefix Templates**. On the Report Pull DHCPv6 Prefix Template page, click **OK**.
- Click **Push Prefix Template** for a specific template (or **Push All Prefix Templates**) to open the Push DHCPv6 Prefix Template Data to Local Cluster page. Choose a data synchronization mode (ensure, replace, or exact), move the desired cluster or clusters to the Selected table, then click **Push Data to Clusters**.

CLI Commands

To create the prefix template, use **prefix-template** *name* **create**. For example:

```
nrcmd> prefix-template example-prefix-template create [attribute=value]
```

You can set and enable the aforementioned attributes in the usual way, and you can show and list prefix templates. In addition:

- To clone a prefix template, use **prefix-template** *name* **create clone=***name*.
- To apply a template to one or more prefixes, use **prefix-template** *name* **apply-to** {**all** | *prefix*[**,***prefix*,...]}.

Using Expressions in Prefix Templates

You can specify expressions in a prefix template to dynamically create prefix names, IP address ranges, and embedded options when creating a prefix. Expressions can include context variables and operations.

Note Expressions are not the same as DHCP extensions. Expressions are commonly used to create client identities or look up clients. Extensions (see Chapter 29, "Using Extension Points") are used to modify request or response packets.

Table 26-3 lists the prefix template predefined variables and Table 26-4 on page 26-18 lists the operators. Note that these variables and operators are not case-sensitive.

Table 26-3 Prefix Template Expression Predefined Variables

Predefined Variable	Description
prefix	Network number and length, based on the template root prefix if applying a link template to a link, or the prefix address if applying a prefix template to a prefix.
vpn	VPN of the prefix.
prefix-addr	Address portion of the prefix.
prefix-length	Number of prefix address bits.
mask-length	Number of prefix mask bits.
template.*attribute*	Attribute of the prefix template.

Table 26-4 Prefix Template Expression Operators

Expression Operator	Description
Arithmetic Operators (unsigned integer arguments only)	
(**+** *arg1 arg2*)	Adds the two argument values, such as (+ 2 3).
(**–** *arg1 arg2*)	Subtracts the second argument value from the first one, such as with *ping-timeout* defined as 100, (– template.ping-timeout 10) yields 90.
(***** *arg1 arg2*)	Multiplies the values of two arguments.
(**/** *arg1 arg2*)	Divides the value of the first argument by that of the second one (which cannot be zero).
(**%** *arg1 arg2*)	Modulo arithmetic operator to determine the remainder of the result of the first argument divided by the second one.
Concatenation Operator	
(**concat** *arg1 ... argn*)	Concatenates the arguments into a string.
List Operator	
(**list** *oper1 ... opern*)	Creates an options list or list of prefixes. Required if needing more than one option for a prefix. Arguments must all be **create-v6-option** or **create-prefix-range** operations. Nesting is not supported.
Create IP Operator	
(**create-prefix-addr** *prefix-name interface-id*)	Creates an IPv6 address string based on the prefix name and interface ID (an IPv6 address that you can specify as a string), which is the lower 64-bit address in the prefix (which need not be contained in the parent prefix). Used in the *range-expr* and *options-expr* attributes.
Create Range Operator	
(**create-prefix-range** *size n*)	Creates an address range for the prefix, used in the *prefix-expr* attribute (assuming that a *template-root-prefix* is defined). The *size* is the number of bits to increase the prefix length. The *n* is the *n*th occurrence of the child prefix. The *size* and *n* must be greater than zero, the *n* must be less than or equal to the *size*, and the *size* must by less than the parent prefix length.
Create Option Operator	
(**create-v6-option** *opt val*)	Creates a DHCPv6 option, used in the *options-expr* attribute. The *opt* can be the literal string or integer identifying the option. The *val* is the string representation of the option value, as defined by the option TLV value. For example: `(list (create-v6-option "dns-servers"` `(create-prefix-addr prefix "::2"))` `(create-v6-option "domain-list" "sales.example.com,example.com"))`

Creating and Editing Prefixes

You can create prefixes directly (and optionally apply an existing template to it; see the "Creating and Editing Prefix Templates" section on page 26-14). These are the prefix attributes that you can set:

- ***name***—Assigns a name to this prefix.
- ***vpn-id***—VPN that contains the prefix.

- ***description***—Describes the prefix.
- ***dhcp-type***—Defines how DHCP manages address assignment for a prefix:
  - **dhcp** (preset value)—Uses the prefix for stateful address assignment.
  - **stateless**—Uses the prefix for stateless option configuration.
  - **prefix-delegation**—Uses the prefix for prefix delegation.
  - **infrastructure**—Uses the prefix to map a client address to a link, when the prefix does not have an address pool.
  - **parent**—Do not have DHCP use the prefix, but use it as a container object to group child prefixes.
- ***address***—Prefix (subnet) that an interface belongs to using the high-order bits of an IPv6 address.
- ***reverse-zone-prefix-length***—Prefix length of the reverse zone for ip6.arpa updates. (See the "Determining Reverse Zones for DNS Updates" section on page 28-4 for details.)
- ***range***—Subrange the server can use to configure prefixes for address assignment. The prefix used depends on the value set for the *dhcp-type* attribute. If unset, the prefix address applies. This value can specify a longer prefix than the prefix address to limit the range of addresses or prefixes available for assignment.
- ***link***—Link associated with the prefix (subnet), used to group prefixes that are on a single link.
- ***policy***—Shared policy to use when replying to clients.
- ***selection-tags***—List of selection tags associated with the prefix.
- ***allocation-algorithms***—One or more algorithms the server uses to select a new address or prefix to lease to a client. The available algorithms are:
  - **client-request** (preset to off)—Controls whether the server uses a client requested lease.
  - **reservation** (preset to on)—Controls whether the server uses an available reservation for the client.
  - **extension** (preset to on)—Controls whether the server calls extensions attached at the **generate-lease** extension point to generate an address or prefix for the client. For details on extensions, see Chapter 29, "Using Extension Points."
  - **interface-identifier** (preset to off)—Controls whether the server uses the interface-identifier from the client (link-local) address to generate an address; ignored for temporary addresses and prefix delegation.
  - **random** (preset to on)—Controls whether the server generates an address using an RFC 3041 algorithm; ignored for prefix delegation.
  - **best-fit** (preset to on)—Controls whether the server delegates the first, best-fit available prefix; ignored for addresses.

  When the server needs an address to assign to a client, it processes the flags in the following order until it finds a usable address: client-request, reservation, extension, interface-identifier, and random. When the server needs to delegate a prefix to a client, it processes the flags in the following order until it finds a usable prefix: client-request, reservation, extension, and best-fit.
- ***max-leases***—Maximum number of nonreserved leases allowed on the prefix. When a new lease needs to be created, the server does so only if the limit is not exceeded. When the limit is exceeded, the server cannot create or offer new leases to clients. If you also enable SNMP traps, the *max-leases* value also calculates the percentage of used and available addresses.

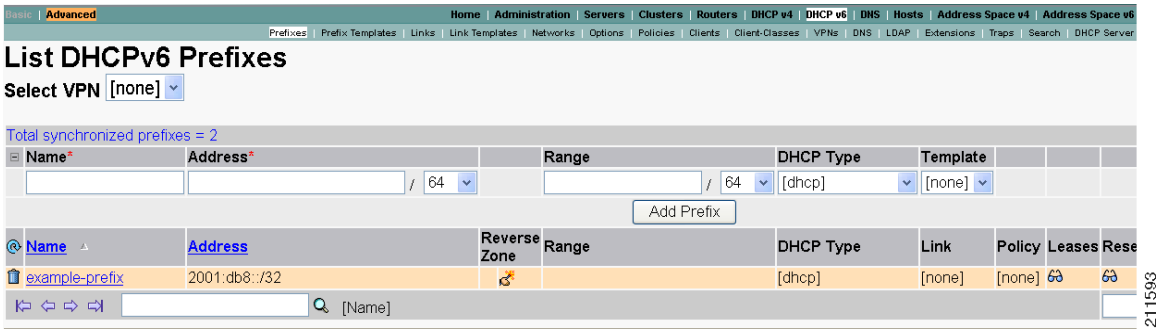
Note Be sure to set the *max-leases* value to the expected maximum so that the SNMP address traps can return meaningful results.

- ***ignore-declines***—Controls whether the server responds to a DHCPv6 DECLINE message that refers to an IPv6 address or a delegated prefix from this prefix. If enabled, the server ignores all declines for leases in this prefix. If disabled (the preset value) or unset, the server sets to UNAVAILABLE every address or delegated prefix requested in a DECLINE message if it is leased to the client.
- ***expiration-time***—Time and date at which a prefix expires. After this date and time, the server neither grants new leases nor renews existing leases from this prefix. Once the *expiration-time* passes, the prefix is no longer used (although old leases and leases with grace or affinity periods continue to exist until those periods elapse). Enter a value in the format **"**[*weekday*] *month day hh*:*mm*[:*ss*] *year***"**; for example, **"Dec 31 23:59 2006"**.
- ***deactivated***—Controls whether a prefix extends leases to clients. A deactivated prefix does not extend leases to any clients and treats all addresses in its ranges as if they were individually deactivated. The preset value is false (activated).
- ***free-address-config***—Identifies which trap captures unexpected free address events on this prefix. If not configured, the server looks for the *free-address-config* attribute value for the parent link. If that attribute is not configured, the server looks at its *v6-default-free-address-config* attribute.
- ***embedded-policy***—Policy embedded in the prefix.

Local Advanced and Regional Web UI

Step 1 Click **DHCP v6**, then **Prefixes**. The List DHCPv6 Prefixes page (see Figure 26-4 for the local version) shows the existing prefixes.

Figure 26-4 List/Add Prefixes Page (Local Advanced)



Step 2 Create the prefix:

a. If creating it in other than the current VPN, choose a VPN from the drop-down list.

b. Enter a prefix name and address, and choose a prefix length from the drop-down list.

c. If you want a range of addresses for the prefix, enter the subnet address and choose a prefix length.

d. Choose a DHCP type (see the attribute descriptions at the top of this section). The default is DHCP.

e. If you want to apply a preconfigured prefix template, choose it from the drop-down list. (Note that the attribute values of an applied template overwrite the ones set for the prefix.)

f. Click **Add Prefix**, which should add the prefix to the list.

g. Reload the DHCP server. When you return to the List DHCPv6 Prefixes page, a message indicates how many prefixes are synchronized.

Step 3 To create a reverse zone from the prefix, click the Create icon ( ) in the Reverse Zone column to open the Create Reverse Zone(s) for Prefix page. On this page, you can select a zone template, click **Report**, then **Run**. Click **Return** to return to the List DHCPv6 Prefixes page. The icon in the Reverse Zone column changes to the View icon ( ), which you can click to open the List/Add Reverse Zones page.

Step 4 Once you create a prefix, you can list and manage the leases for the prefix by clicking the View icon ( ) in the Leases column of the List DHCPv6 Prefixes page. This opens the List DHCP Leases for Prefix page. From here, you can list the leases for the client lookup key and manage each lease separately by clicking its name. Click **Return** to return to the List DHCPv6 Prefixes page.

Step 5 You can list and manage the reservations for the prefix by clicking the View icon ( ) in the Reservations column of the List DHCPv6 Prefixes page. This opens the List/Add DHCP Reservations for Prefix page. Add each reservation IP address and lookup key and whether the lookup key is a string or binary, then click **Add Reservation**. Click **Modify Prefix** to return to the List DHCPv6 Prefixes page.

Step 6 To edit a prefix, click its name on the List/Add DHCPv6 Prefix page. On the Edit DHCPv6 Prefix page, edit the prefix attributes, or create a new or edit an existing embedded policy. To manage an embedded policy:

a. Click **Create New Embedded Policy** or **Edit Existing Embedded Policy** to open the Edit DHCP Embedded Policy for Prefix page.

b. Modify the embedded policy properties (see the "DHCPv6 Policy Hierarchy" section on page 26-7).

c. Click **Modify Embedded Policy**. The next time the Edit DHCPv6 Prefix page appears, you can edit the embedded policy for the prefix.

d. Click **Modify Prefix**.

Step 7 In the regional web UI, you can push prefixes to local clusters and reclaim prefixes on the List DHCPv6 Prefixes page:

- To push the prefix, click **Push Prefix** to open the DHCPv6 Push Prefix page. Choose the cluster or prefix template to which you want to push the prefix, then click **Push Prefix**.
- To reclaim the prefix, click **Reclaim Prefix** to open the DHCPv6 Reclaim Prefix page. Choose the cluster or prefix template to which you want to reclaim the prefix, then click **Reclaim Prefix**.

CLI Commands

Use **prefix** *name* **create** *ipv6address/length*. (The **prefix** command is a synonym for the **dhcp-prefix** command from previous releases.) Reload the DHCP server. For example:

```
nrcmd> prefix example-prefix create 2001:0db8::/32 [attribute=value]
nrcmd> dhcp reload
```

To apply a prefix template during prefix creation, use **prefix** *name* **create** *ipv6address/length* **template=***name*. To apply a template to an existing prefix definition, use **prefix** *name* **applyTemplate** *template-name*. For example:

```
nrcmd> prefix example-prefix create 2001:0db8::/64 template=preftemp-1
nrcmd> prefix example-prefix applyTemplate template=preftemp-1
nrcmd> dhcp reload
```

You can set and enable the aforementioned attributes in the usual way. Add reservations by using **prefix** *name* **addReservation** *ipv6address/length lookup-key* [**–blob** | **–string**]. List leases by using **prefix** *name* **listLeases**. Manage DHCPv6 leases by using these commands:

```
nrcmd> lease6 {vpn-id/ | vpn-name/}ip6address[/prefix-length] activate
nrcmd> lease6 {vpn-id/ | vpn-name/}ip6address[/prefix-length] deactivate
nrcmd> lease6 {vpn-id/ | vpn-name/}ip6address[/prefix-length] force-available
nrcmd> lease6 {vpn-id/ | vpn-name/}ip6address[/prefix-length] get attribute
nrcmd> lease6 {vpn-id/ | vpn-name/}ip6address[/prefix-length] show
nrcmd> lease6 list
```

Tip See the "Reconfigure Support" section on page 26-27 for additional syntax.

You can get an exact count of the total prefixes and links for the DHCP server by using **dhcp getPrefixCount** [**vpn** *name* | **all**]. You can specify a VPN or all VPNs. Omitting the **vpn** *name* returns a count for the current VPN.

Viewing Address Utilization for Prefixes

You can view the current address utilization for prefixes.

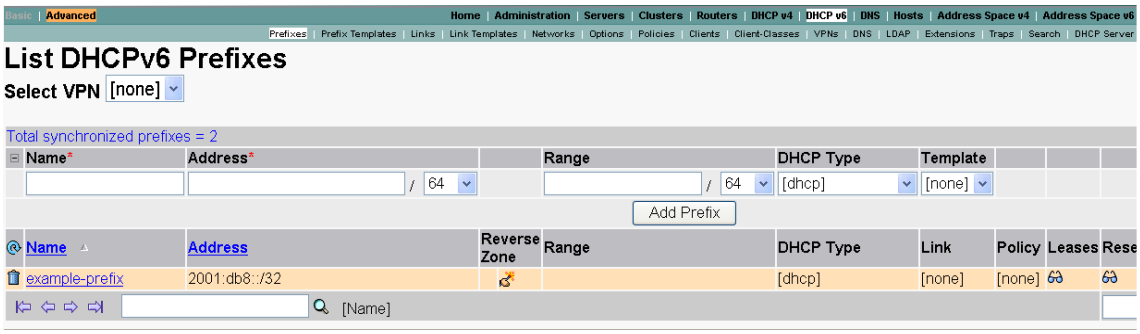
Local Advanced and Regional Web UI

The function is available on the View Unified v6 Address Space page (see the "Viewing Address Space" section on page 9-2).

Tip You can use the View Unified v6 Address Space page to push and reclaim prefixes. Click the **Push** or **Reclaim** link for the desired prefix. (See in the "Creating and Editing Prefixes" section on page 26-18 for details.)

When you click the View icon (🔭) in the Current Usage column, or the **Show Current Utilization for All Prefixes** button, the View Current Prefix Utilization Report page appears (see Figure 26-5).

Figure 26-5 View Current Prefix Utilization Report Page (Local Advanced)



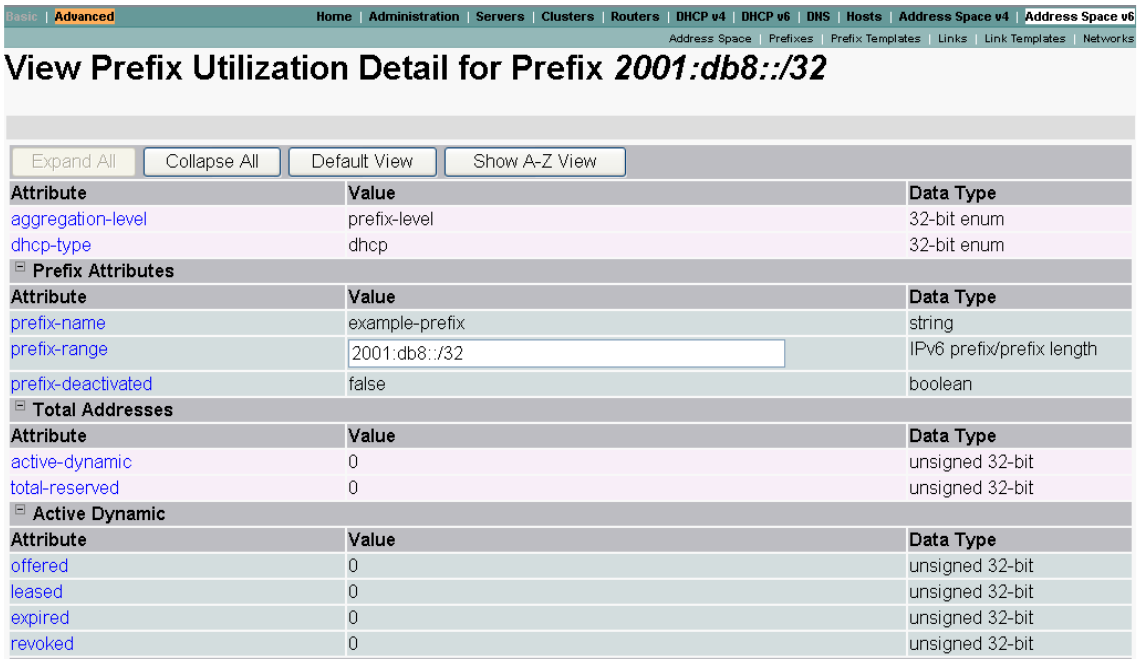
Note To ensure the proper subnet-to-server mapping on this page, you must update the regional address space view so that it is consistent with the relevant local cluster. Do this by pulling the replica address space, or reclaiming the subnet to push to the DHCP server. Also ensure that the particular DHCP server is running.

The other columns on the View Current Utilization Report page identify:

- **Range**—Address range of the prefix.
- **Type**—Whether the address space is a prefix or link.
- **Active Dynamic**—Addresses that are part of a dynamic range managed by DHCP and that are currently leased, but not reserved.

The Utilization Detail column items are expandable on the View Current Utilization Report page so that you can view the prefix or parent prefix data. Clicking the prefix or parent prefix name in this column opens the View Prefix Utilization Detail page (see Figure 26-6 for a partial view for a prefix).

Figure 26-6 View Prefix Utilization Detail Page (Local Advanced)



The View Utilization Detail page is a read-only page that shows detailed address utilization attributes for the prefix or the parent prefix (identified as Totals). The address utilization attributes are described in Table 26-5.

Table 26-5 Address Utilization Attributes

Utilization Attribute	Description
aggregation-level	Granularity of the utilization data. Prefix-level indicates the data is for the specific prefix; totals indicates the data is for the parent prefix, which is the sum of its prefix-level counters.

Table 26-5 Address Utilization Attributes (continued)

Utilization Attribute	Description
dhcp-type	DHCP address assignment type, which can be dhcp (stateful), stateless (option configuration), prefix-delegation, or infrastructure (maps a client address to a link without an address pool).
Total Addresses	
active-dynamic	Total number of dynamic leases in active use (leased, offered, released, expired, or revoked. The Active Dynamic category shows the states of these leases.
total-reserved	Total number of reserved leases.
Active Dynamic	
offered	Number of dynamic (unreserved) leases that are currently offered to clients, but not yet acknowledged as being leased.
leased	Number of dynamic leases that are currently acknowledged as leased to clients.
expired	Number of dynamic leases that are past the lease expiration period, but will not be available for other clients (except after the policy grace-period expires).
revoked	Number of dynamic leases that the client can no longer use, but that some other client could be using.
Reserved	
reserved-active	Number of reserved leases that clients are actively using.
reserved-inactive	Number of reserved leases that clients are not actively using.
Unavailable	
unavail	Number of unreserved dynamic leases that a client declines or the server marks with an address conflict (usually indicating configurations that need correcting).
reserved-unavail	Number of reserved leases that a client declines or the server marks with an address conflict (usually indicating configurations that need correcting).
Deactivated	
deactivated	Number of dynamic and reserved leases that clients are actively leasing (that are not offered, expired, or released), but that an administrator deactivated.
leased-deactivated	Number of dynamic leases that an administrator deactivated.
reserved-leased-deactivated	Number of reserved leases that an administrator deactivated.

Viewing DHCPv6 Networks

To view the networks in the DHCPv6 address space, click **DHCP v6**, then **Networks** to open the View DHCPv6 Networks page. On this page you can add DHCPv6 links using a template and a template root prefix, as you would on the List DHCPv6 Links page. Adding a link opens the Add DHCPv6 Link page. After creating the link, you can select it on the View DHCPv6 Networks page for editing.

Editing DHCPv6 Server Attributes

You can edit DHCP server attributes related to DHCPv6. These attributes are:

- ***v6-client-class-lookup-id***—Expression that determines a client-class based on the DHCPv6 client request and returns a string or <none>. The expression must return a string that is the name of a configured client-class. The attribute has no preset value.
- ***max-client-leases***—Maximum number of leases a DHCPv6 client can have on a link. (Do not use this attribute to limit clients to one lease only.) The preset value is 200.

Local Basic or Advanced Web UI

Click **DHCP v6**, then **DHCP Server** to open the Manage DHCP Server page. Click the Local DHCP Server link to open the Edit DHCP Server page, modify the aforementioned DHCPv6 attribute values, then click **Modify Server**.

CLI Commands

Use **dhcp** to show the aforementioned DHCPv6 server attributes, then modify them by using **dhcp set**.

Configuring DHCPv6 Policies

You can edit DHCPv6 policy attributes, which are:

- ***affinity-period***—See the "Lease Affinity" section on page 26-5 (no preset value).
- ***allow-non-temporary-addresses***—Enable or disable DHCPv6 clients requesting nontemporary (IA_NA) addresses (preset value enable).
- ***allow-rapid-commit***—With Rapid Commit enabled, clients receive information (when solicited) on committed addresses, which are then more quickly committed with a client request (preset value disable). Use Rapid Commit only if one DHCP server is servicing clients, otherwise it might seem like the client is receiving multiple addresses. (See the "DHCPv6 Policy Hierarchy" section on page 26-7 for special handling of this attribute, and Reconfigure support, when used in an embedded or named policy for a prefix.)
- ***allow-temporary-addresses***—Enable or disable DHCPv6 clients requesting temporary (IA_IA) addresses (preset value enable).
- ***default-prefix-length***—For prefix delegation, default prefix length of the delegated prefix if the client or router does not explicitly request it (or *allow-client-hints* is disabled); must always be less than or equal to the prefix range prefix length (preset value 64 bytes).
- ***preferred-lifetime***—Default and maximum preferred lifetime for leases (preset value 1 week).
- ***v6-reply-options***—DHCPv6 options returned in replies to clients (no preset value). (See the "DHCPv6 Policy Hierarchy" section on page 26-7 for special handling of this attribute when used in an embedded or named policy for a prefix.)
- ***valid-lifetime***—Default and maximum valid lifetime for leases (preset value 2 weeks).



Tip For details on the Reconfigure attributes, see the "Reconfigure Support" section on page 26-27.

Local Advanced Web UI

Click **DHCP v6**, then **Policies** to open the List DHCP Policies page. Click **Add Policy** to add a new policy on the Add DHCP Policy page or click an existing policy to open the Edit DHCP Policy page. Both pages have DHCPv4 and DHCPv6 options sections. Add (or delete) options and set attributes as desired, then click **Add Policy** or **Modify Policy**.

CLI Commands

Use **policy list** or **policy** *name* **show** to show the aforementioned policy attributes, then modify them by using **policy** *name* **set** or **enable**.

Configuring DHCPv6 Client-Classes

You can configure DHCPv6 client-class attributes, which are:

- ***v6-client-lookup-id***—Key value to use to look up the DHCPv6 client in the client database (locally or through LDAP), specified as an expression that evaluates to a string (or a blob as a valid string).
- ***v6-override-client-id***—Value that replaces any client-identity value in an incoming packet, specified as an expression that evaluates to a blob.

Local Advanced Web UI

Step 1 Click **DHCP v6**, then **Client-Classes** to open the List DHCP Client-Classes page.

Step 2 Click an existing client-class to open the Edit DHCP Client-Class page, or click **Add Client-Class** to add a new client-class on the Add DHCP Client-Class page. Both pages include the aforementioned attributes.

Step 3 Click **Modify Client-Class**.

Step 4 To generate clients, be sure that *validate-client-name-as-mac* is disabled for the DHCP server. This attribute appears on the Edit DHCP Server page under the Client-Class attributes.

Step 5 Reload the DHCP server.

CLI Commands

Use **client-class list** or **client-class** *name* **show** to show the aforementioned client-class attributes, then modify them using **client-class** *name* **set**. To generate clients, be sure that *validate-client-name-as-mac* is disabled for the DHCP server.

Configuring DHCPv6 Clients

You can configure DHCPv6 clients.

Local Advanced Web UI

Click **DHCP v6**, then **Clients** to open the List/Add DHCP Clients page. Click an existing client to open the Edit DHCP Client page or click **Add Client** to add a new client-class on the List/Add DHCP Client page, choose the client-class that includes the DHCPv6 attributes that were set (see the "Configuring DHCPv6 Client-Classes" section on page 26-26), then click **Modify Client**.



Tip Disable the *validate-client-name-as-mac* attribute for the DHCP server.

CLI Commands

Use **client list** or **client** *name* **show** to show the existing clients. To set the client-class name for the client, use **client** *name* **set client-class-name=***value*. Also ensure that the *validate-client-name-as-mac* attribute is disabled for the DHCP server.

Setting DHCPv6 Options

Set DHCPv6 options and vendor options when you create or edit policies (embedded or named) for prefixes. (See the "DHCPv6 Policy Hierarchy" section on page 26-7 for special handling of the *v6-options* and *v6-vendor-options* policy attributes when used in an embedded or named policy on a prefix.)

Local Advanced Web UI

The DHCPv6 options coexist along with the DHCPv4 options on the Add DHCP Policy or Edit DHCP Policy page. Note that the vendor options appear only if you create these options (see the "Creating DHCP Option Definition Sets and Option Definitions" section on page 21-9).

You can select the options from the drop-down lists. If option descriptions exist, they appear under the Name and Number headings, which you can click to sort the entries.

CLI Commands

Use **policy** *name* **setV6Option** or **policy** *name* **setV6VendorOption**. The option settings require an option name (or ID) and a value. For example:

```
nrcmd> policy dhcpv6-policy setV6Option dns-servers 2222::1,2222::2
nrcmd> policy dhcpv6-policy setV6VendorOption 1234 2222::3,2222::4
```

Reconfigure Support

For DHCPv6, a server can send a DHCPRECONFIGURE message to a client to inform the client that the server has new or updated configuration parameters. If so authorized and through proper authentication, the client then immediately initiates a Renew, Rebind, or Information-request reply transaction with the server so that the client can retrieve the new data. Without this support, a client must wait until it renews its lease to get configuration updates.

You can have the server unicast the Reconfigure packet or deliver it through a relay agent. If you do not specify either way, the client's client-class policy, requested lease's prefix or link policies, or system_default_policy (but not the client policy) determines the preferred method. If the unicast method

is not available (the client has no valid address lease), the server uses the relay agent; with no relay agent, the server tries to unicast; failing both results in an error. With the unicast method, if the specified lease is not usable, the server selects the lease with the longest valid lifetime.

The server and client negotiate Reconfigure support through the added security of a reconfigure key. The internal process is basically:

1. The client sends the server a REQUEST, SOLICIT, or ADVERTISE packet that includes the *reconfigure-accept* option (20) to indicate that the client wants to accept Reconfigure messages. (Conversely, the DHCP server can send a *reconfigure-accept* option to the client about whether the client should accept Reconfigure messages.) This option is required for Reconfigure support.
2. If the Network Registrar policy for the client has the *reconfigure* attribute set to **allow** or **require** (rather than **disallow**), the DHCP server accepts the packet and generates a reconfigure key for the client. (The server records the key value and its generation time in the *client-reconfigure-key* and *client-reconfigure-key-generation-time* attributes for the DHCPv6 lease.)
3. The server sends a Reply packet to the client with the reconfigure key in the *auth* option (11) along with the *reconfigure-accept* option.
4. The client records the reconfigure key to authenticate Reconfigure messages from the server.
5. When the server wants to reconfigure the client, it sends a Reconfigure packet with the *reconfigure-message* option (19) and an *auth* option containing a hash generated from the packet and the reconfigure key. The *reconfigure-message* option indicates in the *msg-type* field whether the client should respond with a Renew or an Information-request packet.
6. Upon receiving the packet, the client validates that the *auth* option contains the valid hash, then returns a Renew, Rebind, or Information-request packet. This packet includes an Option Request (*oro*) option (6) to indicate specific option updates. (If the server does not receive a reply from the client in a preconfigured timeout value of 2 seconds, the server retransmits the Reconfigure message at most 8 times, then aborts the reconfigure process for the client.)
7. The server sends the client a Reply packet that includes options for configuration parameters. The packet might also include options containing addresses and new values for other configuration parameters, even if the client did not request them. The client records these changes.



Note For details on how Reconfigure support affects particular DHCP extension points, see the "Extension Dictionaries" section on page 29-22.

Local Advanced Web UI

The List DHCP Leases for Prefix page includes a **Reconfigure** button in the Reconfigure column for each lease so that you can initiate a reconfiguration request for that particular lease.

CLI Commands

To support Reconfigure, Network Registrar 7.0 includes the following syntax for the **lease6** command:

```
lease6 ipaddr reconfigure [renew | rebind | information-request] [-unicast | -via-relay]
```

The options determine whether to have the client respond to the Reconfigure message with a Renew, Rebind, or Information-request packet, and whether the server should unicast or go through a relay agent. The **lease6 list** and **show** commands also display values for these related attributes:

- ***client-reconfigure-key***—128-bit key that the server generates for Reconfigure messages to the client.
- ***client-reconfigure-key-generation-time***—Time at which the server generated the *client-reconfigure-key*.

The **policy** command includes two related attribute settings:

- ***reconfigure***—Whether to allow (1), disallow (2), or require (3) Reconfigure support; the preset value is allow (1).
- ***reconfigure-via-relay***—Whether to allow reconfiguration over a relay agent; the preset value is false, whereby reconfiguration notification is by unicasting from the server.

DNS Update for DHCPv6

For details on enabling and configuring DNS update for DHCPv6 clients, see the "DNS Update for DHCPv6" section on page 28-2.



CHAPTER 27

Configuring DHCP Failover

DHCP failover is a protocol designed to allow a backup DHCP server to take over for a main server if the main server is taken off the network for any reason.

See Also

Failover Scenarios
Failover Checklist, page 27-3
Creating and Synchronizing Failover Server Pairs, page 27-4
Confirming Failover, page 27-9
State Transitions During Integration, page 27-9
Setting Advanced Failover Attributes, page 27-12
Changing Failover Server Roles, page 27-20
Supporting BOOTP Clients in Failover, page 27-22
DHCPLEASEQUERY and Failover, page 27-23
Troubleshooting Failover, page 27-23

Failover Scenarios

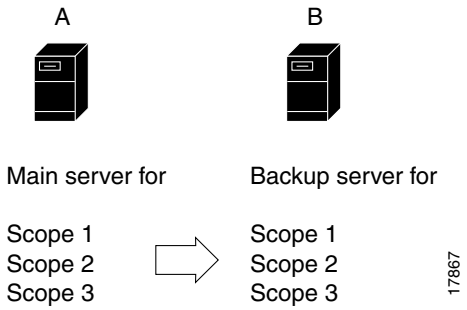
There are three basic failover scenarios:

- **Simple failover (recommended)**—One server acting as main and its partner acting as backup (see the "Simple Failover" section).
- **Back office failover**—Two mains having the same backup server (see the "Back Office Failover" section on page 27-2).
- **Symmetrical failover**—Two servers acting as main and backup for each other (see the "Symmetrical Failover" section on page 27-3).

Simple Failover

Simple failover involves a main server and a single backup server pair (see Figure 27-1 on page 27-2). In the example, main server A has three scopes that must be configured identically on backup server B.

Figure 27-1 Simple Failover Example



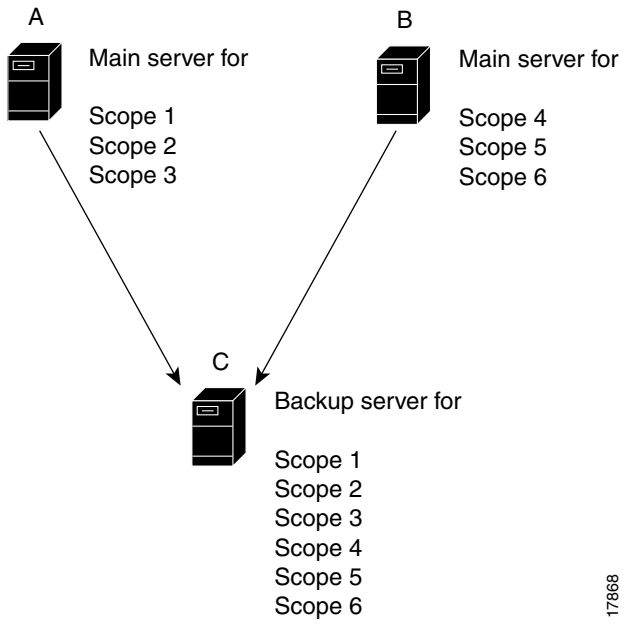
The advantages of simple failover over the other scenarios are:

- It is the easiest to manage as the network changes. It is fully supported by the web UI so that changes to the main server configuration are automatically propagated to the backup server.
- Provides the greatest performance benefits.
- Having the additional load balancing feature in effect eliminates the need for a back office or symmetrical scenario (see the "Setting Load Balancing" section on page 27-19).

Back Office Failover

Back office failover involves two (or more) main servers that share the same backup server (see Figure 27-2). In the example, main servers A and B have different scopes, and backup server C must include all these scopes. This scenario is appropriate for scopes on the same LAN segment, which require the same main and backup servers, but with the sets of scopes on different LAN segments.

Figure 27-2 Back Office Failover Example



An advantage of back office failover over the other scenarios is that it reduces the number of servers managed. However, simple failover is still recommended, because in back office failover:

- The backup server must be sized to handle the sum of the configurations.
- Changes to any of the main servers must be duplicated on the backup server.
- The increased complexity can substantially reduce the actual availability.

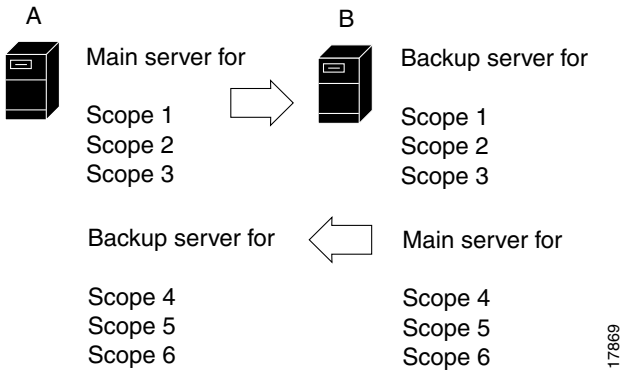
Symmetrical Failover

Symmetrical failover involves servers that act as backups for each other (see Figure 27-3). This scenario is extremely tricky in that there can be no variance in scope attribute values between the servers for the relationship to work properly.

Symmetrical failover used to be a way of load balancing the servers, although this is now more effectively done in a simple failover scenario using the load balancing feature (see the "Setting Load Balancing" section on page 27-19).

Unfortunately, symmetrical failover provides little to no performance benefit over the simple or back office scenarios. A backup server operates at about 40% of the main server to keep its lease database synchronized. If the servers back each other up, a portion of their processing capacity goes to this task, with less capacity available to service clients. Moreover, because each scope must be configured individually, symmetrical failover is more prone to configuration errors.

Figure 27-3 Symmetrical Failover Example



Failover Checklist

Use this checklist to prepare for an effective failover configuration:

- Duplicate the scope, policy, DHCP option, and address configurations on the partner servers by configuring a failover server pair for a simple failover scenario. In the case of a back office or symmetrical failover scenario, you need to modify the Network Match List for the server pair configuration to include the network address or addresses that identify the correct scope or scopes.
- Ensure that both partners are configured with a wide enough range of addresses so that the backup server can provide leases for a reasonable amount of time while the main server is down.

- The following objects must have identical configurations on both servers:
  - Scopes
  - Scope-selection tags
  - Policies
  - IP addresses
  - Reservations
  - Clients
  - Client-classes
  - Dynamic DNS updates
  - Dynamic BOOTP
  - Virtual private networks (VPNs)
  - DHCP extensions
- If you use LDAP, direct the partner servers to the same LDAP server.
- If you use BOOTP relay (IP helpers), configure all BOOTP relay agents to point to both partners. Network Registrar does not automatically detect this. You can only detect BOOTP configuration errors by performing live tests in which you periodically take the main server out of service to verify that the backup server is available to DHCP clients.

Creating and Synchronizing Failover Server Pairs

A failover pair consists of a main and backup DHCP server. You can create failover pairs in the local and regional cluster web UIs. You must synchronize the main and backup servers.

See Also

Adding Failover Pairs
Synchronizing Failover Pairs, page 27-6
Restarting the Failover Servers, page 27-8

Adding Failover Pairs

First create the DHCP failover pair based on cluster main and backup servers. You then synchronize the failover pair so that the scopes, policies, clients, extensions, and other DHCP properties match between the servers.

Web UI

Step 1 Click **DHCP**, then **Failover** to open the List DHCP Failover Pairs page.

Step 2 Click **Add DHCP Failover Pair**.

Step 3 On the Add DHCP Failover Pair page, add a failover pair name. This can be any distinguishing name and is required.

Step 4 Proceed stepwise:

a. Choose the cluster for the main DHCP server. This can be localhost or some other cluster you define. Whatever you select here becomes the IP address value for the *main-server* attribute once you add the failover pair.

Note If you change the IP address of your local host machine, you must modify the localhost cluster (on the Edit Cluster page) to change the address in the IP Address field. Do not set the value to 127.0.0.1.

b. Choose the cluster for the backup DHCP server. This cannot be the same as the main server cluster, but it must be localhost if the main cluster is not localhost. Whatever you select here becomes the IP address value for the *backup-server* attribute once you add the failover pair.

c. For each address block to include in the failover configuration, enter its IP address and mask, then click **Add Address Block**. (See the "Adding Address Blocks" section on page 9-5.)

d. You should not change the IP address values of the *main-server* and *backup-server* attributes unless you have multihomed hosts.

e. If you have a more complex failover scenario, include the list of IP addresses in the Network Match List that identifies the subnets for the primary scopes (or secondary scopes with primary subnets) that you need for the configuration. Add the IP addresses, then click **Add Network Match**.

f. You can set additional attributes, such as the maximum client lead time (*mclt*) or backup percentage (*backup-pct*). Most of the default values are optimized. Leave the *failover* attribute enabled by default unless you want to temporarily disable failover for the pair.

Step 5 Click **Add Failover Pair**. You can edit the failover pair properties.

See Also

Failover Scenarios, page 27-1
Failover Checklist, page 27-3
Synchronizing Failover Pairs, page 27-6
Restarting the Failover Servers, page 27-8

CLI Commands

Use **failover-pair** *name* **create** *main-server-address backup-server-address*. If you want to synchronize between the servers, you must also specify the main and backup clusters by setting the *main* and *backup* attributes. For example:

```
nrcmd> failover-pair example-fo-pair create 192.168.50.1 192.168.60.1 main=Example-cluster
backup=Boston-cluster
```

Configure more complex failover by using **failover-pair** *name* **addMatch** [*vpn*]*/ipaddr/mask*. Specify the IP address of the subnet that identifies the scope or scopes to be included. Use the command for each match. Use **failover-pair** *name* **removeMatch** [*vpn*]*/ipaddr/mask* to remove a match and **failover-pair** *name* **listMatches** to list the matches. You can also specify the match list when creating the failover pair. For example:

```
nrcmd> failover-pair example-fo-pair create 192.168.50.1 192.168.60.1 main=Example-cluster
backup=Boston-cluster addMatch 192.168.70.0/24
```

Synchronizing Failover Pairs

Once you create the failover pairs, you must synchronize the servers.

Tip In Expert mode, the List DHCP Failover Pairs page provides a **Resync CCM Failover Pairs** button.

Web UI

Step 1 On the List DHCP Failover Pairs page, click the Report icon ( ) to open the Report Synchronize Failover Pair page.

Step 2 Choose the synchronization operation, depending on the degree to which you want the main server property values to replace those of the backup server. There are three basic operations:

- **Update**—This is the default and least radical operation. It is appropriate for update synchronizations in that it has the least effect on the unique properties of the backup server.
- **Complete**—This operation is appropriate for all initial synchronizations. It is more complete than an update operation, while still preserving many of the backup server unique properties, such as are required for back office failover configurations.
- **Exact**—This operation is appropriate for initial simple and symmetrical failover configurations, and is not appropriate for back office configurations. It makes the two servers as much as possible mirror images of each other, although it retains unique DHCP server, LDAP remote server, and extension points on the backup server. (For initial failover configurations, use the Exact or Complete operation.)

Each operation performs a different mix of functions on the failover properties, as described in Table 27-1 on page 27-7. There are four functions, with examples based on these property name-value pairs:

On the main server:	On the backup server:
Name1=A	Name2=B
Name2=C	Name3=D

- **no change**—Makes no change to the list of properties or their values on the backup server. For the example, the result would be Name2=B, Name3=D.
- **ensure**—Ensures that a copy of the main server property exists on the backup server, but does not replace its value. For the example, the result would be Name1=A, Name2=B, Name3=D.
- **replace**—Replaces the value of a property that the two servers have in common with that of the main server. For the example, the result would be Name1=A, Name2=C, Name3=D.
- **exact**—Puts an exact copy of the main server list of properties and values on the backup server and removes the unique ones. For the example, the result would be Name1=A, Name2=C.

Table 27-1 Failover Pair Synchronization Functions

Data Description	Update	Complete	Exact
DHCP Server (server level failover pair):	replace	replace	replace
Client-Class Properties Client Hostname Properties DNS Update Properties Failover Tuning Properties (See Table 27-2 for a list of the failover pair attributes affected by failover synchronization)			
All other properties	no change	replace	replace
LDAP Remote Server	ensure	replace	exact
Policy:			
Option List Properties	ensure	replace	exact
Packet Boot File Properties	ensure	replace	exact
All other properties	replace	replace	exact
Client	replace	replace	exact
Client-Class	replace	replace	exact
Scopes (related to failover pair)	exact	exact	exact
DNS Update Configuration	replace	replace	exact
Trap Configuration	ensure	replace	exact
VPN	replace	replace	exact
Key	replace	replace	exact
Extensions (You must copy extension files.)	ensure	replace	exact
Extension Point	replace	replace	replace
Option Information:	ensure	replace	exact
Custom options list Vendor options list			

Table 27-2 Failover Pair Attributes Affected by Failover Synchronization

Affected Failover Pair Attributes
failover-bulking
failover-poll-interval
failover-poll-timeout
recover

Step 3 Click **Run** on the Run Synchronize Failover page, or **Report** on the Report Synchronize Failover page:

- If you click **Run** and if the connection was accepted, the resulting View DHCP Failover Pair Sync Report page shows what change entries the synchronization added.
- If you click **Report**, the resulting View DHCP Failover Pair Sync Report page shows what change entries the synchronization will apply if you run the synchronization. A **Run Update**, **Run Complete**, or **Run Exact** button indicates what kind of synchronization you want to perform. Click the applicable button to run the synchronization.

Step 4 On the List DHCP Failover Pairs page, click the View icon ( ) in the Manage Servers column to open the Manage DHCP Failover Servers page.

Step 5 Click the Reload icon ( ) next to the backup server to reload the backup server.

Step 6 Try to get a lease.

Step 7 On the Manage DHCP Failover Servers page, look at the health of the servers (they should show as). Also, click the Logs icon ( ) to view the log entries on the Log for Server page, and ensure that the servers are in NORMAL failover mode. The log file should contain an item similar to the following:

```
06/19/2003  9:41:19 name/dhcp/1 Info Configuration 0 04092 Failover is enabled
server-wide. Main server name: '192.168.0.1', backup server name: '192.168.0.110', mclt =
3600, backup-pct = 10, dynamic-bootp-backup-pct = 0, use-safe-period: disabled,
safe-period = 0.
```

CLI Commands

Use **failover-pair** *name* **sync** {**update** | **complete** | **exact**}:

```
nrcmd> failover-pair example-fo-pair sync exact
```

Restarting the Failover Servers

For any failover synchronization to take effect, you must first connect to, and restart, both the main and backup failover servers.

Step 1 On the List Failover Pairs page, click the Go Local icon ( ) in the Main DHCP Server column.

Step 2 On the local Manage DHCP Server page for the main server, click the Reload icon ( ) on the right-hand side of the page.

Step 3 Click the Go Regional icon ( ) at the top-right corner of the page.

Step 4 On the regional List Failover Pairs page, click the Go Local icon ( ) in the Backup DHCP Server column.

Step 5 On the local Manage DHCP Server page for the backup server, click the Reload icon ( ) on the right-hand side of the page.

Step 6 Click the Go Regional icon ( ) at the top-right corner of the page.

See Also

Confirming Failover, page 27-9

Confirming Failover

Step 1 Ping from one server to the other to verify TCP/IP connectivity. Make sure that routers are configured to forward clients to both servers.

Step 2 Check that the server is in NORMAL mode by clicking the Related Servers icon ( ) on the Manage DHCP Server or List DHCP Failover Pairs page, or use **dhcp getRelatedServers** in the CLI.

Step 3 After startup, have a client attempt to get a lease.

Step 4 Set the log settings on the main server to include at least *failover-detail*.

Step 5 Confirm that the name_dhcp_1_log log file (in *install-path*/logs) on the main server contains DHCPBNDACK or DHCPBNDUPD messages from each server.

Step 6 Confirm that the name_dhcp_1_log log file on the backup server contains messages that the backup server is dropping requests because failover is in NORMAL state.

Step 7 Repeat Step 2.

See Also

State Transitions During Integration
Setting Advanced Failover Attributes, page 27-12

State Transitions During Integration

During normal operation, the failover partners transition between states. They stay in their current state until all the actions for the state transition are completed and, if communication fails, until the conditions for the next state are fulfilled. Table 27-3 describes what happens when servers enter various states and how they initially integrate and later reintegrate with each other under certain conditions.

Table 27-3 Failover State Transitions and Integration Processes

Integration	Results
Into NORMAL state, the first time the backup server contacts the main server	1. The newly configured backup server contacts the main server, which starts in PARTNER-DOWN state. 2. Because the backup server is a new partner, it goes into RECOVER state and sends a Binding Request message to the main server. 3. The main server replies with Binding Update messages that include the leases in its lease state database. 4. After the backup server acknowledges these messages, the main server responds with a Binding Complete message. 5. The backup server goes into RECOVER-DONE state. 6. Both servers go into NORMAL state. 7. The backup server sends Pool Request messages. 8. The main server responds with the leases to allocate to the backup server based on the *backup-pct* configured.

Table 27-3 Failover State Transitions and Integration Processes (continued)

Integration	Results
After COMMUNICATIONS-INTERRUPTED state	1. When a server comes back up and connects with a partner in this state, the returning server moves into the same state and then immediately into NORMAL state. 2. The partner also moves into NORMAL state.
After PARTNER-DOWN state	When a server comes back up and connects with a partner in this state, the server compares the time it went down with the time the partner went into this state. • If the server finds that it went down and the partner subsequently went into this state: a. The returning server moves into RECOVER state and sends an Update Request message to the partner. b. The partner returns all the binding data it was unable to send earlier and follows up with an Update Done message. c. The returning server moves into RECOVER-DONE state. d. Both servers move into NORMAL state. • If the returning server finds that it was still operating when the partner went into PARTNER-DOWN state: a. The server goes into POTENTIAL-CONFLICT state, which also causes the partner to go into this state. b. The main server sends an update request to the backup server. c. The backup server responds with all unacknowledged updates to the main server and finishes off with an Update Done message. d. The main server moves into NORMAL state. e. The backup server sends the main server an Update Request message requesting all unacknowledged updates. f. The main server sends these updates and finishes off with an Update Done message. g. The backup server goes into NORMAL state.

Table 27-3 Failover State Transitions and Integration Processes (continued)

Integration	Results
After the server loses its lease state database	A returning server usually retains its lease state database. However, it can also lose it because of a catastrophic failure or intentional removal. 1. When a server with a missing lease database returns with a partner that is in PARTNER-DOWN or COMMUNICATIONS-INTERRUPTED state, the server determines whether the partner ever communicated with it. If not, it assumes to have lost its database, moves into RECOVER state, and sends an Update Request All message to its partner. 2. The partner responds with binding data about every lease in its database and follows up with an Update Done message. 3. The returning server waits the maximum client lead time (MCLT) period, typically one hour, and moves into RECOVER-DONE state. For details on the MCLT, see the "Setting the Maximum Client Lead Time" section on page 27-15. 4. Both servers then move into NORMAL state.
After a lease state database backup restoration	When a returning server has its lease state database restored from backup, and if it reconnects with its partner without additional data, it only requests lease binding data that it has not yet seen. This data may be different from what it expects. 1. In this case, you must configure the returning server with the *failover-recover* attribute set to the time the backup occurred. 2. The server moves into RECOVER state and requests all its partner data. The server waits the MCLT period, typically one hour, from when the backup occurred and goes into RECOVER-DONE state. For details on the MCLT, see the "Setting the Maximum Client Lead Time" section on page 27-15. 3. Once the server returns to NORMAL state, you must unset its *failover-recover* attribute, or set it to zero. `nrcmd> dhcp set failover-recover=0`

Table 27-3 Failover State Transitions and Integration Processes (continued)

Integration	Results
After the operational server had failover disabled	If the operating server had failover enabled, disabled, and subsequently reenabled, you must use special considerations when bringing a newly configured backup server into play. The backup server must have no lease state data and must have the *failover-recover* attribute set to the current time minus the MCLT interval, typically one hour. For details on the MCLT, see the "Setting the Maximum Client Lead Time" section on page 27-15. 1. The backup server then knows to request all the lease state data from the main server. Unlike what is described in "After the server loses its lease state database" section of this table, the backup server cannot request this data automatically because it has no record of having ever communicated with the main server. 2. After reconnecting, the backup server goes into RECOVER state, requests all the main server lease data, and goes into RECOVER-DONE state. 3. Both servers go into NORMAL state. At this point, you must unset the backup server *failover-recover* attribute, or set it to zero. `nrcmd> dhcp set failover-recover=0`

Setting Advanced Failover Attributes

The advanced failover properties that are important to set are the following:

- Backup percentage (see the "Setting Backup Percentages" section on page 27-13)
- Backup allocation boundaries (see the "Setting Backup Allocation Boundaries" section on page 27-15)
- Maximum client lead time (MCLT) (see the "Setting the Maximum Client Lead Time" section on page 27-15)
- Safe period (see the "Using the Failover Safe Period to Move Servers into PARTNER-DOWN State" section on page 27-16)
- Request and response packet buffers (see the "Setting DHCP Request and Response Packet Buffers" section on page 27-18)
- Polling attributes (see the "Changing Polling Attributes" section on page 27-18)
- Network discovery (see the "Setting the Network Discovery Attribute" section on page 27-19)
- Load balancing (see the "Setting Load Balancing" section on page 27-19)

Setting Backup Percentages

To keep failover partners operating despite a network partition (when both servers can communicate with clients, but not with each other), allocate more addresses than for a single server. Configure the main server to allocate a percentage of the currently available addresses in each scope to the backup server. This makes these addresses unavailable to the main server. The backup server uses these addresses when it cannot talk to the main server and cannot tell if it is down.

Note If a Network Registrar failover server receives an update from a Network Registrar DHCP server running prior to Network Registrar 6.0, the unavailable leases do not have a timeout value. In this case, the Network Registrar 6.2 and later server uses the *unavailable-timeout* value configured in the scope policy or **system_default_policy** policy as the timeout for the unavailable lease. When the lease times out, the policy causes the lease to transition to available in both failover partners.

You can set the percentage of currently available addresses by setting the *backup-pct* attribute on the failover pair or scope (**failover-pair** *name* **set backup-pct** or **scope** *name* **set backup-pct** in the CLI). Note that setting the backup percentage on the failover pair level sets the value for all scopes not set with that attribute. However, if set at the scope level, the backup percentage overrides the one at the failover pair level. If the *load-balancing* attribute is enabled for the failover pair (**failover-pair** *name* **enable load-balancing** in the CLI), the backup percentage is fixed at 50% and any of the backup percentage attributes (on a failover pair or scope) are ignored. (See the "Load Balancing Compatibility with Earlier Network Registrar Versions" section on page 27-19.)

The backup percentage should be set large enough to allow the backup server to continue serving new clients in the event that the main server fails. The backup percentage is calculated based on the number of available addresses. The default backup percentage is 10% (unless load balancing is in effect, then it is 50%). However, this number can safely be set to a larger value, if extended outages are expected, because the main server periodically reclaims addresses (once per hour) if, in the course of normal leasing activity, the main server's available address pool drops below its predefined percentage. For example, with the default 10% backup percentage, the main server will reclaim addresses if its address pool falls below 90%.

The percentage depends on the new client arrival rate and the network operator reaction time. The backup server needs enough addresses from each scope to satisfy all new clients requests arriving during the time it does not know if the main server is down. Even during PARTNER-DOWN state, the backup server waits for the maximum client lead time (MCLT) and lease time to expire before reallocating leases. See the "Setting the Maximum Client Lead Time" section on page 27-15. When these times expire, the backup server offers:

- Leases from its private pool.
- Leases from the main server pool.
- Expired leases to new clients.

During the day, an operator likely responds within two hours to COMMUNICATIONS-INTERRUPTED state to determine if the main server is working. The backup server then needs enough addresses to support a reasonable upper bound on the number of new clients that could arrive during those two hours.

During off-hours, the arrival rate of previously unknown clients is likely to be less. The operator can usually respond within 12 hours to the same situation. The backup server then needs enough addresses to support a reasonable upper bound on the number of clients that could arrive during those 12 hours.

The number of addresses over which the backup server requires sole control is the greater of the two numbers. You would express this number as a percentage of the currently available (unassigned) addresses in each scope. If you use client-classes, remember that some clients can only use some sets of scopes and not others.

Note During failover, clients can sometimes obtain leases whose expiration times are shorter than the amount configured. This is a normal part of keeping the server partners synchronized. Typically this happens only for the first lease period, or during COMMUNICATIONS-INTERRUPTED state.

See Also

Server and Scope Backup Percentages
BOOTP Backup Percentage

Server and Scope Backup Percentages

For all servers or scopes for which you enable failover, you must set the *backup-pct* attribute. This is the number of currently available (unreserved) leases that the backup server can use for allocations to new DHCP clients when the main server is down. You can use the preset value, which is 10 percent, or specify another value.

Note When failover load balancing is in effect, the main and backup servers actively move available leases between them to maintain the backup percentage of available leases. See the "Setting Load Balancing" section on page 27-19.

BOOTP Backup Percentage

For scopes for which you enable dynamic BOOTP, use the *dynamic-bootp-backup-pct* attribute rather than the *backup-pct* attribute for the failover pair. The *dynamic-bootp-backup-pct* is the percentage of available addresses that the main server should send to the backup server for use with BOOTP clients.

The *dynamic-bootp-backup-pct* is distinct from the *backup-pct* attribute, because if you enable BOOTP on a scope, a server, even in PARTNER-DOWN state, never grants leases on addresses that are available to the other server. Network Registrar does not grant leases because the partner might give them out using dynamic BOOTP, and you can never safely assume that they are available again.

Note You must define the dynamic BOOTP backup percentage on the main server. If you define it on the backup server, Network Registrar ignores it (to enable duplicating configuration through scripts). If you do not define it, Network Registrar uses the default *backup-pct* for the failover pair or scope.

To properly support dynamic BOOTP while using the failover protocol, do this on every LAN segment in which you want BOOTP support:

- Create one scope for dynamic BOOTP.
- Enable BOOTP and dynamic BOOTP.
- Disable DHCP for that scope.

Setting Backup Allocation Boundaries

You can be more specific as to which addresses to allocate to the backup server by using the *failover-backup-allocation-boundary* attribute on the scope. The IP address set as this value is the upper boundary of addresses from which to allocate addresses to a backup server. Only addressees below this boundary are allocated to the backup. If there are none available below this boundary, then the addresses above it, if any, are allocated to the backup. The actual allocation works down from this address, while the normal allocation for DHCP clients works up from the lowest address in the scope.

If you set *failover-backup-allocation-boundary* for the scope, you must also enable the *allocate-first-available* attribute. If *failover-backup-allocation-boundary* is unset or set to zero, then the boundary used is halfway between the first and last addresses in the scope ranges. If there are no available addresses below this boundary, then the first available address is used.

Setting the Maximum Client Lead Time

You can set a property for failover that controls an adjustment to the lease period, the maximum client lead time (MCLT). The MCLT adjusts for a potential period of uncertain connectivity between the servers. It is the maximum time one server can grant (or extend) a lease to a client without first negotiating a longer time with its partner. This time has the following implications:

- Clients may initially (or if the partners are not communicating) only receive leases of the MCLT length. This means that they need to renew leases sooner than they might otherwise without failover. At this renewal, the client should get a full lease time (unless the partners are not communicating).
- If a server enters PARTNER-DOWN state, it must wait until the MCLT after the down time before it can fully take over all leases.
- If a failover recovery occurs where there is uncertainty about what one partner did (such as when it loses its lease database), the partners may have to restrict leasing activity for the MCLT period after they synchronize before they can resume normal failover operations.

The default MCLT is one hour, the optimum for most configurations. As defined by the failover protocol, the lease period given a client can never be more than the MCLT plus the most recently received potential expiration time from the failover partner, or the current time, whichever is later. That is why you sometimes see the initial lease period as only an hour, or an hour longer than expected for renewals. The actual lease time is recalculated when the main server comes back.

The MCLT is necessary because of failover use of lazy updates. Using lazy updates, the server can issue or renew leases to clients before updating its partner, which it can then do in batches of updates. If the server goes down and cannot communicate the lease information to its partner, the partner may try to reoffer the lease to another client based on what it last knew the expiration to be. The MCLT guarantees that there is an added window of opportunity for the client to renew. The way that a lease offer and renewal works with the MCLT is:

1. The client sends a DHCPDISCOVER to the server, requesting a desired lease period (say, three days). The server responds with a DHCPOFFER with an initial lease period of only the MCLT (one hour by default). The client then requests the MCLT lease period and the server acknowledges it.
2. The server sends its partner a bind update containing the lease expiration for the client as the current time plus the MCLT. The update also includes the potential expiration time as the current time plus the client desired period plus the MCLT (three days plus an hour). The partner acknowledges the potential expiration, thereby guaranteeing the transaction.
3. When the client sends a renewal request halfway through its lease (in one-half hour), the server acknowledges with the client desired lease period (three days). The server then updates its partner with the lease expiration as the current time plus the desired lease period (three days), and the

potential expiration as the current time plus the desired period and another half of this period (3 + 1.5 = 4.5 days). The partner acknowledges this potential expiration of 4.5 days. In this way, the main server tries to have its partner always lead the client in its understanding of the client lease period so that it can always offer it to the client.

There is no one correct value for the MCLT. There is an explicit trade-off between various factors in choosing one. Most people use the preset value of one hour effectively and it works well in almost all environments. Here are some of the trade-offs between a short and long MCLT:

- **Short MCLT**—A short MCLT value means that after entering PARTNER-DOWN state, a server only has to wait a short time before it can start allocating its partner IP addresses to DHCP clients. Furthermore, it only has to wait a short time after a lease expires before it can reallocate that address to another DHCP client. However, the down side is that the initial lease interval that is offered to every new DHCP client will be short, which causes increased traffic, because those clients need to send their first renewal in a half of a short MCLT time. Also, the lease extensions that a server in COMMUNICATIONS-INTERRUPTED state can give is the MCLT only after the server has been in that state for around the desired client lease period. If a server stays in that state for that long, then the leases it hands out will be short, increasing the load on that server, possibly causing difficulty.
- **Long MCLT**—A long MCLT value means that the initial lease period will be longer and the time that a server in COMMUNICATIONS-INTERRUPTED state can extend leases (after it being in that state for around the desired client lease period) will be longer. However, a server entering PARTNER-DOWN state must wait the longer MCLT before being able to allocate its partner addresses to new DHCP clients. This may mean that additional addresses are required to cover this time period. Also, the server in PARTNER-DOWN state must wait the longer MCLT from every lease expiration before it can reallocate an address to a different DHCP client.

Using the Failover Safe Period to Move Servers into PARTNER-DOWN State

One or both failover partners could potentially move into COMMUNICATIONS-INTERRUPTED state. Fortunately, they cannot issue duplicate addresses while in this state. However, having a server in this state over longer periods is not a good idea, because there are restrictions on what a server can do. The main server cannot reallocate expired leases and the backup server can run out of addresses from its pool. COMMUNICATIONS-INTERRUPTED state was designed for servers to easily survive transient communication failures of a few minutes to a few days. A server might function effectively in this state for only a short time, depending on the client arrival and departure rate. After that, it would be better to move a server into PARTNER-DOWN state so it can completely take over the lease functions until the servers resynchronize.

There are two ways a server can move into PARTNER-DOWN state:

- **User action**—An administrator sets a server into PARTNER-DOWN state based on an accurate assessment of reality. The failover protocol handles this correctly. Never set both partners to PARTNER-DOWN.
- **Failover safe period expires**—When the servers run unattended for longer periods, they need an automatic way to enter PARTNER-DOWN state.

Network operators might not sense in time that a server is down or uncommunicative. Hence, the failover safe period, which provides network operators some time to react to a server moving into COMMUNICATIONS-INTERRUPTED state. During the safe period, the only requirement is that the operators determine that both servers are still running and, if so, fix the network communications failure or take one of the servers down before the safe period expires.

During this safe period, either server allows renewals from any existing client, but there is a major risk of possibly issuing duplicate addresses. This is because one server can suddenly enter PARTNER-DOWN state while the other is still operating. Because of this risk, the failover safe period is disabled by default. That is why it is best to enable the safe period only if, during a server failure, it is more important to get an address than risk receiving a duplicate one.

The length of the safe period is installation-specific, and depends on the number of unallocated addresses in the pool and the expected arrival rate of previously unknown clients requiring addresses. The safe period is typically 24 hours, although many environments can support periods of several days.

The number of extra addresses required for the safe period should be the same as the expected total of new clients a server encounters. This depends on the arrival rate of new clients, not the total outstanding leases. Even if you can only afford a short safe period, because of a dearth of addresses or a high arrival rate of new clients, you can benefit substantially by allowing DHCP to ride through minor problems that are fixable in an hour. There is minimum chance of duplicate address allocation, and reintegration after the solved failure is automatic and requires no operator intervention.

Here are some guidelines to follow, to help you decide whether to use manual intervention or the safe period for transitioning to PARTNER-DOWN state:

- If your corporate policy is to have minimal manual intervention, set the safe period. Enable the failover pair attribute *use-safe-period* to enable the safe period. Then, set the DHCP attribute *safe-period* to set the duration (86400 seconds, or 24 hours, by default). Set this duration long enough so that operations personnel can explore the cause of the communication failure and assure that the partner is truly down. At least 12 hours is recommended.
- If your corporate policy is to avoid conflict under any circumstances, then never let the backup server go into PARTNER-DOWN state unless by explicit command. Allocate sufficient addresses to the backup server so that it can handle new client arrivals during periods when there is no administrative coverage. You can set PARTNER-DOWN on the View Failover Related Server page of the regional cluster web UI, if the partner is in the Communications-interrupted failover state, you can click **Set Partner Down** in association with an input field for the PARTNER-DOWN date setting. This setting is initialized to the value of the *start-of-communications-interrupted* attribute. (In Normal web UI mode, you cannot set this date to be an earlier value than the initialized date. In Expert web UI mode, you can set this value to any date.) After clicking **Set Partner Down**, you return to the List Related Servers for DHCP Server page to view the result of the PARTNER-DOWN action. Never set both partners to PARTNER-DOWN.

  Use **dhcp setPartnerDown** in the CLI, specifying the name of the partner server. This moves all the scopes running failover with the partner into PARTNER-DOWN state immediately, unless you specify a date and time with the command. This date and time should be when the partner was last known to be operational.

  There are two conventions for specifying the date:

  - *–num unit* (a time in the past), where *num* is a decimal number and *unit* is *s*, *m*, *h*, *d*, or *w* for seconds, minutes, hours, days or weeks respectively. For example, specify -3d for three days.
  - Month (name or its first three letters), day, hour (24-hour convention), year (fully specified year or last two digits). This example notifies the backup server that its main server went down at 12 o'clock midnight on October 31, 2002:

    ```
    nrcmd> server dhcp setPartnerDown dhcp2.example.com. -3d
    nrcmd> server dhcp setPartnerDown dhcp2.example.com. Oct 31 00:00:00 2001
    nrcmd> dhcp reload
    ```

Note Wherever you specify a date and time in the CLI, enter the time that is local to the **nrcmd** process. If the server is running in a different time zone than this process, disregard the time zone where the server is running and use local time instead.

Setting DHCP Request and Response Packet Buffers

The number of request buffers (set through the *max-dhcp-requests* DHCP attribute) sets the maximum number of simultaneous requests that the server can accept. The default value is 500, which is suitable for most deployments, but can be tuned to the capacity of the server. This capacity relates to the leasing rate and average latency of the leasing transaction. For example, if clients receive new leases every 250 milliseconds, then a request buffer value of 500 is sufficient for the server to respond to 2000 clients per second. (This assumes sufficient processing capacity to service clients at that rate.) A lower value would throttle the server performance below this capacity. A higher value allows servicing a greater number of clients without retries during periods of burst load, but results in a higher average latency to each client. Average latency under two seconds is sufficient to properly service clients.

The number of response buffers (set through the *max-dhcp-responses* DHCP attribute) sets the maximum number of simultaneous requests that the server can complete by issuing a client response. When the network operates at a steady state, responses should track with the number of requests accepted. Because the same pool of response buffers serves for both lease and failover activity, when failover is enabled, the server adjusts the response buffer value to be at least four times the request buffers. This ensures that sufficient resources are available to process all pending client and failover activity simultaneously.

Changing Polling Attributes

You can change some system defaults, such as the number of leases that the main server should send to the backup server, or the MCLT. See the "Setting the Maximum Client Lead Time" section on page 27-15. However, you need to change them on both servers.

On each server:

- **Change the poll interval (DHCP attribute *failover-poll-interval*)**—The interval that partners contact each other to confirm network connectivity. The preset value is 10 seconds.
- **Change the poll timeout (DHCP attribute *failover-poll-timeout*)**—Failover partners who cannot communicate during this timeout period will conclude that they lost network connectivity, and change their operational states appropriately. The preset value is 60 seconds.

  Generally, you should not have to change the *failover-poll-timeout*. It is intimately linked to the *failover-poll-interval* and is based on real world experience. Note that for failover load balancing, until the backup detects that the main server is down (after the *failover-poll-timeout* period), the backup discards any requests it receives from clients targeting the main server.

Note To collect subnet utilization history for the failover pair, if you are configuring simple failover, disable individual polling of the main and backup DHCP servers, but enable failover pair polling by setting the failover pair attribute *poll-subnet-util-interval*, so as to collect one set of data from both servers.

Setting the Network Discovery Attribute

If you enable failover on a UNIX system, you could set the *sms-network-discovery* attribute to enable the computing client os-type for leased addresses, which can help if you have a Windows partner server and want to use **dhcp updateSms** in the CLI on it.

Setting Load Balancing

In normal failover mode, the main DHCP server bears most of the burden of servicing clients when the failover partners are in normal communication mode. The main server not only services all new client requests, but has to handle renewal and rebinding requests and expired leases from the backup partner. To distribute the load more evenly between the two servers in a simple failover configuration scenario, Network Registrar introduced the load balancing feature (based on RFC 3074).

Failover load balancing allows both servers to actively service clients and determine which unique clients each will serve without running the risk of both servicing the same ones. Failover load balancing applies only while the servers are in NORMAL state; in other states, both servers can respond to clients. Per RFC 3074, the servers calculate a hash value for each request that the server receives based on the client identifier option value or hardware address, and the request is serviced if the hash value is assigned to that server. With failover load balancing enabled, the servers split the client load evenly (the main server processes 50% of the hash values, the backup the other 50%).

With load balancing, each partner responds to all clients whenever it is not in the NORMAL communications state. When they are in this communications state, they respond only to broadcast DHCPDISCOVER messages from clients that are in their assigned hash values. For broadcast DHCPREQUESTs, the server responds only if it is the targeted one (based on the server identifier option); so, if the targeted server is the main and it is down, the backup does not service the client (unless you release the lease). Broadcast BOOTP and DHCPINFORM requests are also load-balanced.

See Also

Load Balancing Compatibility with Earlier Network Registrar Versions
Configuring Load Balancing

Load Balancing Compatibility with Earlier Network Registrar Versions

Failover load balancing is disabled by default to ensure backward compatibility with earlier Network Registrar releases, and is used only if both servers support load balancing. Hence, the failover pair load balancing is unset and the default value of disabled applies until you explicitly enable it. If you enable load balancing, each server services about 50% of the clients, and the free leases given to the backup will be 50%, regardless of the configured percentage (see the "Setting Backup Percentages" section on page 27-13).

Configuring Load Balancing

In the web UI, when setting the failover properties for the pair (see the "Creating and Synchronizing Failover Server Pairs" section on page 27-4), enable or disable the *load-balancing* attribute in the Failover Settings attributes as desired to enable or disable failover load balancing. In the CLI, use **failover-pair** *name* **set load-balancing**.

Changing Failover Server Roles

Caution Be careful when you change the role of a failover server. Remember that all address states in a scope are lost from a server if it is ever reloaded without that scope in its configuration.

See Also

Making Nonfailover Servers Failover Mains
Replacing Servers Having Defective Storage, page 27-21
Removing Backup Servers and Halting Failover Operation, page 27-21
Adding Main Servers to Existing Backup Servers, page 27-21
Configuring Failover on Multiple Interface Hosts, page 27-22

Making Nonfailover Servers Failover Mains

You can update an existing installation and increase the availability of the DHCP service it offers. You can use this procedure only if the original server never participated in failover.

Step 1 Install Network Registrar on the original server and ensure that it operates correctly after the installation.

Step 2 Install Network Registrar on the machine that is to be the backup server. Note the machine DNS name.

Step 3 Enable failover on the original server. Use the DNS name of the recently installed backup server. See the "Simple Failover" section on page 27-1.

Step 4 Reload the main server. It should go into PARTNER-DOWN state. It cannot locate the backup server, because it is not yet configured. There should be no change in main server operation at this point.

Step 5 Duplicate the main server configuration on the backup server, including scopes (including secondary), policies, and client-classes. If you use client-classes, make sure the clients are entered into each cluster or that each server can access an LDAP database with the client data.

Step 6 Enable failover on the backup server. Be sure to define the main server.

Step 7 Reconfigure all operational BOOTP relays to forward broadcast packets to the main and backup server.

Step 8 Reload the backup server.

After you complete these steps:

1. The backup server detects the main server and moves into RECOVER state.
2. The backup server refreshes its stable storage with the main server lease data and, when complete, moves into RECOVER-DONE state.
3. The main server moves into NORMAL state.
4. The backup server moves into NORMAL state.
5. The backup server uses a pool request to ask the main server for addresses to allocate if communication is interrupted.
6. After allocating these addresses, the main server sends this data to the backup server.

Replacing Servers Having Defective Storage

If a failover server loses its stable storage (hard disk), you can replace the server and have it recover its state information from its partner.

Step 1 Determine which server lost its stable storage.

Step 2 Use **dhcp setPartnerDown** in the CLI to tell the other server that its partner is down. If you do not specify a time, the current time is used.

Step 3 When the server is again operational, reinstall Network Registrar.

Step 4 Duplicate the server configuration from its partner. However, do not recover any lease databases from an earlier backup or the partner system.

Step 5 Reload the replacement server.

After you complete these steps:

1. The recovered server moves into RECOVER state.
2. Its partner sends it all its data.
3. The server moves into RECOVER-DONE state when it reaches it maximum client lead time (and any time set for *failover-recover*).
4. Its partner moves into NORMAL state.
5. The recovered server moves into NORMAL state. It can request addresses, but can allocate few new ones, because its partner already sent it all its previously allocated addresses.

Removing Backup Servers and Halting Failover Operation

Sometimes you might need to remove the backup server and halt all failover operations.

Step 1 On the backup server, remove all the scopes that were designated as a backup to the main server.

Step 2 On the main server, remove the failover capability from those scopes that were main for the backup server, or disable failover server-wide if that is how it was configured.

Step 3 Reload both servers.

Adding Main Servers to Existing Backup Servers

You can use an existing backup server for a main server.

Step 1 Duplicate the main server scopes, policies, and other configurations on the backup server.

Step 2 Configure the main server to enable failover and point to the backup server.

Step 3 Configure the backup server to enable failover for the new scopes that point to the new main server.

Step 4 Reload both servers. Network Registrar performs the same steps as those described in the "Making Nonfailover Servers Failover Mains" section on page 27-20.

Configuring Failover on Multiple Interface Hosts

If you plan to use failover on a server host with multiple interfaces, you must explicitly configure the local server name or address. This requires an additional command. For example, if you have a host with two interfaces, serverA and serverB, and you want to make serverA the a main failover server, you must define serverA as the failover-main-server before you set the backup server name (external serverB). If you do not do this, failover might not initialize correctly and tries to use the wrong interface.

Set the DHCP server properties *failover-main-server* and *failover-backup-server.*

With multiple interfaces on one host, you must specify a hostname that points to only one address or A record. You cannot set up your servers for round-robin support.

Supporting BOOTP Clients in Failover

You can configure scopes to support two types of BOOTP clients—static and dynamic.

See Also

Static BOOTP
Dynamic BOOTP
Configuring BOOTP Relays, page 27-23

Static BOOTP

You can support static BOOTP clients using DHCP reservations. When you enable failover, remember to configure both the main and the backup server with identical reservations.

Dynamic BOOTP

You can enable dynamic BOOTP clients by enabling the *dynamic-bootp* attribute on a scope. When using failover, however, there are additional restrictions on address usage in such scopes, because BOOTP clients get permanent addresses and leases that never expire.

When a server whose scope does not have the *dynamic-bootp* option enabled goes to PARTNER-DOWN state, it can allocate any available (unassigned) address from that scope, whether or not it was initially available to any partner. However, when the *dynamic-bootp* option is set, each partner can only allocate its own addresses. Consequently, scopes that enable the *dynamic-bootp* option require more addresses to support failover.

When using dynamic BOOTP:

- Segregate dynamic BOOTP clients to a single scope. Disable DHCP clients from using that scope by disabling the *dhcp* attribute on the scope.
- Set the *dynamic-bootp-backup-pct* failover pair attribute to allocate a greater percentage of addresses to the backup server for this scope, as much as 50 percent higher than a regular backup percentage.

Configuring BOOTP Relays

The Network Registrar failover protocol works with BOOTP relay (also called IP helper), a router capability that supports DHCP clients that are not locally connected to a server.

If you use BOOTP relay, ensure that the implementations point to both the main and backup servers. If they do not and the main fails, clients are not serviced, because the backup cannot see the required packets. If you cannot configure BOOTP relay to forward broadcast packets to two different servers, configure the router to forward the packets to a subnet-local broadcast address for a LAN segment, which could contain both the main and backup servers. Then, ensure that both the main and backup servers are on the same LAN segment.

DHCPLEASEQUERY and Failover

To accommodate DHCPLEASEQUERY messages sent to a DHCP failover backup server when the master server is down, the master server must communicate the *relay-agent-info* (82) option values to its partner server. To accomplish this, the master server uses DHCP failover update messages.

Troubleshooting Failover

This section describes how to avoid failover configuration mistakes, monitor failover operations, and detect and handle network problems.

See Also

Monitoring Failover Operations
Detecting and Handling Network Failures, page 27-24

Monitoring Failover Operations

You can examine the DHCP server log files on both partner servers to verify your failover configuration.

You can make a few important log and debug settings to troubleshoot failover. Set the DHCP log settings to *failover-detail* to track the number and details of failover messages logged. To ensure that previous messages do not get overwritten, add the *failover-detail* attribute to the end of the list. Use the *no-failover-conflict* attribute to inhibit logging server failover conflicts, or the *no-failover-activity* attribute to inhibit logging normal server failover activity. Then, reload the server.

You can also isolate misconfigurations more easily by clicking the Related Servers icon () on the Manage DHCP Server or List DHCP Failover Pairs page, or by using **dhcp getRelatedServers** in the CLI.

Detecting and Handling Network Failures

Table 27-4 describes some symptoms, causes, and solutions for failover problems.

Table 27-4 Detecting and Handling Failures

Symptom	Cause	Solution
New clients cannot get addresses	A backup server is in COMMUNICATIONS-INTERRUPTED state with too few addresses	Increase the backup percentage on the main server.
Error messages about mismatched scopes	There are mismatched scope configurations between partners	Reconfigure your servers.
Log messages about failure to communicate with partner	Server cannot communicate with its partner	Check the status of the server.
Main server fails. Some clients cannot renew or rebind leases. The leases expire even when the backup server is up and possibly processing some client requests.	Some BOOTP relay (ip-helper) was not configured to point at both servers; see the "Configuring BOOTP Relays" section on page 27-23.	• Reconfigure BOOTP relays to point at both main and backup server • Run a fire drill test—Take the main server down for a day or so and see if your user community can get and renew leases
SNMP trap: other server not responding	Server cannot communicate with its partner	Check the status of the server.
SNMP trap: dhcp failover configuration mismatch	Mismatched scope configurations between partners	Reconfigure your servers.
Users complain that they cannot use services or system as expected	Mismatched policies and client-classes between partners	Reconfigure partners to have identical policies; possibly use LDAP for client registration if currently registering clients directly in partners.



CHAPTER 28

Configuring DNS Update

The DNS Update protocol (RFC 2136) integrates DNS with DHCP. The latter two protocols are complementary; DHCP centralizes and automates IP address allocation, while DNS automatically records the association between assigned addresses and hostnames. When you use DHCP with DNS update, this configures a host automatically for network access whenever it attaches to the IP network. You can locate and reach the host using its unique DNS hostname. Mobile hosts, for example, can move freely without user or administrator intervention.

This chapter explains how to use DNS update with Cisco Network Registrar servers, and its special relevance to Windows client systems.

See Also

DNS Update Process, page 28-2
Special DNS Update Considerations, page 28-2
DNS Update for DHCPv6, page 28-2
Creating DNS Update Configurations, page 28-5
Creating DNS Update Maps, page 28-7
Configuring Access Control Lists and Transaction Security, page 28-9
Configuring DNS Update Policies, page 28-13
Confirming Dynamic Records, page 28-17
Scavenging Dynamic Records, page 28-18
Troubleshooting DNS Update, page 28-19
Configuring DNS Update for Windows Clients, page 28-20

DNS Update Process

To configure DNS updates, you must:

1. Create a DNS update configuration for a forward or reverse zone or both. See the "Creating DNS Update Configurations" section on page 28-5.
2. Use this DNS update configuration in either of two ways:
   - Specify the DNS update configuration on a named, embedded, or default DHCP policy. See the "Creating and Applying DHCP Policies" section on page 21-3.

     Note Upgrading to Network Registrar 7.0 from pre-6.2 migrates the scope DNS update configuration attributes to a DNS update configuration object.

   - Define a DNS update map to autoconfigure a single DNS update relationship between a Network Registrar DHCP server or failover pair and a DNS server or High-Availability (HA) pair. Specify the update configuration in the DNS update map. See the "Creating DNS Update Maps" section on page 28-7.
3. Optionally define access control lists (ACLs) or transaction signatures (TSIGs) for the DNS update. See the "Configuring Access Control Lists and Transaction Security" section on page 28-9.
4. Optionally create one or more DNS update policies based on these ACLs or TSIGs and apply them to the zones. See the "Configuring DNS Update Policies" section on page 28-13.
5. Adjust the DNS update configuration for Windows clients, if necessary; for example, for dual zone updates. See the "Configuring DNS Update for Windows Clients" section on page 28-20.
6. Configure DHCP clients to supply hostnames or request that Network Registrar generate them.
7. Reload the DHCP and DNS servers, if necessary based on the edit mode.

Special DNS Update Considerations

Consider these two issues when configuring DNS updates:

- For security purposes, the Network Registrar DNS update process does not modify or delete a name an administrator manually enters in the DNS database.
- If you enable DNS update for large deployments, and you are not using HA DNS (see Chapter 18, "Configuring High-Availability DNS Servers"), divide primary DNS and DHCP servers across multiple clusters. DNS update generates an additional load on the servers.

DNS Update for DHCPv6

Network Registrar currently supports DHCPv6 DNS update over IPv4 only. For DHCPv6, DNS update applies to nontemporary stateful addresses only, not delegated prefixes.

DNS update for DHCPv6 involves AAAA and PTR RR mappings for leases. Network Registrar 7.0 supports server- or extension-synthesizing fully qualified domain names and the DHCPv6 *client-fqdn* option (39). Because Network Registrar is compliant with RFCs 4701, 4703, and 4704, it supports the

DHCID resource record (RR). All RFC-403-compliant updaters can generate DHCID RRs and result in data that is a hash of the client identifier (DUID) and the FQDN (per RFC 4701). Nevertheless, you can use AAAA and DHCID RRs in update policy rules.

DNS update processing for DHCPv6 is similar to that for DHCPv4 except that a single FQDN can have more than one lease, resulting in multiple AAAA and PTR RRs for a single client. The multiple AAAA RRs can be under the same name or a different name; however, PTR RRs are always under a different name, based on the lease address. RFC-4703-compliant updaters use the DHCID RR to avoid collisions among multiple clients.

Note Because DHCPv4 uses TXT RRs and DHCPv6 uses DHCID RRs for DNS update, to avoid conflicts, dual-stack clients cannot use single forward FQDNs. These conflicts primarily apply to client-requested names and not generated names, which are generally unique. To avoid these conflicts, use different zones for the DHCPv4 and DHCPv6 names.

See Also

DHCPv6 Upgrade Considerations
Generating Synthetic Names in DHCPv6
Determining Reverse Zones for DNS Updates, page 28-4
Using the Client FQDN, page 28-5

DHCPv6 Upgrade Considerations

If you use any policy configured prior to Network Registrar 7.0 that references a DNS update object for DHCPv6 processing (see the "DHCPv6 Policy Hierarchy" section on page 26-7), after the upgrade, the server begins queuing DNS updates to the specified DNS server or servers. This means that DNS updates might automatically (and unexpectedly) start for DHCPv6 leases.

Caution If you use earlier versions of Network Registrar or other DNS servers, you might experience interoperability issues for zone transfers and DNS updates, because of recent DHCID RR standards changes. You might need to upgrade DNS servers to support DHCPv6 DNS updates.

Generating Synthetic Names in DHCPv6

If clients do not supply hostnames, DHCPv6 includes a synthetic name generator. Because a DHCPv6 client can have multiple leases, Network Registrar uses a different mechanism than that for DHCPv4 to generate unique hostnames. The *v6-synthetic-name-generator* attribute for the DNS update configuration allows appending a generated name to the *synthetic-name-stem* based on the:

- Hash of the client DHCP Unique Identifier (DUID) value (the preset value).
- Raw client DUID value (as a hex string with no separators).
- CableLabs *cablelabs-17* option *device-id* suboption value (as a hex string with no separators, or the hash of the client DUID if not found).
- CableLabs *cablelabs-17* option *cm-mac-address* suboption value (as a hex string with no separators, or the hash of the client DUID if not found).

Caution Some generation methods might cause privacy issues if the domain is accessible from the Internet.

See the "Creating DNS Update Configurations" section on page 28-5 for how to create a DNS update configuration with synthetic name generation.

In the CLI, an example of this setting is:

```
nrcmd> dhcp-dns-update example-update-config set v6-synthetic-name-generator=hashed-duid
```

Determining Reverse Zones for DNS Updates

The DNS update configuration uses the prefix length value in the specified *reverse-zone-prefix-length* attribute to generate a reverse zone in the ip6.arpa domain. You do not need to specify the full reverse zone, because you can synthesize it by using the ip6.arpa domain. You set this attribute for the reverse DNS update configuration (see the "Creating DNS Update Configurations" section on page 28-5). Here are some rules for *reverse-zone-prefix-length*:

- Use a multiple of 4 for the value, because ip6.arpa zones are on 4-bit boundaries. If not a multiple of 4, the value is rounded up to the next multiple of 4.
- The maximum value is 124, because specifying 128 would create a zone name without any possible hostnames contained therein.
- A value of 0 means none of the bits are used for the zone name, hence ip6.arpa is used.
- If you omit the value from the DNS update configuration, the server uses the value from the prefix or, as a last resort, the prefix length derived from the *address* value of the prefix (see the "Configuring Prefixes" section on page 26-14).

Note that to synthesize the reverse zone name, the *synthesize-reverse-zone* attribute must remain enabled for the DHCP server. Thus, the order in which a reverse zone name is synthesized for DHCPv6 is by:

1. Using the full *reverse-zone-name* in the reverse DNS update configuration.
2. Basing it on the ip6.arpa zone from the *reverse-zone-prefix-length* in the reverse DNS update configuration.
3. Basing it on the ip6.arpa zone from the *reverse-zone-prefix-length* in the prefix definition.
4. Basing it on the ip6.arpa zone from the prefix length for the *address* in the prefix definition.

In the CLI, an example of setting the reverse zone prefix length is:

```
nrcmd> dhcp-dns-update example-update-config set reverse-zone-prefix-length=32
```

To create a reverse zone for a prefix in the web UI, the List/Add Prefixes page includes a **Create Reverse Zone** button for each prefix. (See the "Creating and Editing Prefixes" section on page 26-18.)

The CLI also provides the **prefix** *name* **createReverseZone** [**–range**] command to create a reverse zone for a prefix (from its address or range value). Delete the reverse zone by using **prefix** *name* **deleteReverseZone** [**–range**].

You can also create a reverse zone from a DHCPv4 subnet or DHCPv6 prefix by entering the subnet or prefix value when directly configuring the reverse zone. See the "Adding Primary Reverse Zones" section on page 15-12 for details.

Using the Client FQDN

The existing DHCP server *use-client-fqdn* attribute controls whether the server pays attention to the DHCPv6 client FQDN option in the request. The rules that the server uses to determine which name to return when multiple names exist for a client are in the following order of preference:

1. The server FQDN that uses the client requested FQDN if it is in use for any lease (even if not considered to be in DNS).
2. The FQDN with the longest valid lifetime considered to be in DNS.
3. The FQDN with the longest valid lifetime that is not yet considered to be in DNS.

Creating DNS Update Configurations

A DNS update configuration defines the DHCP server framework for DNS updates to a DNS server or HA DNS server pair. It determines if you want to generate forward or reverse zone DNS updates (or both). It optionally sets TSIG keys for the transaction, attributes to control the style of autogenerated hostnames, and the specific forward or reverse zone to be updated. You must specify a DNS update configuration for each unique server relationship.

For example, if all updates from the DHCP server are directed to a single DNS server, you can create a single DNS update configuration that is set on the server default policy. To assign each group of clients in a client-class to a corresponding forward zone, set the forward zone name for each in a more specific client-class policy.

Local Advanced Web UI

Step 1 Click **DHCP**, then **DNS** to open the List DNS Update Configurations page.

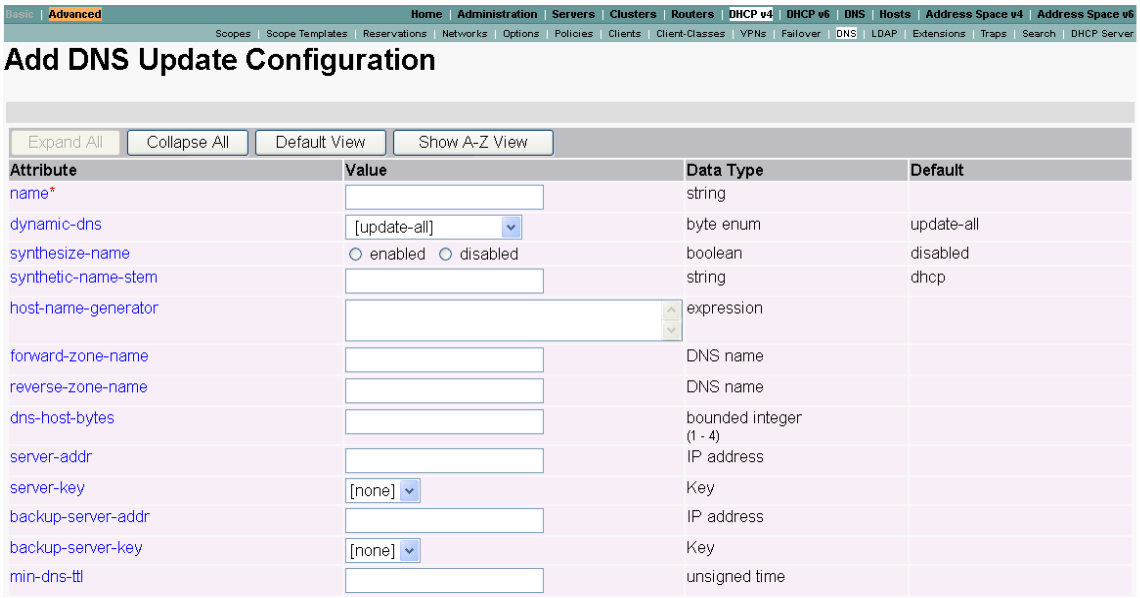
Step 2 Click **Add DNS Update Configuration** to open the Add DNS Update Configuration page (see Figure 28-1 on page 28-6 for a partial view).

Step 3 Enter a name for the update configuration in the *name* attribute field.

Step 4 Click the appropriate *dynamic-dns* setting:

- **update-none**—Do not update forward or reverse zones.
- **update-all**—Update forward and reverse zones (the default value).
- **update-fwd-only**—Update forward zones only.
- **update-reverse-only**—Update reverse zones only.

Figure 28-1 Add DNS Update Configuration Page (Local Advanced)



Step 5 Set the other attributes appropriately:

a. If necessary, enable *synthesize-name* and set the *synthetic-name-stem* value.

You can set the stem of the default hostname to use if clients do not supply hostnames, by using *synthetic-name-stem*. For DHCPv4, enable the *synthesize-name* attribute to trigger the DHCP server to synthesize unique names for clients based on the value of the *synthetic-name-stem*. The resulting name is the name stem appended with the hyphenated IP address. For example, if you specify a *synthetic-name-stem* of **host** for address 192.168.50.1 in the example.com domain, and enable the *synthesize-name* attribute, the resulting hostname is host-192-168-50-1.example.com. The preset value for the synthetic name stem is **dhcp**.

The *synthetic-name-stem* must:

- Be a relative name without a trailing dot.
- Include alphanumeric values and hyphens (**–**) only. Space characters and underscores become hyphens and other characters are removed.
- Include no leading or trailing hyphen characters.
- Have DNS hostnames of no more than 63 characters per label and 255 characters in their entirety. The algorithm uses the configured forward zone name to determine the number of available characters for the hostname, and truncates the end of the last label if necessary.

For DHCPv6, see the "Generating Synthetic Names in DHCPv6" section on page 28-3.

b. Set *forward-zone-name* to the forward zone, if updating forward zones. Note that the policy *forward-zone-name* takes precedence over the one set in the DNS update configuration.

For DHCPv6, the server ignores the client and client-class policies when searching for a *forward-zone-name* value in the policy hierarchy. The search for a forward zone name begins with the prefix embedded policy.

c. For DHCPv4, set *reverse-zone-name* to the reverse (in.addr.arpa) zone to be updated with PTR and TXT records. If unset and the DHCP server *synthesize-reverse-zone* attribute is enabled, the server synthesizes a reverse zone name based on the address of each lease, scope subnet number, and DNS update configuration (or scope) *dns-host-bytes* attribute value.

The *dns-host-bytes* value controls the split between the host and zone parts of the reverse zone name. The value sets the number of bytes from the lease IP address to use for the hostname; the remaining bytes are used for the in-addr.arpa zone name. A value of 1 means use just one byte for the host part of the domain and the other three from the domain name (reversed). A value of 4 means use all four bytes for the host part of the address, thus using just the in-addr.arpa part of the domain. If unset, the server synthesizes an appropriate value based on the scope subnet size, or if the *reverse-zone-name* is defined, calculates the host bytes from this name.

For DHCPv6, see the "Determining Reverse Zones for DNS Updates" section on page 28-4.

d. Set *server-addr* to the IP address of the primary DNS server for the forward zone (or reverse zone if updating reverse zones only.

e. Set *server-key* and *backup-server-key* if you are using a TSIG key to process all DNS updates (see the "Transaction Security" section on page 28-11).

f. Set *backup-server-addr* to the IP address of the backup DNS server, if HA DNS is configured.

g. If necessary, enable or disable *update-dns-first* (preset value disabled) or *update-dns-for-bootp* (preset value enabled). The *update-dns-first* setting controls whether DHCP updates DNS before granting a lease. Enabling this attribute is not recommended.

Step 6 At the regional level, you can also push update configurations to the local clusters, or pull them from the replica database on the List DNS Update Configurations page.

Step 7 Click **Add DNS Update Configuration**.

Step 8 To specify this DNS update configuration on a policy, see the "Creating and Applying DHCP Policies" section on page 21-3.

CLI Commands

Use **dhcp-dns-update** *name* **create**. For example:

```
nrcmd> dhcp-dns-update example-update-config create
```

Set the *dynamic-dns* attribute to its appropriate value (update-none, update-all, update-fwd-only, or update-reverse-only). For example:

```
nrcmd> dhcp-dns-update example-update-config set dynamic-dns=update-all
```

See Also

DNS Update Process, page 28-2
Special DNS Update Considerations, page 28-2
DNS Update for DHCPv6, page 28-2

Creating DNS Update Maps

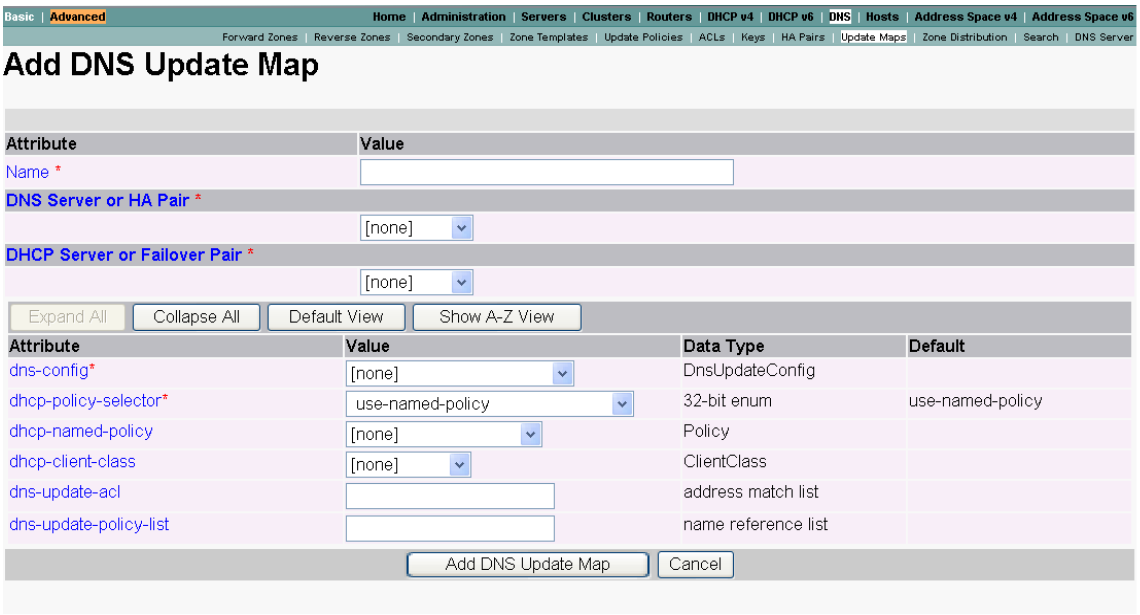
A DNS update map facilitates configuring DNS updates so that the update properties are synchronized between HA DNS server pairs or DHCP failover server pairs, based on an update configuration, so as to reduce redundant data entry. The update map applies to all the primary zones that the DNS pairs service, or all the scopes that the DHCP pairs service. You must specify a policy for the update map. To use this function, you must be an administrator assigned the server-management subrole of the dns-management or central-dns-management role, and the dhcp-management role (for update configurations).

Web UI

Step 1 Click **Advanced**, then **DNS**, then **Update Maps** to open the List DNS Update Maps page.

Step 2 Click **Add DNS Update Map** to open the Add DNS Update Map page (see Figure 28-2).

Figure 28-2 Add DNS Update Map Page (Local Advanced)



Step 3 Enter a name for the update map in the Name field.

Step 4 Enter the DNS update configuration from the previous section in the *dns-config* field.

Step 5 Set the kind of policy selection you want for the *dhcp-policy-selector* attribute. The choices are:

- **use-named-policy**—Use the named policy set for the *dhcp-named-policy* attribute (the preset value).
- **use-client-class-embedded-policy**—Use the embedded policy from the client-class set for the *dhcp-client-class* attribute.
- **use-scope-embedded-policy**—Use the embedded policy from the scope.

Step 6 If using update ACLs (see the "Configuring Access Control Lists and Transaction Security" section on page 28-9) or DNS update policies (see the "Configuring DNS Update Policies" section on page 28-13), set either the *dns-update-acl* or *dns-update-policy-list* attribute. Either value can be one or more addresses separated by commas. The *dns-update-acl* takes precedence over the *dns-update-policy-list*.

If you omit both values, a simple update ACL is constructed whereby only the specified DHCP servers or failover pair can perform updates, along with any *server-key* value set in the update configuration specified for the *dns-config* attribute.

Step 7 Click **Add DNS Update Map**.

Step 8 You can also push update maps to the local clusters, or pull them from the replica database on the List DNS Update Maps page.

CLI Commands

Specify the name, cluster of the DHCP and DNS servers (or DHCP failover or HA DNS server pair), and the DNS update configuration when you create the update map, using **dns-update-map** *name* **create** *dhcp-cluster dns-cluster dns-config*. For example:

```
nrcmd> dns-update-map example-update-map create Example-cluster Boston-cluster
    example-update-config
```

Set the *dhcp-policy-selector* attribute value to use-named-policy, use-client-class-embedded-policy, or use-scope-embedded-policy. If using the use-named-policy value, also set the *dhcp-named-policy* attribute value. For example:

```
nrcmd> dns-update-map example-update-map set dhcp-policy-selector=use-named-policy
nrcmd> dns-update-map example-update-map set dhcp-named-policy=example-policy
```

Configuring Access Control Lists and Transaction Security

ACLs are authorization lists, while transaction signatures (TSIG) is an authentication mechanism:

- ACLs enable the server to allow or disallow the request or action defined in a packet.
- TSIG ensures that DNS messages come from a trusted source and are not tampered with.

For each DNS query, update, or zone transfer that is to be secured, you must set up an ACL to provide permission control. TSIG processing is performed only on messages that contain TSIG information. A message that does not contain, or is stripped of, this information bypasses the authentication process.

For a totally secure solution, messages should be authorized by the same authentication key. For example, if the DHCP server is configured to use TSIG for DNS updates and the same TSIG key is included in the ACL for the zones to be updated, then any packet that does not contain TSIG information fails the authorization step. This secures the update transactions and ensures that messages are both authenticated and authorized before making zone changes.

ACLs and TSIG play a role in setting up DNS update policies for the server or zones, as described in the "Configuring DNS Update Policies" section on page 28-13.

See Also

Access Control Lists
Configuring Zones for Access Control Lists, page 28-10
Transaction Security, page 28-11

Access Control Lists

You assign ACLs on the DNS server or zone level. ACLs can include one or more of these elements:

- **IP address**—In dotted decimal notation; for example, 192.168.1.2.
- **Network address**—In dotted decimal and slash notation; for example, 192.168.0.0/24. In this example, only hosts on that network can update the DNS server.

- **Another ACL**—Must be predefined. You cannot delete an ACL that is embedded in another one until you remove the embedded relationship. You should not delete an ACL until all references to that ACL are deleted.
- **Transaction Signature (TSIG) key**—The value must be in the form **key** *value*, with the keyword **key** followed by the secret value. To accommodate space characters, the entire list must be enclosed in double quotes. For TSIG keys, see the "Transaction Security" section on page 28-11.

You assign each ACL a unique name. However, the following ACL names have special meanings and you cannot use them for regular ACL names:

- **any**—Anyone can perform a certain action
- **none**—No one can perform a certain action
- **localhost**—Any of the local host addresses can perform a certain action
- **localnets**—Any of the local networks can perform a certain action

Note the following:

- If an ACL is not configured, **any** is assumed.
- If an ACL is configured, at least one clause must allow traffic.
- The negation operator (**!**) disallows traffic for the object it precedes, but it does not intrinsically allow anything else unless you also explicitly specify it. For example, to disallow traffic for subnet 192.168.50.0 only, use **!192.168.50.0, any**.

Local Advanced Web UI

Click **DNS**, then **ACLs** to open the List/Add Access Control Lists page. Add an ACL name and match list. Note that a **key** *value* pair should not be in quotes. At the regional level, you can additionally pull replica ACLs or push ACLs to local clusters.

CLI Commands

Use **acl** *name* **create** *match-list*, which takes a name and one or more ACL elements. The ACL list is comma-separated, with double quotes surrounding it if there is a space character. The CLI does not provide the pull/push function.

For example, the following commands create three ACLs. The first is a key with a value, the second is for a network, and the third points to the first ACL. Including an exclamation point (**!**) before a value negates that value, so that you can exclude it in a series of values:

```
nrcmd> acl sec-acl create "key h-a.h-b.example.com."
nrcmd> acl dyn-update-acl create "!192.168.2.13,192.168.2.0/24"
nrcmd> acl main-acl create sec-acl
```

Configuring Zones for Access Control Lists

To configure ACLs for the DNS server or zones, set up a DNS update policy, then define this update policy for the zone (see the "Configuring DNS Update Policies" section on page 28-13).

Transaction Security

Transaction Signatures (TSIG) enable the DNS server to authenticate each message that it receives. Communication between servers is not encrypted but it becomes digitally signed, which allows validation of the authenticity of the data and the source of the packet.

When you configure the Network Registrar DHCP server to use TSIG for DNS updates, the server appends a TSIG RR to the messages. Part of the TSIG record is a digital signature.

When the DNS server receives a message, it looks for the TSIG record. If it finds one, it first verifies that the key name in it is one of the keys it recognizes. It then verifies that the time stamp in the update is reasonable (to help fight against traffic replay attacks). Finally, the server looks up the key shared secret that was sent in the packet and calculates its own signature. If the resulting calculated signature matches the signature included in the packet, then the contents are considered to be authentic.

See Also

Creating TSIG Keys
Generating Keys, page 28-12
Considerations for Managing Keys, page 28-13
Adding Supporting TSIG Attributes, page 28-13

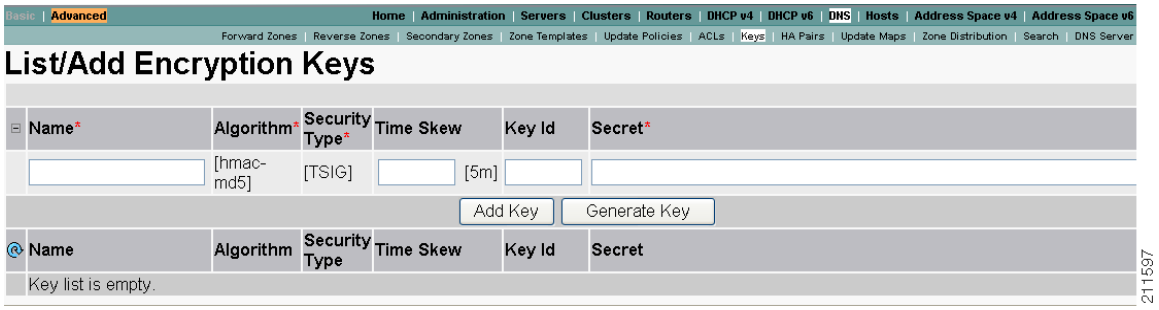
Creating TSIG Keys

Note If you want to enable key authentication for Address-to-User Lookup (ATUL) support, you must also define a key identifier (*id* attribute value). See the "Setting DHCP Forwarding" section on page 23-21.

Local Advanced Web UI

Click **Administration** or **DNS**, then **Keys**, to open the List/Add Encryption Keys page (see Figure 28-3).

Figure 28-3 List/Add Encryption Keys Page (Local Advanced)



For a description of the Algorithm, Security Type, Time Skew, Key ID, and Secret values, see Table 28-1 on page 28-12. See also the "Considerations for Managing Keys" section on page 28-13.

To edit a TSIG key, click its name on the List/Add Encryption Keys page to open the Edit Encryption Key page.

At the regional level, you can additionally pull replica keys, or push keys to local clusters.

CLI Commands

Use **key** *name* **create** *secret*. Provide a name for the key (in domain name format; for example, hosta-hostb-example.com.) and a minimum of the shared secret as a base-64 encoded string (see Table 28-1 on page 28-12 for a description of the optional time skew attribute). An example in the CLI would be:

```
nrcmd> key hosta-hostb-example.com. create secret-string
```

Generating Keys

It is recommended that you use the Network Registrar **cnr_keygen** utility to generate TSIG keys so that you add them or import them using **import keys**.

Execute the **cnr_keygen** key generator utility from a DOS prompt, or a Solaris or Linux shell:

- On Windows, the utility is in the *install-path*\bin folder.
- On Solaris and Linux, the utility is in the *install-path*/usrbin directory.

An example of its usage (on Solaris and Linux) is:

```
> /opt/nwreg2/local/usrbin/cnr_keygen -n a.b.example.com. -a hmac-md5 -t TSIG -b 16
    -s 300
        key "a.b.example.com." {
        algorithm hmac-md5;
        secret "xGVCsFZ0/6e0N97HGF50eg==";
        # cnr-time-skew 300;
        # cnr-security-type TSIG;
};
```

The only required input is the key name. The options are described in Table 28-1.

Table 28-1 Options for the cnr_keygen Utility

Option	Description
–n *name*	Key name. Required. The maximum length is 255 bytes.
–a hmac-md5	Algorithm. Optional. Only hmac-md5 is currently supported.
–b *bytes*	Byte size of the secret. Optional. The preset value is 16 bytes. The valid range is 1 through 64 bytes.
–s *skew*	Time skew for the key, in seconds. This is the maximum difference between the time stamp in packets signed with this key and the local system time. Optional. The preset value is 5 minutes. The range is one second through one hour.
–t tsig	Type of security used. Optional. Only TSIG is currently supported.
–h	Help. Optional. Displays the syntax and options of the utility.
–v	Version. Optional. Displays the version of the utility.

The resulting secret is base64-encoded as a random string.

You can also redirect the output to a file if you use the right-arrow (**>**) or double-right-arrow (**>>**) indicators at the end of the command line. The > writes or overwrites a given file, while the **>>** appends to an existing file. For example:

```
> /opt/nwreg2/local/usrbin/cnr_keygen -n example.com > keyfile.txt
> /opt/nwreg2/local/usrbin/cnr_keygen -n example.com >> addtokeyfile.txt
```

You can then import the key file into Network Registrar using the CLI to generate the keys in the file. The key import can generate as many keys as it finds in the import file. The path to the file should be fully qualified. For example:

```
nrcmd> import keys keydir/keyfile.txt
```

Considerations for Managing Keys

If you generate your own keys, you must enter them as a base64-encoded string. This means that the only characters allowed are those in the base64 alphabet and the equals sign (=) as pad character. Entering a nonbase64-encoded string results in an error message.

Here are some other suggestions:

- Do not add or modify keys using batch commands.
- Change shared secrets frequently; every two months is recommended. Note that Network Registrar does not explicitly enforce this.
- The shared secret length should be at least as long as the keyed message digest (HMAC-MD5 is 16 bytes). Note that Network Registrar does not explicitly enforce this and only checks that the shared secret is a valid base64-encoded string, but it is the policy recommended by RFC 2845.

Adding Supporting TSIG Attributes

To add TSIG support for a DNS update configuration (see the "Creating DNS Update Configurations" section on page 28-5), set these attributes:

- *server-key*
- *backup-server-key*

Configuring DNS Update Policies

DNS update policies provide a mechanism for managing update authorization at the RR level. Using update policies, you can grant or deny DNS updates based on rules that are based on ACLs as well as RR names and types. ACLs are described in the "Access Control Lists" section on page 28-9.

See Also

Compatibility with Previous Network Registrar Releases
Creating and Editing Update Policies, page 28-14
Defining and Applying Rules for Update Policies, page 28-14

Compatibility with Previous Network Registrar Releases

Previous Network Registrar releases used static RRs that administrators entered, but that DNS updates could not modify. This distinction between static and dynamic RRs no longer exists. RRs can now be marked as protected or unprotected (see the "Protecting Resource Record Sets" section on page 16-3). Administrators creating or modifying RRs can now specify whether RRs should be protected. A DNS update cannot modify a protected RR set, even if an RR of the given type does not yet exist in the set.

Note Previous releases allowed DNS updates only to A, TXT, PTR, CNAME and SRV records. This was changed to allow updates to all but SOA and NS records in unprotected name sets. To remain compatible with a previous release, use an update policy to limit RR updates.

Creating and Editing Update Policies

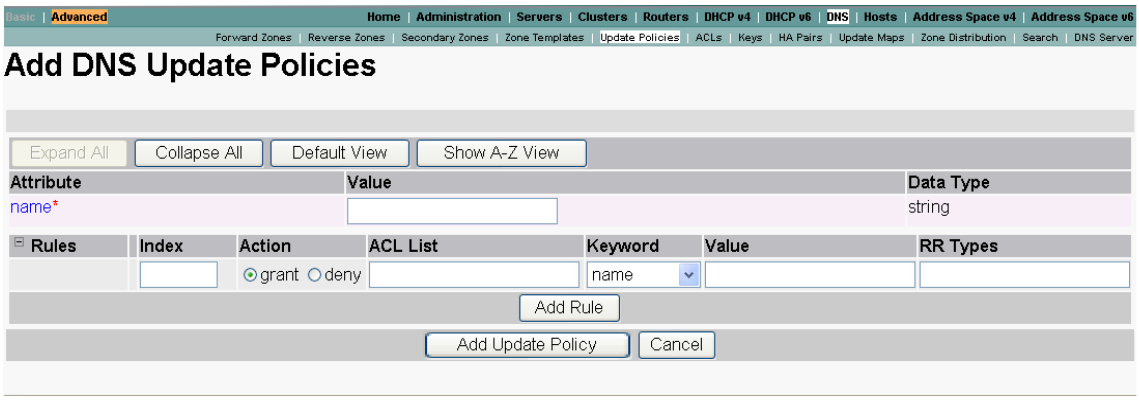
Creating an update policy initially involves creating a name for it.

Local Advanced Web UI

Step 1 Click **DNS**, then **Update Policies** to open the List DNS Update Policies page.

Step 2 Click **Add Policy** to open the Add DNS Update Policy page (see Figure 28-4).

Figure 28-4 Add DNS Update Policies Page (Local Advanced)



Step 3 Enter a name for the update policy.

Step 4 Proceed to the "Defining and Applying Rules for Update Policies" section.

CLI Commands

Use **update-policy** *name* **create**; for example:

```
nrcmd> update-policy policy1 create
```

Defining and Applying Rules for Update Policies

DNS update policies are effective only if you define rules for each that grant or deny updates for certain RRs based on an ACL. If no rule is satisfied, the default (last implicit) rule is to deny all updates (**"deny any wildcard * *"**).

See Also

Defining Rules for Named Update Policies
Applying Update Policies to Zones, page 28-17

Defining Rules for Named Update Policies

Defining rules for named update policies involves a series of Grant and Deny statements.

Local Advanced Web UI

Step 1 Create an update policy, as described in the "Creating and Editing Update Policies" section on page 28-14, or edit it.

Step 2 On the Add DNS Update Policies or Edit DNS Update Policy page:

a. Enter an optional value in the Index field.

b. Click Grant to grant the rule, or Deny to deny the rule.

c. Enter an access control list in the ACL List field.

d. Choose a keyword from the Keyword drop-down list.

e. Enter a value based on the keyword in the Value field. This can be a RR or subdomain name, or, if the **wildcard** keyword is used, it can contain wildcards (see Table 28-2).

Table 28-2 Wildcard Values for Update Policy Rules

Wildcard	Description
*	Matches zero or more characters. For example, the pattern **example*** matches all strings starting with *example*, including **example–**.
?	Matches a single character only. For example, the pattern **example?.com** matches **example1.com** and **example2.com**, but not **example.com**.
/[/]	Matches any characters in the (escaped) brackets; for example, **/[abc/]**. Each square bracket must be escaped using a slash (/). The characters can also be in a range; for example, **/[0–9/]** and **/[a–z/]**. If a pattern should include a hyphen, make the hyphen the first character; for example, **example/[–a–z/]**.

f. Enter one or more RR types, separated by commas, in the RR Types field, or use * for "all RRs." RRs can be of any of those described in Appendix A, "Resource Records." You can use negated values, which are values prefixed by an exclamation point; for example, **!PTR**.

g. Click **Add Policy**.

Step 3 At the regional level, you can also push update policies to the local clusters, or pull them from the replica database on the List DNS Update Policies page.

Step 4 To edit an update policy, click the name of the update policy on the List DNS Update Policies page to open the Edit DNS Update Policy page, make changes to the fields, then click **Edit Policy**.

CLI Commands

Create or edit an update policy (see the "Creating and Editing Update Policies" section on page 28-14, then use **update-policy** *name* **rules add** *rule*, with *rule* being the rule. (See Table 28-2 on page 28-15 for the rule wildcard values.) For example:

```
nrcmd> update-policy policy1 rules add "grant 192.168.50.101 name host1 A,TXT" 0
```

The rule is enclosed in quotes. To parse the rule syntax for the example:

- **grant**—Action that the server should take, either **grant** or **deny**.
- **192.168.50.101**—The ACL, in this case an IP address. The ACL can be one of the following:
  - Name—ACL created by name, as described in the "Access Control Lists" section on page 28-9.
  - IP address, as in the example.
  - Network address, including mask; for example, **192.168.50.0/24**.
  - TSIG key—Transaction signature key, in the form **key=***key*, (as described in the "Transaction Security" section on page 28-11.
  - One of the reserved words:

    any—Any ACL

    none—No ACL

    localhost—Any local host addresses

    localnets—Any local network address

  You can negate the ACL value by preceding it with an exclamation point (**!**).
- **name**—Keyword, or type of check to perform on the RR, which can be one of the following:
  - **name**—Name of the RR, requiring a name value.
  - **subdomain**—Name of the RR or the subdomain with any of its RRs, requiring a name or subdomain value.
  - **wildcard**—Name of the RR, using a wildcard value (see Table 28-2 on page 28-15).
- **host1**—Value based on the keyword, in this case the RR named host1. This can also be a subdomain name or, if the **wildcard** keyword is used, can contain wildcards (see Table 28-2 on page 28-15).
- **A,TXT**—RR types, each separated by a comma. This can be a list of any of the RR types described in Appendix A, "Resource Records." You can negate each record type value by preceding it with an exclamation point (**!**).
- Note that if this or any assigned rule is not satisfied, the default is to deny all RR updates.

Tacked onto the end of the rule, outside the quotes, is an index number, in the example, **0**. The index numbers start at 0. If there are multiple rules for an update policy, the index serves to add the rule in a specific order, such that lower numbered indexes have priority in the list. If a rule does not include an index, it is placed at the end of the list. Thus, a rule always has an index, whether or not it is explicitly defined. You also specify the index number in case you need to remove the rule.

To replace a rule, use **update-policy** *name* **delete**, then recreate the update policy. To edit a rule, use **update-policy** *name* **rules remove** *index*, where *index* is the explicitly defined or system-defined index number (remembering that the index numbering starts at 0), then recreate the rule. To remove the second rule in the previous example, enter:

```
nrcmd> update-policy policy1 rules remove 1
```

Applying Update Policies to Zones

After creating an update policy, you can apply it to a zone (forward and reverse) or zone template.

Local Advanced Web UI

Step 1 Click **DNS**, then **Forward Zones** to open the List/Add Forward Zones page.

Step 2 Click the name of the zone to open the Edit Forward Zone page.

Tip You can also perform this function for zone templates on the Edit Zone Template page, and primary reverse zones on the Edit Primary Reverse Zone page (see Chapter 15, "Managing Zones.").

Step 3 Enter the name or (comma-separated) names of one or more of the existing named update policies in the *update-policy-list* attribute field.

Note The server processes the *update-acl* before it processes the *update-policy-list*.

Step 4 Click **Modify Zone**.

CLI Commands

Use **zone** *name* **set update-policy-list**, equating the *update-policy-list* attribute with a quoted list of comma-separated update policies, as defined in the "Creating and Editing Update Policies" section on page 28-14. For example:

```
nrcmd> zone example.com set update-policy-list="policy1,policy2"
```

Confirming Dynamic Records

The Network Registrar DHCP server stores all pending DNS update data on disk. If the DHCP server cannot communicate with a DNS server, it periodically tests for re-established communication and submits all pending updates. This test typically occurs every 40 seconds.

Local Advanced Web UI

Click **DNS**, then **Forward Zones**. Click the View icon ( ) in the RRs column to open the List/Add DNS Server RRs for Zone page.

CLI Commands

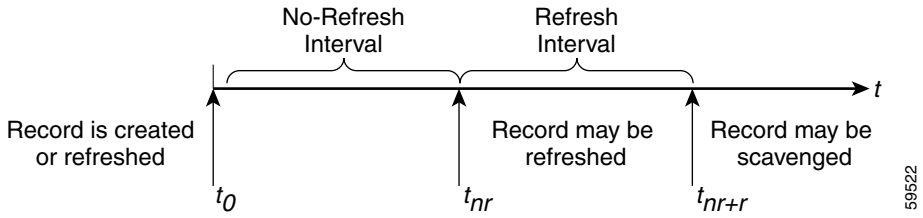
Use **zone** *name* **listRR dns**.

Scavenging Dynamic Records

Microsoft Windows 2003 and XP DNS clients that get DHCP leases can update (refresh) their Address (A) records directly with the DNS server. Because many of these clients are mobile laptops that are not permanently connected, some A records may become obsolete over time. The Windows DNS server scavenges and purges these primary zone records periodically. Network Registrar provides a similar feature that you can use to periodically purge stale records.

Scavenging is normally disabled by default, but you should enable it for zones that exclusively contain Windows clients. Zones are configured with *no-refresh* and *refresh* intervals. A record expires once it ages past its initial creation date plus these two intervals. Figure 28-5 shows the intervals in the scavenging time line.

Figure 28-5 Address Record Scavenging Time Line Intervals



The Network Registrar process is:

1. When the client updates the DNS server with a new A record, this record gets a timestamp, or if the client refreshes its A record, this may update the timestamp ("Record is created or refreshed").
2. During a no-refresh interval (a default value of seven days), if the client keeps sending the same record without an address change, this does not update the record timestamp.
3. Once the record ages past the no-refresh interval, it enters the refresh interval (also a default value of seven days), during which time DNS updates refresh the timestamp and put the record back into the no-refresh interval.
4. A record that ages past the refresh interval is available for scavenging when it reaches the scavenge interval.

The following zone attributes affect scavenging:

- ***scvg-interval***—Period during which the DNS server checks for stale records in a zone. The value can range from one hour to 365 days. You can also set this for the server (the default value is one week), although the zone setting overrides it.
- ***scvg-no-refresh-interval***—Interval during which actions, such as dynamic or prerequisite-only DNS updates, do not update the record timestamp. The value can range from one hour to 365 days. The zone setting overrides the server setting (the default value is one week).
- ***scvg-refresh-interval***—Interval during which DNS updates increment the record timestamp. After both the no-refresh and refresh intervals expire, the record is a candidate for scavenging. The value can range from one hour to 365 days. The zone setting overrides the server setting (the default value is one week).
- ***scvg-ignore-restart-interval***—Ensures that the server does not reset the scavenging time with every server restart. Within this interval, the Network Registrar ignores the time between when the server went down and was restarted, which is usually fairly short. The value can range from two hours to

one day. With any time longer than that set, Network Registrar recalculates the scavenging period to allow for record updates that cannot take place while the server is stopped. The zone setting overrides the server setting (the default value is 2 hours.

Enable scavenging only for zones where a Network Registrar DNS server receives updates exclusively from Windows clients (or those known to do automatic periodic DNS updates). Set the attributes listed above. The Network Registrar scavenging manager starts at server startup. It reports records purged through scavenging to the changeset database. Network Registrar also notifies secondary zones by way of zone transfers of any records scavenged from the primary zone. In cases where you create a zone that has scavenging disabled (the records do not have a timestamp) and then subsequently enable it, Network Registrar uses a proxy timestamp as a default timestamp for each record.

You can monitor scavenging activity using one or more of the log settings scavenge, scavenge-details, ddns-refreshes, and ddns-refreshes-details.

Local Advanced Web UI

On the Manage DNS Server page, click the Run icon ( ) in the Commands column to open the DNS Server Commands page (see Figure 7-3 on page 7-3). On this page, click the Run icon next to Scavenge all zones.

To scavenge a particular forward or reverse zone only, go to the Zone Commands for Zone page, which is available by clicking the Run icon ( ) on the List/Add Zones page or List/Add Reverse Zones page. Click the Run icon again next to Scavenge zone on the Zone Commands for Zone page. To find out the next time scavenging is scheduled for the zone, click the Run icon next to Get scavenge start time.

CLI Commands

Use **dns scavenge** for all zones that have scavenging enabled, or **zone** *name* **scavenge** for a specific zone that has it enabled. Use the **getScavengeStartTime** action on a zone to find out the next time scavenging is scheduled to start.

Troubleshooting DNS Update

You can use a standard DNS tool such as **dig** and **nslookup** to query the server for RRs. The tool can be valuable in determining whether dynamically generated RRs are present. For example:

```
$ nslookup
default Server: server2.example.com
Address: 192.168.1.2
> leasehost1.example.com
Server: server2.example.com
Address: 192.168.1.100
> set type=ptr
> 192.168.1.100
Server: server2.example.com
Address: 192.168.1.100

100.40.168.192.in-addr.arpa name = leasehost1.example.com
40.168,192.in-addr.arpa nameserver = server2.example.com
```

You can monitor DNS updates on the DNS server by setting the *log-settings* attribute to *ddns*, or show even more details by setting it to *ddns-details*.

Configuring DNS Update for Windows Clients

The Windows 2003 and XP operating systems rely heavily on DNS and, to a lesser extent, DHCP. This reliance represents a significant change from previous versions of Windows and requires careful preparation on the part of network administrators prior to wide-scale Windows deployments. Windows clients can add entries for themselves into DNS by directly updating forward zones with their address (A) record. They cannot update reverse zones with their pointer (PTR) records.

See Also

Client DNS Updates
Dual Zone Updates for Windows Clients, page 28-22
DNS Update Settings in Windows Clients, page 28-22
Windows Client Settings in DHCP Servers, page 28-23
SRV Records and DNS Updates, page 28-23
Recommended Windows Design Practices, page 28-25
Issues Related to Windows Environments, page 28-26
Frequently Asked Questions About Windows Integration, page 28-30

Client DNS Updates

It is not recommended that clients be allowed to update DNS directly. (See the "Recommended Windows Design Practices" section on page 28-25.)

For a Windows client to send address record updates to the DNS server, two conditions must apply:

- The Windows client must have the **Register this connection's addresses in DNS** box checked on the **DNS** tab of its TCP/IP control panel settings.
- The DHCP policy must enable direct updating (Network Registrar policies do so by default).

The Windows client notifies the DHCP server of its intention to update the A record to the DNS server by sending the *client-fqdn* DHCP option (81) in a DHCPREQUEST packet. By indicating the fully qualified domain name (FQDN), the option states unambiguously the client location in the domain namespace. Along with the FQDN itself, the client or server can send one of these possible flags in the *client-fqdn* option:

- **0**—Client should register its A record directly with the DNS server, and the DHCP server registers the PTR record (done through the policy *allow-client-a-record-update* attribute being enabled).
- **1**—Client wants the DHCP server to register its A and PTR records with the DNS server.
- **3**—DHCP server registers the A and PTR records with the DNS server regardless of the client request (done through the policy *allow-client-a-record-update* attribute being disabled, which is the default value). Only the DHCP server can set this flag.

The DHCP server returns its own *client-fqdn* response to the client in a DHCPACK based on whether DNS update is enabled. However, if the 0 flag is set (the *allow-client-a-record-update* attribute is enabled for the policy), enabling or disabling DNS update is irrelevant, because the client can still send its updates to DNS servers. See Table 28-3 on page 28-21 for the actions taken based on how various properties are set.

Table 28-3 Windows Client DNS Update Options

DHCP Client Action	DNS Update	DHCP Server Action
Checks **Register this connection's addresses in DNS** and sends *client-fqdn*; DHCP server enables *allow-client-a-record-update*	Enabled or disabled	Responds with *client-fqdn* that it allows the client to update its A records (sets flag 0), but the DHCP server still updates the PTR records.
Checks **Register**... and sends *client-fqdn*; DHCP disables *allow-client-a-record-update*	Enabled	Responds with *client-fqdn* that it does not allow the client to update the DNS server directly (sets flag 3), and updates the A and PTR records.
	Disabled	Does not respond with *client-fqdn* and does not update the DNS server.
Unchecks **Register**... and sends *client-fqdn*	Enabled	Responds with *client-fqdn* that it is updating the A and PTR records.
	Disabled	Does not respond with *client-fqdn* and does not update the DNS server.
Does not send *client-fqdn*	Enabled	Does not respond with *client-fqdn*, but updates the A and PTR records.
	Disabled	Does not respond with *client-fqdn* and does not update the DNS server.

A Windows DHCP server can set the *client-fqdn* option to ignore the client request. To enable this behavior in Network Registrar, create a policy for Windows clients and disable the *allow-client-a-record-update* attribute for this policy.

The following attributes are enabled by default in Network Registrar:

- **Server *use-client-fqdn***—The server uses the *client-fqdn* value on incoming packets and does not examine the *host-name*. The DHCP server ignores all characters after the first dot in the domain name value, because it determines the domain from the defined scope for that client. Disable *use-client-fqdn* only if you do not want the server to determine hostnames from *client-fqdn*, possibly because the client is sending unexpected characters.
- **Server *use-client-fqdn-first***—The server examines *client-fqdn* on incoming packets from the client before examining the *host-name* option (12). If *client-fqdn* contains a hostname, the server uses it. If the server does not find the option, it uses the *host-name* value. If *use-client-fqdn-first* is disabled, the server prefers the *host-name* value over *client-fqdn*.
- **Server *use-client-fqdn-if-asked***—The server returns the *client-fqdn* value in the outgoing packets if the client requests it. For example, the client might want to know the status of DNS activity, and hence request that the DHCP server should present the *client-fqdn* value.
- **Policy *allow-client-a-record-update***—The client can update its A record directly with the DNS server, as long as the client sets the *client-fqdn* flag to 0 (requesting direct updating). Otherwise, the server updates the A record based on other configuration properties.

The hostnames returned to client requests vary depending on these settings (see Table 28-4 on page 28-22).

Table 28-4 Hostnames Returned Based on Client Request Parameters

Client Request	With Server/Policy Settings	Resulting Hostname
Includes *host-name* (option 12)	*use-host-name*=true *use-client-fqdn*=false (or *use-client-fqdn-first*=false) *trim-host-name*=true	*host-name* trimmed at first dot. Example: *host-name* host1.bob is returned host1.
	(same except:) *trim-host-name*=false	*host-name*. Example: *host-name* host1.bob is returned host1.bob.
Includes *client-fqdn* (option 81)	*use-client-fqdn*=true *use-host-name*=false (or *use-client-fqdn-first*=true)	*client-fqdn* trimmed at first dot. Example: *client-fqdn* host1.bob is returned host1.
Omits *host-name* (option 12) and *client-fqdn* (option 81)	Or: *use-host-name*=false *use-client-fqdn*=false	Set by client/policy hierarchy.
	(same as the previous except:) hostname is undefined in the client/policy hierarchy, with *synthesize-name*=true	Synthesized following the synthesizing rule, which is to append the hyphenated IP address of the host after the specified *synthetic-name-stem*.
	(same as the previous except:) *synthesize-name*=false	Undefined.

Dual Zone Updates for Windows Clients

Windows DHCP clients might be part of a DHCP deployment where they have A records in two DNS zones. In this case, the DHCP server returns the *client-fqdn* so that the client can request a dual zone update. To enable a dual zone update, enable the policy attribute *allow-dual-zone-dns-update*.

The DHCP client sends the 0 flag in *client-fqdn* and the DHCP server returns the 0 flag so that the client can update the DNS server with the A record in its main zone. However, the DHCP server also directly sends an A record update based on the client secondary zone in the behalf of the client. If both *allow-client-a-record-update* and the *allow-dual-zone-dns-update* are enabled, allowing the dual zone update takes precedence so that the server can update the secondary zone A record.

DNS Update Settings in Windows Clients

The Windows client can set advanced properties to enable sending the *client-fqdn* option.

Step 1 On the Windows client, go to the Control Panel and open the TCP/IP Settings dialog box.

Step 2 Click the **Advanced** tab.

Step 3 Click the **DNS** tab.

Step 4 To have the client send the *client-fqdn* option in its request, leave the **Register this connection's addresses in DNS** box checked. This indicates that the client wants to do the A record update.

Windows Client Settings in DHCP Servers

You can apply a relevant policy to a scope that includes the Windows clients, and enable DNS updates for the scope.

Step 1 Create a policy for the scope that includes the Windows clients. For example:

a. Create a policywin2k.

b. Create a win2k scope with the subnet 192.168.1.0/24 and policywin2k as the policy. Add an address range of 192.168.1.10 through 192.168.1.100.

Step 2 Set the scope attribute *dynamic-dns* to update-all, update-fwd-only, or update-rev-only.

Step 3 Set the zone name, server address (for A records), reverse zone name, and reverse server address (for PTR records), as described in the "Creating DNS Update Configurations" section on page 28-5.

Step 4 If you want the client to update its A records at the DNS server, enable the policy attribute *allow-client-a-record-update* (this is the preset value). There are a few caveats to this:

- If *allow-client-a-record-update* is enabled and the client sends the *client-fqdn* with the update bit enabled, the *host-name* and *client-fqdn* returned to the client match the client *client-fqdn*. (However, if the *override-client-fqdn* is also enabled on the server, the hostname and FQDN returned to the client are generated by the configured hostname and policy domain name.)
- If, instead, the client does not send the *client-fqdn* with the update bit enabled, the server does the A record update, and the *host-name* and *client-fqdn* (if requested) returned to the client match the name used for the DNS update.
- If *allow-client-a-record-update* is disabled, the server does the A record updates, and the *host-name* and *client-fqdn* (with the update bit disabled) values returned to the client match the name used for the DNS update.
- If *allow-dual-zone-dns-update* is enabled, the DHCP server always does the A record updates. (See the "Dual Zone Updates for Windows Clients" section on page 28-22.)
- If *use-dns-update-prereqs* is enabled and *update-dns-first* is disabled, the hostname and *client-fqdn* returned to the client are not guaranteed to match the DNS update, because of delayed name disambiguation. However, the lease data will be updated with the new names.

Step 5 If in staged scope edit mode, reload the DHCP server.

SRV Records and DNS Updates

Windows relies heavily on the DNS protocol for advertising services to the network. Table 28-5 on page 28-24 describes how Windows handles service location (SRV) DNS RRs and DNS updates.

You can configure the Network Registrar DNS server so that Windows domain controllers can dynamically register their services in DNS and, thereby, advertise themselves to the network. Because this process occurs through RFC-compliant DNS updates, you do not need to do anything out of the ordinary in Network Registrar.

Table 28-5 Windows SRV Records and DNS Updates

Feature	Description
SRV records	Windows domain controllers use the SRV RR to advertise services to the network. This RR is defined in the experimental RFC 2782, "A DNS RR for specifying the location of services (DNS SRV)." The RFC defines the format of the SRV record (DNS type code 33) as: `_service._protocol.name ttl class SRV priority weight port target` There should always be an A record associated with target of the SRV record, so that the client can resolve the service back to a host. In the Windows implementation of SRV records, the records might look like this: `myserver.example.com A 10.100.200.11` `_ldap._tcp.example.com SRV 0 0 389 myserver.example.com` `_kdc._tcp.example.com SRV 0 0 88 myserver.example.com` `_ldap._tcp.dc_msdcs.example.com SRV 0 0 88 myserver.example.com` An underscore always precedes the service and protocol names. In the example, _kdc is the Kerberos Data Center. The priority and weight help you choose between target servers providing the same service (the weight differentiating those with equal priorities). With zero priority and weight, the listed order determines the priority. Windows domain controllers automatically place these SRV records in DNS.
How SRV records are used	When a Windows client boots up, it tries to initiate the network login process to authenticate against its Windows domain controller. The client must first discover where the domain controller is, and they do so using the dynamically generated SRV records. Before launching the net-login process, the client queries DNS with a service name; for example, _ldap._tcp.dc_msdcs.example.com. The DNS server SRV record target, for example, is my-domain-controller.example.com. The Windows client then queries DNS with the hostname my-domain-controller.example.com. DNS returns the host address and the client uses this address to find the domain controller. The net-login process fails without these SRV records.
DNS updates	When a Windows server is configured as a domain controller, you statically configure the name of the domain it manages through the Active Directory management console. This Windows domain should have a corresponding DNS zone associated with it. The domain controller should also have a series of DNS resolvers configured in its TCP/IP properties control panel. When the Windows domain controller boots up, it performs these steps to register itself in DNS and advertise its services to the network: 1. Queries DNS asking for the start of authority (SOA) record for the DNS domain that mostly closely encapsulates its Windows domain. 2. Identifies the primary DNS server for the DNS zone (from the SOA record) that mostly closely encapsulates its Windows domain name. 3. Creates a series of SRV records in this zone using the RFC 2136 DNS Update protocol.

Table 28-5 Windows SRV Records and DNS Updates (continued)

Feature	Description
Server boot process log file examples	Under normal operating conditions, the Network Registrar primary DNS server writes these log entries when a Windows domain controller boots up and creates its SRV records: `data time name/dns/1 Activity Protocol 0 Added type 33 record to name "_ldap._tcp.w2k.example.com", zone "w2k.example.com"` `data time name/dns/1 Activity Protocol 0 Update of zone "w2k.example.com" from address [10.100.200.2] succeeded.` This log shows only one DNS update for a single SRV record. A Windows domain controller typically registers 17 of these SRV records when it boots up.

To configure Network Registrar to accept these dynamic SRV record updates:

Step 1 Determine the IP addresses of the devices in the network that need to advertise services through DNS.

Step 2 If they do not exist, create the appropriate forward and reverse zones for the Windows domains.

Step 3 Enable DNS updates for the forward and reverse zones.

Step 4 Set up a DNS update policy to define the IP addresses of the hosts to which you want to restrict accepting DNS updates (see the "Configuring DNS Update Policies" section on page 28-13). These are usually the DHCP servers and any Windows domain controllers. (The Windows domain controllers should have static IP addresses.)

If it is impractical or impossible to enter the list of all the IP addresses from which a DNS server must accept updates, you can configure Network Registrar to accept updates from a range of addresses, although Cisco does not recommend this configuration.

Step 5 If in staged scope edit mode, reload the DNS and DHCP servers.

Recommended Windows Design Practices

Most Windows deployments are migrations from Windows NT 4.0 environments. Cisco suggests that you preserve the existing Windows NT 4.0 domain structure wherever possible. These rules and recommendations make Windows deployments more manageable.

See Also

Windows Rules and Suggestions, page 28-25
Naming Rules for Windows Domains, page 28-26

Windows Rules and Suggestions

Consider these rules and suggestions for Windows environments:

- Windows domains must not violate naming conventions for DNS hostnames.
- Statically configure the IP addresses of all Windows domain controllers.
- Change existing Windows NT 4.0 domain names to support DNS naming conventions.

- Where necessary, create new DNS zones for Windows domains; for example, the w2k.example.com subzone and the local.w2k.example.com domain.
- Do not allow clients to dynamically update their zones. Instead, configure the DHCP server so that it is responsible for all DNS updates.

Naming Rules for Windows Domains

Because the Windows domain space overlaps with the DNS name space, certain naming restrictions apply to Windows domains. Section 2.3.1 of RFC 1035 ("Domain Names Implementation and Specification") defines naming conventions for hostnames and domain names. Because of the dependence on DNS, these naming conventions apply to Windows domains.

- Hostnames and domain names in DNS are not case sensitive.
- Hostnames must:
  - Start with a letter.
  - End with a letter or a digit.
  - Contain internal characters that are letters, digits, or hyphens.

Therefore, you cannot use many characters commonly used in Windows NT 4.0 domain names for Windows domain names. Characters that you should not use in Windows domain names include the underscore (_), "at" symbol (@), and ampersand (&), all commonly used in Windows NT 4.0 domain names. Also, mixed-case domain names are no longer useful. For example, Windows recognizes ExampleDomain as equivalent to exampledomain.

Issues Related to Windows Environments

Table 28-6 describes the issues concerning interoperability between Windows and Network Registrar, The information in this table is intended to inform you of possible problems before you encounter them in the field. For some frequently asked questions about Windows interoperability, see the "Frequently Asked Questions About Windows Integration" section on page 28-30.

Table 28-6 Issues Concerning Windows and Network Registrar Interoperability

Issue	Description
Invisible dynamically created RRs	Network Registrar, when properly configured, accepts DNS updates from both DHCP and Windows servers. You can use the CLI to access the dynamic portion of the DNS zone for viewing and deleting records. Enter this command to view all DNS RRs in a given zone: `nrcmd> zone myzone listRR dynamic myfile` This redirects the output to the myfile file (see Example 28-1 on page 28-29). You can delete dynamically generated records by entering this command: `nrcmd> zone myzone removeDynRR myname [type]` You can also use **nslookup** to verify their existence, and you can use version 5.*x* (shipped with Windows) to view dynamic SRV records. In this version, use **set type=SRV** to enable viewing SRV records.

Table 28-6 Issues Concerning Windows and Network Registrar Interoperability (continued)

Issue	Description
Domain controller registration	A Windows domain controller has to register itself in DNS using DNS updates. The DNS RFCs dictate that only a primary DNS server for a particular zone can accept edits to the zone data. Hence, the Windows domain controller has to discover which DNS server is the primary for the zone that includes its Windows domain name. The domain controller discovers this by querying the first DNS server in its resolver list (configured in the TCP/IP properties control panel). The initial query is for the SOA record of the zone that includes the Windows domain of the domain controller. The SOA record includes the name of the primary server for the zone. If no zone exists for the domain name, the domain controller keeps removing the left-most label of the domain name and sends queries until it finds an SOA record with a primary server included in that domain. Once the domain controller has the name of the primary DNS server for its domain, it sends it DNS updates to create the necessary SRV records. Ensure that the name of the zone primary DNS server is in its SOA record.
Failure of A record DNS updates	When a Windows domain controller tries to advertise itself to the network, it sends several DNS update requests to the DNS server of record for its domain. Most of these update requests are for SRV records. However, the domain controller also requests an update for a single A record of the same name as the Windows domain. If the Network Registrar DNS server is also authoritative for a zone identical to this Windows domain, it rejects registering its A record, because the DNS A record update conflicts with the static SOA and NS records. This is to prevent possible security infractions, such as a dynamic host registering itself and spoofing Web traffic to a site. For example, the domain controller might control the w2k.example.com Windows zone. If a zone with the same name exists on the Network Registrar DNS server, these RRs could be part of that zone. (Example follows.) `w2k.example.com. 43200 SOA nameserver.example.com.` `hostmaster.example.com.` `(` `98011312 ;serial` `3600 ;refresh` `3600 ;retry` `3600000 ;expire` `43200 ) ;minim` `w2k.example.com.86400 NS nameserver.example.com` The domain controller would try to add an additional record; for example: `w2k.example.com. 86400 A 192.168.2.1` Network Registrar does not allow a DNS update to conflict with any statically configured name in the zone, even if the record type associated with that name is different. In the above example, an attempt to add an A record associated with the name w2k.example.com collides with the SOA and NS records.

Table 28-6 Issues Concerning Windows and Network Registrar Interoperability (continued)

Issue	Description
	When the domain controller boots up, a DNS log file entry such as this appears: `08/10/2000 16:35:33 name/dns/1 Info Protocol 0 Error - REFUSED - Update of static name "w2k.example.com", from address [10.100.200.2]` This is how Network Registrar responds to DNS updates of static DNS data. Additionally, you can ignore this DNS update failure. Windows clients do not use this A record. Allocation of domain controllers happens through SRV records. Microsoft added the A record to accommodate legacy NT clients that do not support SRV records. Note that failing to register the controller A record slows down the domain controller bootup process, affecting the overall login of worker clients. As mentioned earlier, the workaround is to define the Windows domain as a subdomain of the authoritative zone, or enable the DNS server *simulate-zone-top-dynupdate* attribute. If this is not possible, contact the Cisco Technical Assistance Center for help.
Windows RC1 DHCP clients	Microsoft released Windows build 2072 (known as RC1) with a broken DHCP client. This client sends a misformed packet that Network Registrar cannot parse. Network Registrar drops the packet and cannot serve the client, logging this error: `08/10/2000 14:56:23 name/dhcp/1 Activity Protocol 0 10.0.0.15 Lease offered to Host:'My-Computer' CID: 01:00:a0:24:1a:b0:d8 packet'R15' until True, 10 Aug 2000 14:58:23 -0400. 301 ms.` `08/10/2000 14:56:23 name/dhcp/1 Warning Protocol 0 Unable to find necessary Client information in packet from MAC address:'1,6,00:d0:ba:d3:bd:3b'. Packet dropped!` Network Registrar includes error checking specifically designed to deal with errors such as this improperly built FQDN option. However, if you encounter this problem, install the Microsoft patch to the RC1 client on the DHCP client. You must obtain this patch from Microsoft.
Windows plug-and-play network interface card (NIC) configuration	If configured to use DHCP, a Windows system tries to obtain a DHCP lease on startup. If no DHCP server is available, Windows may automatically configure the computer interface with a plug-and-play IP address. This address is not one that the network administrator or DHCP server configured or selected. These plug-and-play addresses are in the range 169.254.0.0/16. If you see devices in this address range on a network, it means that Windows autoconfigured the interfaces because it could not obtain a lease from a DHCP server. This can cause significant network and troubleshooting problems. The Windows system no longer informs the user that the DHCP client could not obtain a lease. Everything appears to function normally, but the client cannot route packets off its local subnet. Additionally, you may see the DHCP client trying to operate on the network with an address from the 169.254.0.0/16 network. This may lead you to think that the Network Registrar DHCP server is broken and handing out the wrong addresses.

Table 28-6 Issues Concerning Windows and Network Registrar Interoperability (continued)

Issue	Description
	If this problem occurs, perform these steps: 1. Ensure that the DHCP client has an active network port and a properly configured NIC. 2. Ensure that the network between the client and the DHCP server(s) are properly configured. Ensure that all router interfaces are configured with the correct IPHelper address. 3. Reboot the DHCP client. 4. If necessary, look at the DHCP log file. If the DHCP client can successfully route packets to the server, this logs a DHCPDISCOVER, even if Network Registrar does not answer the packet. If the network is correctly configured, and if the DHCP client is not broken, Network Registrar should receive the packet and log it. If there is no log entry for a packet receipt, there is a problem somewhere else in the network.
Scavenging Windows client address records	Windows clients do not clean up after themselves, potentially causing their dynamic record registration to remain indefinitely. This leaves stale address records on the DNS server. To ensure that these stale records are periodically removed, you must enable scavenging for the zone (see the "Scavenging Dynamic Records" section on page 28-18).

Example 28-1 Output Showing Invisible Dynamically Created RRs

```
Dynamic Resource Records
_ldap._tcp.test-lab._sites 600 IN SRV 0 100 389 CNR-MKT-1.w2k.example.com.
_ldap._tcp.test-lab._sites.gc._msdcs 600 IN SRV 0 100 3268 CNR-MKT-1.w2k.example.com.
_kerberos._tcp.test-lab._sites.dc._msdcs 600 IN SRV 0 100 88 CNR-MKT-1.w2k.example.com.
_ldap._tcp.test-lab._sites.dc._msdcs 600 IN SRV 0 100 389 CNR-MKT-1.w2k.example.com.
_ldap._tcp 600 IN SRV 0 100 389 CNR-MKT-1.w2k.example.com.
_kerberos._tcp.test-lab._sites 600 IN SRV 0 100 88 CNR-MKT-1.w2k.example.com.
_ldap._tcp.pdc._msdcs 600 IN SRV 0 100 389 CNR-MKT-1.w2k.example.com.
_ldap._tcp.gc._msdcs 600 IN SRV 0 100 3268 CNR-MKT-1.w2k.example.com.
_ldap._tcp.1ca176bc-86bf-46f1-8a0f-235ab891bcd2.domains._msdcs 600 IN SRV 0 100 389
    CNR-MKT-1.w2k.example.com.
e5b0e667-27c8-44f7-bd76-6b8385c74bd7._msdcs 600 IN CNAME CNR-MKT-1.w2k.example.com.
_kerberos._tcp.dc._msdcs 600 IN SRV 0 100 88 CNR-MKT-1.w2k.example.com.
_ldap._tcp.dc._msdcs 600 IN SRV 0 100 389 CNR-MKT-1.w2k.example.com.
_kerberos._tcp  600 IN SRV 0 100 88 CNR-MKT-1.w2k.example.com.
_gc._tcp 600 IN SRV 0 100 3268 CNR-MKT-1.w2k.example.com.
_kerberos._udp 600 IN SRV 0 100 88 CNR-MKT-1.w2k.example.com.
_kpasswd._tcp 600 IN SRV 0 100 464 CNR-MKT-1.w2k.example.com.
_kpasswd._udp 600 IN SRV 0 100 464 CNR-MKT-1.w2k.example.com.
gc._msdcs 600 IN A 10.100.200.2
_gc._tcp.test-lab._sites 600 IN SRV 0 100 3268 CNR-MKT-1.w2k.example.com.
```

Frequently Asked Questions About Windows Integration

These questions are frequently asked about integrating Network Registrar DNS services with Windows:

Q. What happens if both Windows clients and the DHCP server are allowed to update the same zone? Can this create the potential for stale DNS records being left in a zone? If so, what can be done about it?

A. The recommendation is not to allow Windows clients to update their zones. Instead, the DHCP server should manage all the client dynamic RR records. When configured to perform DNS updates, the DHCP server accurately manages all the RRs associated with the clients that it served leases to. In contrast, Windows client machines blindly send a daily DNS update to the server, and when removed from the network, leave a stale DNS entry behind.

Any zone being updated by DNS update clients should have DNS scavenging enabled to shorten the longevity of stale RRs left by transient Windows clients. If the DHCP server and Windows clients are both updating the same zone, three things are required in Network Registrar:

a. Enable scavenging for the zone.

b. Configure the DHCP server to refresh its DNS update entries as each client renews its lease. By default, Network Registrar does not update the DNS record again between its creation and its final deletion. A DNS update record that Network Registrar creates lives from the start of the lease until the lease expires. You can change this behavior using a DHCP server (or DNS update configuration) attribute, *force-dns-updates*. For example:

```
nrcmd> dhcp enable force-dns-updates
100 Ok
force-dns-updates=true
```

c. If scavenging is enabled on a particular zone, then the lease time associated with clients that the DHCP server updates that zone on behalf of must be less than the sum of the *no-refresh-interval* and *refresh-interval* scavenging settings. Both of these settings default to seven days. You can set the lease time to 14 days or less if you do not change these default values.

Q. What needs to be done to integrate a Windows domain with a pre-existing DNS domain naming structure if it was decided not to have overlapping DNS and Windows domains? For example, if there is a pre-existing DNS domain called example.com and a Windows domain is created that is called w2k.example.com, what needs to be done to integrate the Windows domain with the DNS domain?

A. In the example, a tree in the Windows domain forest would have a root of w2k.example.com. There would be a DNS domain named example.com. This DNS domain would be represented by a zone named example.com. There may be additional DNS subdomains represented in this zone, but no subdomains are ever delegated out of this zone into their own zones. All the subdomains will always reside in the example.com. zone.

Q. *In this case, how are DNS updates from the domain controllers dealt with?*

A. To deal with the SRV record updates from the Windows domain controllers, limit DNS updates to the example.com. zone to the domain controllers by IP address only. (Later, you will also add the IP address of the DHCP server to the list.) Enable scavenging on the zone. The controllers will update SRV and A records for the w2k.example.com subdomain in the example.com zone. There is no special configuration required to deal with the A record update from each domain controller, because an A record for w2k.example.com does not conflict with the SOA, NS, or any other static record in the example.com zone.

The example.com zone then might include these records:

```
example.com. 43200 SOA ns.example.com. hostmaster.example.com. (
    98011312 ;serial
    3600 ;refresh
    3600 ;retry
    3600000 ;expire
    43200 ) ;minimum
example.com.86400 NS ns.example.com
ns.example.com. 86400 A 10.0.0.10
_ldap._tcp.w2k.example.com. IN SRV 0 0 389 dc1.w2k.example.com
w2k.example.com 86400 A 10.0.0.25
...
```

Q. *In this case, how are zone updates from individual Windows client machines dealt with?*

A. In this scenario, the clients could potentially try to update the example.com. zone with updates to the w2k.example.com domain. The way to avoid this is to close down the zone to updates except from trusted sources. Before Network Registrar 6.0, the DNS server should be configured to accept updates from the DHCP server by IP address only. With Network Registrar 6.0, you can use transaction signatures (TSIG) between the DHCP server and the primary DNS server for the example.com zone.

Configure the DHCP server to do DNS updates to the example.com zone and the appropriate reverse zone for each client, and use option 81 to prevent the clients from doing DNS updates themselves.

Q. *Has security been addressed in this case?*

A. By configuring the forward and reverse zone to accept only updates from trusted IP addresses, you close the zone to updates from any other device on the network. Security by IP is not the most ideal solution, as it would not prevent a malicious attack from a spoofed IP address source. You can secure updates from the DHCP server by configuring TSIG between the DHCP server and the DNS server.

Q. *Is scavenging required in this case?*

A. No. Updates are only accepted from the domain controllers and the DHCP server. The DHCP server accurately maintains the life cycle of the records that they add and do not require scavenging. You can manage the domain controller dynamic entries manually by using the Network Registrar single-record dynamic RR removal feature.

Q. What needs to be done to integrate a Windows domain that shares its namespace with a DNS domain? For example, if there is a pre-existing DNS zone called example.com and a Windows Active Directory domain called example.com needs to be deployed, how can it be done?

A. In this example, a tree in the Windows domain forest would have a root of example.com. There is a pre-existing domain that is also named example.com that is represented by a zone named example.com.

Q. *In this case, how are DNS updates from individual Windows client machines dealt with?*

A. To deal with the SRV record updates, create subzones for:

```
_tcp.example.com.
_sites.example.com.
_msdcs.example.com.
_msdcs.example.com.
_udp.example.com.
```

Limit DNS updates to those zones to the domain controllers by IP address only. Enable scavenging on these zones.

To deal with the A record update from each domain controller, enable a DNS server attribute, *simulate-zone-top-dynupdate*.

```
nrcmd> dns enable simulate-zone-top-dynupdate
```

It is not required, but if desired, manually add an A record for the domain controllers to the example.com zone.

Q. *In this case, how are zone updates from individual Windows client machines dealt with?*

A. In this scenario, the clients could potentially try to update the example.com zone. The way to avoid this is to close down the zone to updates except from trusted sources. Before Network Registrar 6.0, the DNS server should be configured to accept updates from the DHCP server by IP address only. As of Network Registrar 6.0, you can use transaction signatures (TSIG) between the DHCP server and the primary DNS server for the example.com zone.

Configure the DHCP server to do DNS updates to the example.com zone and the appropriate reverse zone for each client, and use option 81 to prevent the clients from doing DNS updates themselves.

Q. *Has security been addressed in this case?*

A. By configuring the forward and reverse zone to accept only updates from trusted IP addresses, you close the zone to updates from other devices on the network. Security by IP is not the most ideal solution, as it would not prevent a malicious attack from a spoofed source. Updates from the DHCP server are more secure when TSIG is configured between the DHCP server and the DNS server.

Q. *Has scavenging been addressed in this case?*

A. Yes. The subzones _tcp.example.com, _sites.example.com, _msdcs.example.com, _msdcs.example.com, and _udp.example.com zones accept updates only from the domain controllers, and scavenging was turned on for these zones. The example.com zone accepts DNS updates only from the DHCP server.



CHAPTER 29

Using Extension Points

You can write extensions to affect how Cisco Network Registrar handles and responds to DHCP requests, and to change the behavior of a DHCP server that you cannot normally do using the user interfaces. This chapter describes the extension points to which you can attach extensions for DHCPv4 and DHCPv6.

Note Extensions are distinct from expressions (see Chapter 25, "Using Expressions"). Use extensions to modify request or response packets. Use expressions to create client identities or look up clients.

See Also

Using Extensions
Language-Independent API, page 29-4
Tcl Extensions, page 29-7
C/C++ Extensions, page 29-9
DHCP Request Processing Using Extensions, page 29-12
Extension Dictionaries, page 29-22
Extension Point Descriptions, page 29-27
Handling Objects and Options, page 29-43

Using Extensions

You can alter and customize the operation of the Network Registrar DHCP server by using extensions, functions that you can write in Tcl or C/C++.

Follow this process to create an extension for use in the DHCP server:

1. Determine the task to perform. What DHCP packet process do I want to modify?
2. Determine the approach to use. How do I want to modify the packet process?
3. Determine the extension point to which to attach the extension.
4. Choose the language (Tcl or C/C++).
5. Write (and possibly compile and link) the extension.
6. Add the extension to the DHCP server configuration.
7. Attach the extension to the extension point.
8. Reload the DHCP server so that it recognizes the extension.
9. Test and debug the results.

See Also

Creating, Editing, and Attaching Extensions
Determining Tasks, page 29-3
Deciding on Approaches, page 29-4
Choosing Extension Languages, page 29-4

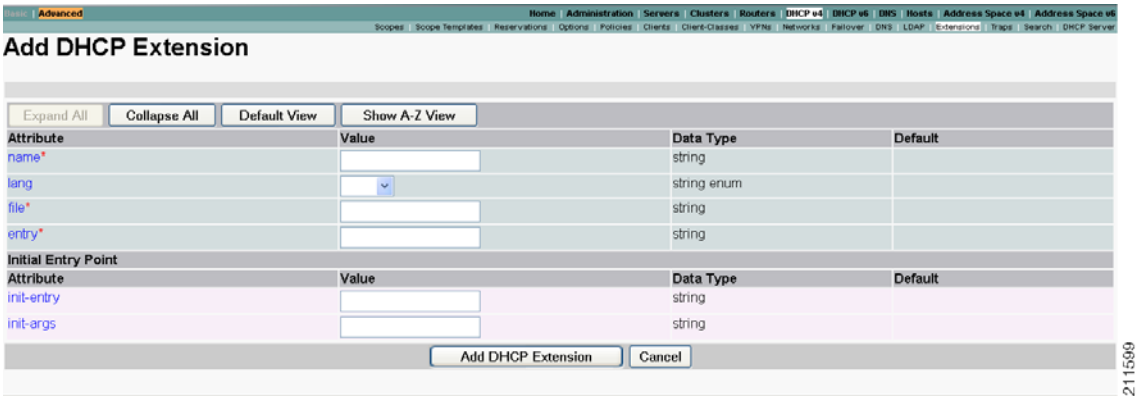
Creating, Editing, and Attaching Extensions

You can create, edit, and attach extensions.

Web UI

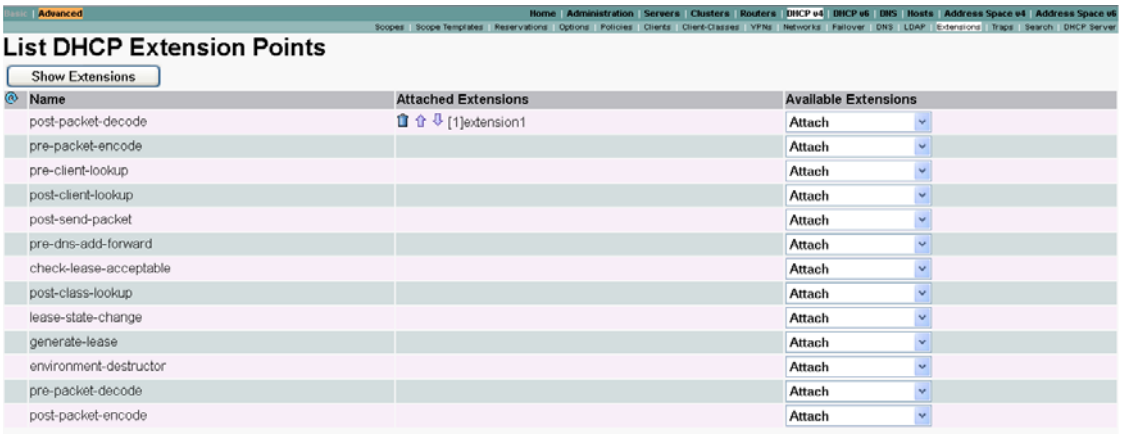
Click **Advanced**, then **DHCP**, then **Extensions** to open the List DHCP Extensions page. Click **Add DHCP Extension** to open the Add DHCP Extension page (see Figure 29-1).

Figure 29-1 Add DHCP Extension Page (Local Advanced)



To show the extension points where you can attach extensions, on the List DHCP Extensions page, click **Show Extension Points** at the top of the page to open the List DHCP Extension Points page (see Figure 29-2 on page 29-3). Once you create an extension, you can attach it to one or more of the extension points on this page, then click **Modify DHCP Extension**.

Figure 29-2 List DHCP Extension Points Page (Local Advanced)



See Also

Using Extensions, page 29-1

CLI Commands

Use the **extension** command, which requires this syntax:

```
nrcmd> extension name create language extension-file entry-point
```

The *entry-point* is the name of the entry point in the *extension-file*. You can also set an optional *init-entry* attribute value for the initial entry point each time the DHCP server loads the file (see the "init-entry" section on page 29-28). You can call this function from any extension point bound to this module. You can also list the extensions using **extension list**.

To attach and detach an extension, use **dhcp attachExtension** and **dhcp detachExtension** for the DHCP server, which require this syntax (the sequence number is optional):

```
nrcmd> dhcp attachExtension extension-point extension-name [sequence-number]
nrcmd> dhcp detachExtension extension-point [sequence-number]
```

Determining Tasks

The task to which to apply an extension is usually some modification of the DHCP server processing so that it better meets the needs of your environment. You can apply an extension at each of these DHCP server processing points, from receiving a request to responding to the client:

1. Receive and decode the packet.
2. Look up, modify, and process any client-class.
3. Build a response type.
4. Determine the subnet (or link, in the case of DHCPv6).
5. Find any existing leases.
6. Serialize the lease requests.
7. Determine the lease acceptability for the client.

8. Gather and encode the response packet.
9. Update stable storage of the packet.
10. Return the packet.

A more complete list of these steps (along with the extension points to use at each step) appears in the "DHCP Request Processing Using Extensions" section on page 29-12.

For example, you might have an unusual routing hub that uses BOOTP configuration. This device issues a BOOTP request with an Ethernet hardware type (1) and MAC address in the *chaddr* field. It then sends out another BOOTP request with the same MAC address, but with a hardware type of Token Ring (6). Specifying two different hardware types causes the DHCP server to allocate two IP addresses to the device. The DHCP server normally distinguishes between a MAC address with hardware type 1 and one with type 6, and considers them different devices. In this case, you might want to write an extension that prevents the DHCP server from handing out two different addresses to the same device.

Deciding on Approaches

Many solutions are often available to a single problem. When choosing the type of extension to write, you should first consider rewriting the input DHCP packet. This is a good approach, because it avoids having to know the internal processing of the DHCP server.

For the problem described in the "Determining Tasks" section on page 29-3, you can write an extension to solve it in either of these ways:

- Drop the Token Ring (6) hardware type packet.
- Change the packet to an Ethernet packet and then switch it back again on exit.

Although the second way involves a more complex extension, the DHCP client could thereby use either reply from the DHCP server. The second approach involves rewriting the packet, in this case using the **post-packet-encode** extension point (see the "post-packet-encode" section on page 29-41). Other approaches require other extensions and extension points.

Choosing Extension Languages

You can write extensions in Tcl or C/C++. The capabilities of each language, so far as the DHCP server is concerned, are similar, although the application programming interface (API) is slightly different to support the two very different approaches to language design:

- **Tcl**—Although scripting in Tcl might be somewhat easier than scripting in C/C++, it is interpreted and single-threaded, and might require more resources. However, you might be less likely than in C/C++ to introduce a serious bug, and there are fewer chances of a server failure. Network Registrar currently supports Tcl version 8.4.
- **C/C++**—This language provides the maximum possible performance and flexibility, including communicating with external processes. However, the C/C++ API is more complex than the Tcl API. Also, the possibility of a bug in the extension causing a server failure is also more likely in C/C++.

Language-Independent API

The following concepts are independent of whether you write your extensions in Tcl or C/C++.

See Also

Routine Signature
Dictionaries
Utility Methods in Dictionaries, page 29-6
Configuration Errors, page 29-6
Communicating with External Servers, page 29-6
Recognizing Extensions, page 29-7

Routine Signature

You need to define the extension as a routine in a file, which can contain multiple extension functions. You then attach the extension to one or more of the DHCP server extension points. When the DHCP server reaches that extension point, it calls the routine that the extension defines. The routine returns with a success or failure. You can configure the DHCP server to drop a packet on an extension failure.

You can configure one file—Tcl source file or C/C++ .dll or .so file—as multiple extensions to the DHCP server by specifying a different entry point for each configured extension.

The server calls every routine entry point with at least three arguments, the three dictionaries—request, response, and environment. Each dictionary contains many data items, each being a key-value pair:

- The extension can retrieve data items from the DHCP server by performing a get method on a dictionary for a particular data item.
- The extension can alter data items by performing a put or remove operation on many of the same named data items.

Although you cannot use all dictionaries at every extension point, the calling sequence for all routines is the same for every extension point. The extension encounters an error if it tries to reference a dictionary that is not present at a particular extension point. See the "Environment Dictionary" section on page 29-22 and the "Request and Response Dictionaries" section on page 29-25.

Dictionaries

You access data in the request, response, and server through a dictionary interface. Extension points include three types of dictionaries—request, response, and environment:

- **Request dictionary**—Information associated with the DHCP request, along with all that came in the request itself. Data is string-, integer-, IP address-, and blob-valued.
- **Response dictionary**—Information associated with generating a DHCP response packet to return to the DHCP client. Data is string-, integer-, IP address-, and blob-valued.
- **Environment dictionary**—Information passed between the DHCP server and extension.

For a description of the dictionaries, see the "Extension Dictionaries" section on page 29-22.

You can also use the environment dictionary to communicate between an extension attached at different extension points. When encountering the first extension point at which some extension is configured, the DHCP server creates an environment dictionary. The environment dictionary is the only one in which the DHCP server does not fix the names of the allowable data items. You can use the environment dictionary to insert any string-valued data item.

Every extension point in the flow of control between the request and response for the DHCP client (all extension points except **lease-state-change**, depending on the cause of the change), share the same environment dictionary. Thus, an extension can determine that some condition exists, and place a sentinel in the environment dictionary so that a subsequent extension can avoid determining the same condition.

In the previous example, the extension at the **post-packet-decode** extension point determines that the packet was the interesting one—from a particular manufacturer device, BOOTP, and Token Ring—and then rewrites the hardware type from Token Ring to Ethernet. It also places a sentinel in the environment dictionary and then, at a very simple extension at the **post-packet-encode** extension point, rewrites the hardware type back to Token Ring.

Utility Methods in Dictionaries

Each dictionary has associated utility methods with which you can reset the trace level for an extension and log values to an output file.

Configuration Errors

Extensions can fail for many reasons. For example:

- The server cannot find the file.
- The entry point or **init-entry** point does not appear in the file.
- The extension itself can return a failure from an **init-entry** call.

By itself, an extension failure is not fatal and does not prevent the DHCP server from starting. However, if you configure that failed extension at any extension point, the server will not start. Therefore, to debug the configuration process, you can configure the extension at the **init-entry** point (see the "init-entry" section on page 29-28) without attaching it to an extension point. When you complete that process successfully, you can attach your extension to an extension point.

Communicating with External Servers

You can write extensions that communicate with external servers or databases to affect the client class or validate incoming DHCP client requests. Writing such extensions is a complex task, requiring considerable skill and debugging expertise. Such extensions must be multithreaded, and need to communicate very swiftly with the external server if the DHCP server performance is to remain at an acceptable level.

Performance degradations can result from extensions stalling the threads that are processing requests. A thread stalls while an extension communicates with an external server. If this interaction takes more than 50 to 100 milliseconds, this severely affects server performance. This might or might not impact you in the particular environment in which you deploy this extension.

One way to avoid having to communicate with an external server synchronously (that is, stalling the incoming DHCP client request processing to communicate with the external server) is to avoid performing this communication while processing the DHCP client request. This sounds obvious, and it also sounds, on the face of it, impossible. However, due to the nature of the DHCP client-server protocol, there is a way to decouple the access to the external server from the DHCP client request processing.

To avoid this bottleneck, use a caching mechanism as part of the extension. When the server calls the extension for a request, have it check a cache (with proper locking to avoid multithreading problems) for the client data. If the client is:

- In the cache (and did not expire), have the extension accept or reject the request depending on the data in the cache.
- Not in the cache, have the extension queue a request to the external server (preferably over UDP), and then drop the DHCP client request. By the time the client retransmits the request, the data should be in the cache.

This caching mechanism requires the extension to have a receiver thread (started and stopped in the **init-entry** extension point). This thread reads the socket and updates the cache with the response. This thread (or a separate one) would also need to time out and remove old items from the cache. Using a single thread, however, might require setting a larger receive socket buffer size.

These techniques are only necessary if the load on the DHCP server is high and the speed of the external server is not high enough to support the required performance of the DHCP server load. However, this situation turns out to be all too common in practice. And, consider what can happen if the external server is unreachable (when connection timeouts are likely to be for minutes and not seconds).

Recognizing Extensions

The DHCP server only recognizes extensions when it initially configures itself at start or reload time. You can change an extension or the configuration for extensions in general. However, until you reload or restart the server, the changes have no effect. Forgetting to reload the DHCP server can be a frequent source of errors while debugging extensions.

The reason Network Registrar requires a reload is to ensure minimum processing impact by preloading extensions and getting them ready at server configuration time. While this approach is useful in production mode, it might cause some frustration when you debug extensions.

Tcl Extensions

If you choose to write your extensions in Tcl, you should understand the Tcl API, how to handle errors and Boolean variables, and how to initialize Tcl extensions. Network Registrar uses Tcl version 8.4.

See Also

Tcl Application Program Interface
Dealing with Tcl errors, page 29-8
Handling Boolean Variables in Tcl, page 29-8
Configuring Tcl Extensions, page 29-8
Init-Entry Extension Point in Tcl, page 29-9

Tcl Application Program Interface

Every Tcl extension has the same routine signature:

```
proc yourentry { request response environ } { # your-code }
```

To operate on the data items in any dictionary, you must treat these arguments as commands. Thus, to get the *giaddr* of the input packet, you would write:

```
set my_giaddr [ $request get giaddr ]
```

This sets the Tcl variable *my_giaddr* to the string value of the *giaddr* in the packet; for example, 10.10.1.5 or 0.0.0.0.

You could rewrite the *giaddr* in the input packet by using this Tcl statement:

```
$request put giaddr "1.2.3.4"
```

To configure one routine entry for multiple extension points and to alter its behavior depending on the extension point from which the server calls it, the DHCP server passes the ASCII name of the extension point in the environment dictionary under the key **extension-point**.

For some sample Tcl extensions, see the Network Registrar directory; by default:

- **Solaris and Linux**—/opt/nwreg2/examples/dhcp/tcl
- **Windows**—\Program Files\Network Registrar\examples\dhcp\tcl

Dealing with Tcl errors

You generate a Tcl error if you:

- Reference a dictionary that is not available.
- Reference a dictionary data item that is not available.
- Request a put operation on an invalid data item, for example, an invalid IP address.

In these cases, the extension immediately fails unless you surround the statement with a catch error statement:

```
catch { $request put giaddr "1.2.3.a" } error
```

Handling Boolean Variables in Tcl

In the environment dictionary, the Boolean variables are string-valued and have a value of **true** or **false**. The DHCP server expects an extension to set the value to **true** or **false**. However, in the request or response dictionaries, Boolean values are single-byte numeric format, and **true** is **1** and **false** is **0**. While more efficient for the C/C++ extensions, this approach does make the Tcl API a bit more complex.

Configuring Tcl Extensions

To configure a Tcl extension, write it and place it in the extensions directory. For UNIX and Linux, this is the /opt/nwreg2/extensions/dhcp/tcl directory. For Windows, this is the \Program Files\Network Registrar\extensions\dhcp\tcl directory.

When the DHCP server configures an extension during startup, it reads the Tcl source file into an interpreter. Any syntax errors in the source file that would render Tcl interpreter unable to load the file would also fail the extension. Typically, the DHCP server generates an error traceback in the log file from Tcl to help you to find the error.

Init-Entry Extension Point in Tcl

Tcl extensions support the **init-entry** extension point (see the "init-entry" section on page 29-28), and the arguments supplied in the *init-args* parameter to the command line appear in the environment dictionary associated with the key **arguments**.

Multiple Tcl interpreters can run in the DHCP server, for performance purposes, each in its own Tcl context. The server calls the Tcl extension once at the **init-entry** point for every Tcl context (interpreter) it runs. Ensure that your Tcl extension **init-entry** is robust, given multiple calls.

Information cannot flow between the Tcl contexts, but the **init-entry** can initialize global Tcl variables in each Tcl interpreter that any Tcl extension can access, regardless of the interpreter.

Note that all Tcl extensions share the Tcl interpreters. If your Tcl extension initializes global variables or defines procedures, ensure that these do not conflict with some other Tcl extension global variables or procedure names.

C/C++ Extensions

All DHCP C/C++ extensions are **dex** extensions, short for DHCP Extension.

See Also

C/C++ API
Using Types in C/C++, page 29-10
Building C/C++ Extensions, page 29-10
Using Thread-Safe Extensions in C/C++, page 29-10
Configuring C/C++ Extensions, page 29-11
Debugging C/C++ Extensions, page 29-11

C/C++ API

The routine signature for both the **entry** and **init-entry** routines for the C/C++ API is:

```
typedef int (DEXAPI * DexEntryPointFunction)(
    int iExtensionPoint,
    dex_AttributeDictionary_t* pRequest,
    dex_AttributeDictionary_t* pResponse,
    dex_EnvironmentDictionary_t* pEnviron );
```

Along with pointers to three structures, the integer value of the extension point is one of the parameters of each routine.

The C/C++ API is specifically constructed so that you do not have to link your shared library with any Network Registrar DHCP server files. You configure the entry to your routine when you configure the extension. The necessary call-back information for the operations to perform on the request, response, and environment dictionaries is in the structures that comprise the three dictionary parameters passed to your extension routine.

The DHCP server returns all binary information in network order, which is not necessarily properly aligned for the executing architecture.

Using Types in C/C++

Many C/C++ routines are available that use types, for example, **getByType()**. These routines are designed for use in performance-sensitive environments. The reasoning behind these routines is that the extension can acquire pointers to types once, for example, in the **init-entry** point, and thereafter use the pointers instead of string-valued names when calling the routines of the C/C++ API. Using types in this manner removes one hash table lookup from the extension processing flow of execution, which should improve (at least fractionally) the performance of any extension.

Building C/C++ Extensions

The directories /opt/nwreg2/examples/dhcp/dex (UNIX and Linux) and \Program Files\Network Registrar\examples\dhcp\dex (Windows) contain sample C/C++ extension code, as well as a short makefile designed to build the sample extensions. To build your own extensions, you need to modify this file. It has sections for Microsoft Visual C++ V5.0, GNU C++, and SunPro C++. Simply move the comment lines to configure the file for your environment.

Your extension needs to reference the include file dex.h. This file contains the information your program needs to use the C/C++ API. When building C/C++ extensions on Windows, remember to add your entry points to the .def file.

After you build the .dll (Windows) or .so (UNIX) file (all dex extensions are shared libraries), you need to move them into the /opt/nwreg2/extensions/dhcp/dex directory (UNIX), or the \Program Files\Network Registrar\extensions\dhcp\dex directory (Windows). You can then configure them.

Using Thread-Safe Extensions in C/C++

The DHCP server is multithreaded, so any C/C++ extensions written for it must be thread-safe. Multiple threads, and possibly multiple processors, must be capable of calling these extensions simultaneously at the same entry point. You should have considerable experience writing code for a multithreaded environment before designing C/C++ extensions for Network Registrar.

Caution All C/C++ extensions must be thread-safe. If not, the DHCP server will not operate correctly and will fail in ways that are extremely difficult to diagnose. All libraries and library routines that these extensions use must also be thread-safe.

On several operating systems, you must ensure that the runtime functions used are really thread-safe. Check the documentation for each function. Special thread-safe versions are provided (often *functionname*_r) on several operating systems. Because Windows provides different versions of libraries for multithreaded applications that are thread-safe, this problem usually does not apply.

Be aware that if any thread makes a non-thread-safe call, it affects any of the threads that make up the safe or locked version of the call. This can cause memory corruptions, server failures, and so on.

Diagnosing these problems is extremely difficult, because the cause of these failures are rarely apparent. To cause a server failure, you need very high server loads or multiprocessor machines with many processes. You might need running times of several days. Often, problems in extension implementation might not appear until after sustained periods of heavy load.

Because some runtime or third-party libraries might make non-thread-safe calls that you cannot detect, check your executables as to what externals are being linked (**nm** on UNIX).

If the thread (versions without the _r suffix by any thread on Solaris) calls the single-threaded version of any of the functions in the following list, do not use them. The interfaces to the thread-safe versions of these library routines can be different on different operating systems.

Thread-safe versions of the functions are:

asctime_r	gethostbyname_r	getservbyport_r
ctermid_r	gethostent_r	getservent_r
ctime_r	getnetbyaddr_r	getspent_r
fgetgrent_r	getnetbyname_r	getspnam_r
fgetpwent_r	getnetent_r	gmtime_r
fgetspent_r	getprotobyname_r	lgamma_r
gamma_r	getprotobynumber_r	localtime_r
getgrid_r	getprotoent_r	nis_sperror_r
getgrnam_r	getpwent_r	rand_r
getlogin_r	getrpcbyname_r	readdir_r
getpwnam_r	getrpcbynumber_r	strtok_r
getpwuid_r	getrpcent_r	tmpnam_r
getgrent_r	getservbyname_r	ttyname_r
gethostbyaddr_r		

Configuring C/C++ Extensions

Because the .dll and .so files are active when the server is running, it is not a good idea to overwrite them. After you stop the server, you can overwrite the .dll and .so files with newer versions.

Debugging C/C++ Extensions

Because your C/C++ shared library runs in the same address space as the DHCP server, and receives pointers to information in the DHCP server, any bugs in your C/C++ extension can very easily corrupt the DHCP server memory, leading to a server failure. For this reason, use extreme care when writing and testing a C/C++ extension. Frequently, you should try the approach to an extension with a Tcl extension and then code the extension in C/C++ for increased performance.

See Also

Pointers into DHCP Server Memory in C/C++
Init-Entry Entry Point in C/C++

Pointers into DHCP Server Memory in C/C++

The C/C++ extension interface routines return pointers into DHCP server memory in two formats:

- char* pointer to a series of bytes.
- Pointer to a structure called an abytes_t, which provides a pointer to a series of bytes with an associated length (defined in dex.h).

In both cases, the pointers into DHCP server memory are valid while the extension runs at that extension point. They are also valid for the rest of the extension points in the series processing this request. Thus, an abytes_t pointer returned in the **post-packet-decode** extension point is still valid in the **post-send-packet** extension point.

The pointers are valid for as long as the information placed in the environment dictionary is valid. However, there is one exception. One C/C++ routine, **getType**, returns a pointer to an abytes_t that references a type. These pointers are valid through the entire life of the extension. Typically, the server would call this routine in the **init-entry** extension point and save the pointers to the abytes_t structures that define the types in the static data of the shared library. Pointers to abytes_t structures returned by **getType** are valid from the **init-entry** call for initialization until the call for uninitialization.

Init-Entry Entry Point in C/C++

The DHCP server calls the **init-entry** extension point (see the "init-entry" section on page 29-28) once when configuring each extension and once when unconfiguring it. The dex.h file defines two extension point values passed as the extension points for the configure and unconfigure calls: DEX_INITIALIZE for configure and DEX_UNINITIALIZE for unconfigure. The environment dictionary value of the extension-point data item is **initialize** or **uninitialize** in each call.

When calling the **init-entry** extension point for **initialize**, if the environment dictionary data item **persistent** contains the value **true**, you can save and use the environment dictionary pointer any time before the return from the **uninitialize** call. In this way, background threads can use the environment dictionary pointer to log messages in the server log file. Note that you must interlock all access to the dictionary to ensure that at most one thread processes a call to the dictionary at a time. You can use the saved dictionary pointer up to when the extension returns from the **uninitialize** call. This way, the background threads can log messages during termination.

DHCP Request Processing Using Extensions

The Network Registrar DHCP server has extension points to which you can attach your own extensions. They have descriptive names that indicate where in the processing flow of control to use them.

Because the extension points are tied to the processing of input requests from DHCP clients, it is helpful to understand how the DHCP server handles requests. The stages in processing a request in the DHCP server (with slight variations between DHCPv4 and DHCPv6) are included in Table 29-1 on page 29-13.

Table 29-1 Stages in Processing Client Requests Using Extensions

Client Request Processing Stage	Extension Point Used
1. Receive a packet from a DHCP client.	**pre-packet-decode**
2. Decode the packet.	**post-packet-decode**
3. Determine any client-classes.	
4. Modify any client-class.	**post-class-lookup**
5. Process any client-classes, looking up clients.	**pre-client-lookup** **post-client-lookup**
6. Build a response container from the request.	
7. Determine the DHCPv4 network or DHCPv6 link from which the request arrived.	
8. In DHCPv4, find a lease already associated with this client, if any, or locate a new lease for the client.	
9. Serialize all requests associated with this client (processing continues when the request reaches the head of the serialization queue).	
10. In DHCPv6, process the client request, generating leases if necessary. The server tries to provide the client with at least one preferred lease for each usable prefix per binding. Note that you can generate leases and change lease states multiple times for a client request, but not for reserved leases.	**generate-lease** and **lease-state-change** (multiple calls are possible for both in DHCPv6)
11. Determine if the lease is (still) acceptable for this client (can occur multiple times in DHCPv6).	**check-lease-acceptable**
12. Initiate DNS Update operations as necessary (can occur multiple times in DHCPv6).	
13. Gather all the data to include in the response packet.	
14. Prepare the response packet for encoding.	**pre-packet-encode**
15. Encode the response packet for transmission to the client.	**post-packet-encode**
16. Update stable storage, if necessary.	
17. Send the packet to the client.	**post-send-packet**
18. Release all context for the client and request.	**environment-destructor**

These steps and additional opportunities for using extensions are explained in the following sections. The extension points are indicated in **bold**.

Note The **pre-packet-decode**, **post-packet-encode**, **generate-lease** (DHCPv6 only), and **environment-destructor** extension points are new in Network Registrar 7.0.

See Also

Enabling DHCPv6 Extensions
Receiving Packets, page 29-14
Decoding Packets, page 29-14

Determining Client-Classes, page 29-15
Modifying Client-Classes, page 29-15
Processing Client-Classes, page 29-15
Building Response Containers, page 29-16
Determining Networks and Links, page 29-16
Finding Leases, page 29-16
Serializing Lease Requests, page 29-17
Determining Lease Acceptability, page 29-17
DHCPv6 Leasing, page 29-19
Gathering Response Packet Data, page 29-20
Encoding Response Packets, page 29-20
Updating Stable Storage, page 29-21
Sending Packets, page 29-21
Processing DNS Requests, page 29-21
Tracing Lease State Changes, page 29-21

Enabling DHCPv6 Extensions

By default, extensions are assumed to support only DHCPv4. To write DHCPv6 extensions, you must implement an **init-entry** extension point that must:

1. Set the *dhcp-support* environment data item to **v4** (for DHCPv4 only, the preset value), **v6** (for DHCPv6 only), or **v4,v6** (for DHCPv4 and DHCPv6). This data item indicates to the server what the extension is willing to support.
2. Set the *extension-extensionapi-version* environment data item to **2**. (The *dhcp-support* data item is ignored unless the *extension-extension-api-version* is set to **2**.)

You will likely need to write separate extensions for DHCPv4 and DHCPv6, because of the differences in packet formats, DHCP protocol, and internal server data. However, the fundamentals of both kinds of extensions are very much the same. The server calls these extension points at essentially the same places during processing, although it can call some DHCPv6 extension points multiple times due to the possibility of multiple lease requests per client.

Receiving Packets

The DHCP server receives DHCPv4 packets on port 67 and DHCPv6 packets on port 547 (the DHCP input ports) and queues them for processing. It attempts to empty the UDP input queue as quickly as possible and keeps all of the requests that it receives on an internal list for processing as soon as a free thread is available to process them. You can configure the length of this queue, and it will not grow beyond its maximum configured length.

Decoding Packets

When a free thread is available, the DHCP server allocates to it the task of processing an input request. The first action it takes is to decode the input packet to determine if it is a valid DHCP client packet. As part of this decoding process, the DHCP server checks all options to see if they are valid—if the lengths of the options make sense in the overall context of the request packet. It also checks all data in the DHCP request packet, but takes no action on any data in the packet at this stage.

Use the **pre-packet-decode** extension point to rewrite the input packet. After the DHCP server passes this extension point, it stores all information from the packet in several internal data structures to make subsequent processing more efficient.

Determining Client-Classes

If you configure an expression in the *client-class-lookup-id*, it is at this stage that the DHCP server evaluates the expression (see Chapter 25, "Using Expressions," for a description of expressions). The result of the expression is either <null>, or something converted to a string. The value of the string must be either a client-class name or <none>. In the case of <none>, the server continues to process the packet in the same way as if there were no *client-class-lookup-id* configured. In the case of a <null> response or an error evaluating the *client-class-lookup-id*, the server logs an error message and drops the packet (unless an extension configured at the **post-class-lookup** extension point specifically instructs the server not to drop the packet). As part of the process of setting the client-class, the DHCP server evaluates any *limitation-id* configured for that client-class and stores it with the request.

Modifying Client-Classes

After the DHCP server evaluates the *client-class-lookup-id* and sets the client-class, it calls any extension attached to the **post-class-lookup** extension point. You can use that extension to change any data that the client-class caused to become associated with the request, including the *limitation-id*. The extension also learns if the evaluation of the *client-class-lookup-id* dropped the packet. The extension not only finds out if it needs to drop the packet, it instructs the server not to drop the packet if it wants the server not to do so.

Also, an extension running at the **post-class-lookup** extension point can set a new client-class for the request, and uses the data from that client-class instead of the current one. This is the only extension point where setting the client-class actually uses that client-class for the request.

Processing Client-Classes

If you enabled client-class processing, the DHCP server performs it at this stage.

Use the **pre-client-lookup** extension point to affect the client to look up, possibly by preventing the lookup or supplying data that overrides the existing data. After the DHCP server passes the **pre-client-lookup** extension point, it looks up the client (unless the extension specifically prevents it) in the local database or in an LDAP database, if one was configured.

After the server looks up the client, it uses the data in the client entry to fill in additional internal data structures. The DHCP server uses data in the specified client-class entry to complete any data that the client entry does not specify. When the DHCP server retrieves all the data stored in the various places in the internal data structures for additional processing, it runs the next extension point.

Use the **post-client-lookup** extension point to review the operation of the client-class lookup process, such as examining the internal server data structures filled in from the client-class processing. You can also use the extension point to change any data before the DHCP server does additional processing.

Building Response Containers

At this stage, the DHCP server determines the request type and builds an appropriate response container based on the input. For example, if the request is a DHCPDISCOVER, the server creates a DHCPOFFER response to perform the processing. If the input request is a BOOTP request, the server creates a BOOTP response to perform the response processing.

For DHCPv6, a server creates an ADVERTISE or REPLY packet, depending on the request.

Determining Networks and Links

The DHCP server must determine the subnet from which every request originated and map that into a set of address pools, scopes, or links that contain IP addresses.

For DHCPv4, internal to the DHCP server is the concept of a network, which, in this context, refers to a LAN segment or physical network. In the DHCP server, every scope belongs to a single network. Some scopes are grouped together on the same network because their network numbers and subnet masks are identical. Others are grouped because they are related through the primary-scope pointer.

The Network Registrar DHCP server determines the network to use to process a DHCP client request by:

- Determining the source address, either the *giaddr* or, if the *giaddr* is zero, the address of the interface on which the request arrived.
- Using this address to search the scopes for any scope that was configured in the server that is on the same subnet as this address. If the server does not find a scope, it drops the request.
- After finding the scope, using its network in subsequent processing.

For DHCPv6 processing, see the "Determining Links and Prefixes" section on page 26-3.

Finding Leases

For DHCPv4, now that when the DHCP server establishes the network, it searches the hash table held at the network level to see if the network already knows the *client-id*. "Already knows," in this context, means that this client previously received an offer or a lease on this network, and the lease was not offered to or leased by a different client since that time. Thus, a current lease or an available expired lease appears in the network level hash table. If the DHCP server finds a lease, it proceeds to the next step, which is to serialize all requests for the same IP address.

If the DHCP server does not find a lease, and if this is a BOOTP or DHCPDISCOVER request, the server looks for a reserved lease from a scope in the network. If it finds a reserved lease, the server checks whether the scope and lease are both acceptable. The following must be true of the reserved lease and the scope that contains it:

- The lease must be available (not leased to another DHCP client).
- The scope must support the request type (BOOTP or DHCP).
- The scope must not be in a deactivated state.
- The lease must not be in a deactivated state.
- The scope-selection tags must contain all of the client *selection-criteria* and none of the client *selection-criteria-excluded*.
- The scope must not be in a renew-only state.

If the reserved lease is acceptable, the server proceeds to the next step, which is to serialize all requests for the IP address. Having failed to find an existing or reserved lease for this client, the server now attempts to find any available IP addresses for this client.

The general process the DHCP server uses is to scan all of the scopes associated with this network in round-robin order, looking for one that is acceptable for this client and also has available addresses. An acceptable scope has these characteristics:

- If the client has *selection-criteria* associated with it, the scope-selection tags must contain all of the client inclusion criteria.
- If the client has *selection-criteria-excluded* associated with it, the scope-selection tags must contain none of the client exclusion criteria.
- The scope must support the client request type—If the client request is a DHCPREQUEST, you must enable the scope for DHCP. Likewise, if the request is a BOOTP request, you must enable the scope for BOOTP and dynamic BOOTP.
- It must not be in a renew-only state.
- It must not be in deactivated state.
- It must have an available address.

If the server does not find an acceptable scope, it logs a message and drops the packet.

For DHCPv6 processing, see the "Determining Links and Prefixes" section on page 26-3.

Serializing Lease Requests

Because multiple DHCP requests can be in process simultaneously for one client and lease, you must serialize DHCPv4 requests at the lease level. The server queues them on the lease and processes them in the order of queueing.

For DHCPv6, the server serializes on the client (per link) and not on the lease.

Determining Lease Acceptability

For DHCPv4, the DHCP server now determines if the lease is (still) acceptable for the client. In the case where this is a newly acquired lease for a first-time client, it will be acceptable. However, in the case where the server processes a renewal for an existing lease, the acceptability criteria might have changed since the server granted the lease, and you need to check its acceptability again.

If the client has a reservation that is different from the current lease, the server first determines if the reserved lease is acceptable. The criteria for release acceptability are:

- The reserved lease must be available.
- The reserved lease must not be in a deactivated state.
- The scope must not be in a deactivated state.
- If the request is BOOTP, the scope must support BOOTP.
- If the request is DHCP, the scope must support DHCP.
- If the client has any *selection-criteria*, the scope-selection tags must contain all of the client inclusion criteria.

- If the client has any *selection-criteria-excluded*, the scope-selection tags must contain none of the client exclusion criteria.
- If the client previously associated with this lease is not the current client, the scope must not be in a renew-only state.

If the reserved lease meets all of these criteria, the DHCP server considers the current lease unacceptable. If there is no reserved lease for this client, or the reserved lease did not meet the criteria for acceptability, the DHCP server examines the current lease for acceptability.

The criteria for acceptability are:

- The lease must not be in a deactivated state.
- The scope must not be in a deactivated state.
- If the request is BOOTP, the scope must support BOOTP. If the request is DHCP, the scope must support DHCP.
- If the client does not have a reservation for this lease, and the request is BOOTP, the scope must support dynamic BOOTP.
- If the client does not have a reservation for this lease, no other client can either.
- If the client has any *selection-criteria*, the scope-selection tags must contain all of the client inclusion criteria.
- If the client has any *selection-criteria-excluded*, the scope-selection tags must contain none of the client exclusion criteria.
- If the client previously associated with this lease is not the current client, the scope must not be in a renew-only state.

Tip At this point in the DHCP server processing, you can use the **check-lease-acceptable** extension point. You can use it to change the results of the acceptability test. Do this only with extreme care.

Upon determining that a lease is unacceptable, the DHCP server takes different actions, depending on the particular DHCP request currently being processed.

- **DHCPDISCOVER**—The DHCP server releases the current lease and attempts to acquire a different, acceptable lease for this client.
- **DHCPREQUEST SELECTING**—The DHCP server sends a NACK to the DHCP client because the lease is invalid. The client should then immediately issue a DISCOVER request to acquire a new DHCPOFFER.
- **DHCPRENEW**, **DHCPREBIND**—The DHCP server sends a NACK to the DHCP client to attempt to force the DHCP client into the INIT phase (attempt to force the DHCP client into issuing a DHCPDISCOVER request). The lease is still valid until the client actually issues the request.
- **BOOTP**—The DHCP server releases the current lease and attempts to acquire a different, acceptable lease for this client.

Caution Take extreme care with the **check-lease-acceptable** extension point. If the answer the extension point returns does not match the acceptability checks in the search for an available lease performed in a DHCPDISCOVER or dynamic BOOTP request, an infinite server loop can result (either immediately, on the next DHCPDISCOVER or BOOTP request). In this case, the server would acquire a newly available lease, determine that it was not acceptable, try to acquire a newly available lease, and determine that it was not acceptable, in a continuous loop.

DHCPv6 Leasing

The DHCP server processes IPv6 lease requests by scanning the client request for IA_NA, IA_TA, and IA_PD options (see the "DHCPv6 Bindings" section on page 26-5). For each of these options, the server considers any leases that the client explicitly requests. If the lease already exists for the client and binding (IA option and IAID), the server determines if the lease is still acceptable. For leases that the client requests that do not already exist for the client, the server tries to give that lease to the client if:

- Another client or binding is not already using the lease.
- The prefix for the lease has the client-request flag set in its *allocation-algorithms* attribute.
- The lease is usable and on a usable prefix (see the "DHCPv6 Prefix Usability" section on page 29-19).

Next, the server tries to assure that clients are using reservations and that a client has a usable lease with a nonzero preferred lifetime on each usable prefix on the link. Thus, the server processes each of these bindings as follows:

1. Adds any client reservations (not already in use) to the binding, provided the reservation flag is set in the prefix *allocation-algorithms* attribute. The server uses the first binding of the appropriate type for the reservation; that is, it uses address leases for IA_NA bindings and prefix leases for IA_PD bindings.
2. If the client has no lease with a nonzero preferred lifetime on each prefix that the client can use, the server tries to allocate a lease to the client. The prefix *allocation-algorithms* flags control how the server allocates the lease.

See Also

DHCPv6 Prefix Usability
DHCPv6 Lease Usability, page 29-19
DHCPv6 Lease Allocation, page 29-20

DHCPv6 Prefix Usability

A usable prefix:

- Is not deactivated.
- Did not expire.
- Allows leases of the binding type.
- Matches the client selection criteria, if any.
- Does not match the client selection exclusion criteria, if any.

DHCPv6 Lease Usability

A usable lease is:

- Not unavailable.
- Not revoked.
- Not deactivated.
- Not reserved for a different client.
- Not subject to *inhibit-all-renews* or *inhibit-renews-at-reboot*.

- Renewable if being renewed (IA_TA leases are not renewable).
- Leasable with a nonzero valid lifetime.

DHCPv6 Lease Allocation

When the server needs to allocate a new lease on a prefix, it calls any extensions registered at the **generate-lease** extension point if the prefix extension flag is set in the *allocation-algorithms* attribute. (See the "generate-lease" section on page 29-35.) The extensions can supply the address (IA_NA or IA_TA binding) or prefix (IA_PD binding) to be assigned, request that the server use its normal allocation algorithm (if enabled in *allocation-algorithms*), or request the server to skip assigning a lease on this prefix. The server might call the extension again if it supplied an address or prefix that is not valid or is already in use.

If extensions are not allowed, there are no extensions registered, or the extension requests the normal allocation algorithm of the server, the server allocates a randomly generated address or finds the first best-fit available prefix (as controlled by the prefix *allocation-algorithms* attribute) and creates the lease.

Once the server has a lease and does an acceptability check on it (see the "DHCPv6 Lease Usability" section on page 29-19), the server calls any extensions registered at the **check-lease-acceptable** extension point to allow the extension to alter the acceptability of the lease. (See the "check-lease-acceptable" section on page 29-36.) You would typically only use this extension point to change an acceptable result into a nonacceptable one; however, the server allows a nonacceptable result to be changed to an acceptable one, although this is strongly discouraged because of possibly adverse consequences. If the lease is not acceptable, the server likely tries to allocate another lease; thus, use care to avoid an infinite loop. In some cases, you might need the **check-lease-acceptable** and **generate-lease** extension points for full control of the leases a client gets: **generate-lease** can request the server to skip allocation of the lease.

The server calls the **check-lease-acceptable** extension point for each client request for each lease.

Gathering Response Packet Data

In this stage of processing, the DHCP server collects all the data to send back in the DHCP response and determines the address and port to which to send the response. You can use the **pre-packet-encode** extension point to change the data sent back to the DHCP client in the response, or to change the address to which to send the DHCP response. (See the "pre-packet-encode" section on page 29-39.)

Caution Any packets that you drop at the **pre-packet-encode** extension point, whether they be DHCP or BOOTP packets, still show the address to be leased in the Network Registrar lease state database, for as long as the remaining lease time. Because of this, it is advisable to drop packets at an earlier point.

Encoding Response Packets

In this stage, the DHCP encodes the information in the response data structure into a network packet. If this DHCP client requires DNS activity, the DHCP server queues a DNS work request to the DNS processing subsystem in the DHCP server. That request runs whenever it can, but generally not before sending the packet to the client. (See the "Processing DNS Requests" section on page 29-21.)

Updating Stable Storage

At this stage, the DHCP server ensures that the on-disk copy of the information is up to date with respect to the IP address before proceeding. For DHCPv6, this can involve multiple leases.

Sending Packets

Use the **post-send-packet** extension point (see the "post-send-packet" section on page 29-42) for any processing that you want to perform outside of the serious time constraints of the DHCP request-response cycle. After the server sends the packet to the client, it calls this extension point.

Processing DNS Requests

Here is a simplified view of what the DHCP server does to add names to DNS:

1. **It builds up a name to use for the A record**—The DHCP server creates the name that it will use in the forward (A record) DNS request. For DHCPv6, these are AAAA records. The DNS name comes from option values in the DHCP request or the server synthesizes them if enabled to do so. Note that you can use extensions to insert or modify option values for the purpose of constructing a name from data in the request packet.
2. **Attempts to add the name, asserting that none exists yet**—At this stage, the prerequisites for the DNS name update request indicate that the name should not exist. If it succeeds, the DHCP server proceeds to update the reverse record.
3. **Attempts to add the name, asserting that the DHCP server should supply it**—The DHCP server attempts to add the hostname, asserting that the host exists and that it has the same TXT record as the one that was sent. If this succeeds, the server proceeds to the next task, which is to update the reverse record. If it fails, the server checks if it exhausted its naming retries. If it did, it exits and logs an error. If not, it returns to the first step, which is to build up a name for the A record.

   For DHCPv6, the server uses DHCID records instead of TXT records. Also, DHCPv6 clients can have multiple leases and the forward zones can be the same or potentially different.
4. **Updates the reverse record**—Now that the DHCP server knows which name to associate with the reverse (PTR) record, it can update the reverse record with no prerequisites, because it can assume it is the owner of the record. If the update fails, the DHCP server logs an error.

Tracing Lease State Changes

The server calls the **lease-state-change** extension point whenever (and only when) a lease changes state. The existing state is in the response dictionary **lease-state** data item. The new state is in the environment dictionary under **new-state**. The **new-state** never equals the existing state (if it did, the server would not call the extension). You should consider this extension to be read-only and not make modifications to any dictionary items, because the server calls it in many different places. Use this extension point only for tracking changes to a lease state.

Extension Dictionaries

Every extension is a routine with three arguments. These arguments represent the request dictionary, response dictionary, and environment dictionary. Not every dictionary is available to every extension. Table 29-2 shows the extensions points and the dictionaries that are available to them.

Table 29-2 Extensions Points and Relevant Dictionaries

Extension Point	Dictionary
init-entry	Environment
pre-packet-decode	Request, Environment
post-packet-decode	Request, Environment
pre-client-lookup	Request, Environment
post-client-lookup	Request, Environment
post-class-lookup	Request, Environment
generate-lease	Request, Response, Environment
lease-state-change	Response, Environment
check-lease-acceptable	Request, Response, Environment
pre-packet-encode	Request, Response, Environment
post-packet-encode	Request, Response, Environment
post-send-packet	Request, Response, Environment
environment-destructor	Environment

Note When the server sends DHCPv6 Reconfigure messages, it can call the **pre-packet-encode**, **post-packet-encode**, and **post-send-packet** extension points without a request.

For the request and response dictionaries, you can use the **isValid** method to probe if the dictionary is available for an extension point.

Each of the three dictionaries consists of name-value pairs. The environment dictionary, which is available to every extension point, is the simplest dictionary. The request and response dictionaries are more complex and their data is typed. Thus, when you set a value in one of these dictionaries, you need to match the data type to the value. You can use the dictionaries for getting, putting, and removing values.

See Also

Environment Dictionary
Request and Response Dictionaries, page 29-25

Environment Dictionary

The environment dictionary is available at all extension points. It is strictly a set of name-value pairs in which both the name and the value are strings.

The DHCP server uses the environment dictionary to communicate with extensions in different ways at different extension points. At some extension points, the server places information in the environment dictionary for the extension to modify. In others, the extension can place values in the environment dictionary to control the flow or data after the extension finishes its processing.

The environment dictionary is unique in that an extension can put any name-value pair in it. Although you do not get an error for using undocumented name-value pairs, the server does not recognize them. These name-value pairs can be useful for your extension points to communicate data among them.

The DHCP server creates the environment dictionary when a DHCP request arrives and the dictionary remains with that request through the processing. Thus, an extension that runs at the **post-packet-decode** extension point can put data into the environment dictionary, and then an extension run at the **pre-packet-encode** extension point might read that data from the dictionary.



Note The **init-entry** extension point has a unique environment dictionary.

See Also

Environment Dictionary Data Items
Initial Environment Dictionary, page 29-25

Environment Dictionary Data Items

The data items in Table 29-3 are valid in the environment dictionary.

Table 29-3 Environment Dictionary Data Items

Data Item	Description
attempts (read-only)	For the **generate-lease** extension point only. The number of times that the server calls this extension for a single lease.
default-prefix-length (read-only)	For the **generate-lease** extension point only. Set to the default prefix length (from policies). (Note that the Expert mode *longest-prefix-length* and *shortest-prefix-length* data items, if not set, default to the *default-prefix-length*.)
dhcp-support (read/write)	For the **init-entry** extension point only. The version or versions of DHCP for which the server should call the registered extension points for this extension. Can be **v4**, **v6**, or **v4,v6**.
drop (read/write)	If the *drop* value is equal to the string **true** when the extension exits, then the DHCP server drops the input packet and logs a message in the log file. Available at most extension points, but not all (such as **generate-lease**).
extension-extensionapi-version (write-only)	For the **init-entry** extension point only. The minimum version of the extension API required by the extension. Set it to **2** if the extension uses API features introduced in Network Registrar Release 7.0, otherwise **1** for earlier releases.

Table 29-3 Environment Dictionary Data Items (continued)

Data Item	Description
extension-name (read-only)	Name with which the extension was configured. You can configure the same piece of code as several different extensions and at several different extension points. This allows one piece of code to do different things, depending on how you configure it. The code can also use this string to find itself in the extension-name-sequence string, for which it needs to know its own name.
extension-name-sequence (read-only)	Provides a comma-separated string representing the configured extensions for this extension point. It allows an extension to determine which extensions can run before and after it (the *extension-name* data item provides the currently running extension). For example, if you configure **tclfirst** as the first extension and **dexscript** as the fifth, the *extension-name-sequence* would contain **"tclfirst,,,,dexscript"**.
extension-point (read-only)	Name of the extension point. For example, **post-packet-decode**.
extension-sequence (read-only)	String that is the sequence number of this extension at this extension point.
generated-address (write-only)	For the **generate-lease** extension point only. The address the extension wants the server to use for the lease.
generated-prefix (write-only)	For the **generate-lease** extension point only. The delegated DHCPv6 prefix the extension wants the server to use for the lease.
log-drop-message (read/write)	If the *drop* value is equal to the string **true** and the *log-drop-message* value is equal to the string **false** when the extension exits, then the DHCP server drops the input packet, but does not log a message in the log file. For all extension points that have a request dictionary, you can set the data items that begin with *log* and *verbose-logging* at any time. The DHCP server reads them as needed.
prefix-length (write-only)	For the **generate-lease** extension point only. Set to the requested or default prefix length.
server-dhcp-support (read-only)	For the **init-entry** extension point only. The server sets this data item to indicate what the server is configured to support. Can be **v4**, **v6**, or **v4,v6**, depending on the DHCP server *dhcp-support* attribute setting (which requires setting expert attribute visibility=3) and whether any prefixes are configured: • If *dhcp-support*=**both** and prefixes are not configured, then *server-dhcp-support* is set to **v4**. • If *dhcp-support*=**both** and one or more prefixes are configured, then *server-dhcp-support* is set to **v4,v6**. • If *dhcp-support*=**v4**, then *server-dhcp-support* is set to **v4**. • If *dhcp-support*=**v6** and one or more prefixes are configured, then *server-dhcp-support* is set to **v6**.
server-extensionapi-version (read-only)	For the **init-entry** extension point only. The version of the server extension API. The value is **2** for Network Registrar Release 7.0 and later, and **1** for earlier releases.

Table 29-3 Environment Dictionary Data Items (continued)

Data Item	Description
skip-lease (write-only)	For the **generate-lease** extension point only. Set to TRUE if the extension does not want the server to generate the lease.
trace-level (write-only)	Setting this to a number makes that number the current setting of the *extension-trace-level* server attribute for all extensions processing this request.
user-defined-data (read/write)	Set with the *user-defined-data* attribute of a lease stored with the lease before request processing. You can have it written to disk *before* a **pre-packet-encode**. If set to null, the server ignores the *user-defined-data* from the lease. You cannot write this data item for all requests, especially for **pre-packet-encode** and **post-packet-encode** extension points.

Initial Environment Dictionary

You can configure an extension with *init-args* and **init-entry**. Alternatively, you can specify configuration information for an extension to read out of the environment dictionary. You can set the DHCP property *initial-environment-dictionary* with a series of attribute-value pairs, and each pair is available in every environment dictionary. Using this capability, you can specify a variety of configuration and customizing information. Any extension can simply read this data directly out of the environment dictionary, without having to store it in some static data area, as is required with the *init-args* or **init-entry** approach.

You can read the values defined using the *initial-environment-dictionary* approach from any environment dictionary. You can also define new values for any attributes that appear in the *initial-environment-dictionary*. These new values are then available for the life of that environment dictionary (usually the life of the request packet being processed). However, this does not change the contents of any other environment dictionary. Any new environment dictionary (associated with a different request) sees the attribute-value pairs defined by the *initial-environment-dictionary* property of the DHCP server.

In addition, these *initial-environment-dictionary* attribute-value pairs do not appear in an enumeration of the values of the environment dictionary. They are only available if you request an attribute value not currently defined in the environment dictionary. The attribute-value pairs do not actually appear in the environment dictionary. Thus, if you define a new value for one of the attributes, that new value does appear in the environment dictionary. If you later delete the value, the original one is again available if you should request it.

Request and Response Dictionaries

The request and response dictionaries have a fixed set of accessible names. However, you cannot access all names from every extension point. These dictionaries make internal server data structures available to the extension for read-write or, in some cases, read-only access. Each data item has a particular data type. If you omit the correct data type (for C/C++ extensions) on a put operation, or if the DHCP server cannot convert it to the correct data type (for Tcl extensions), the extension will fail.

The request dictionary is available at the beginning of the processing of a request. After the server creates a response, both the request and response dictionaries are available. It is an error to access a response dictionary before it is available.

In general, you cannot use an extension to change information data in the server. In some cases, however, you can use an extension to change configured data, but only for the duration of the processing for just that single request.

See Also

Decoded DHCP Packet Data Items
Using Parameter List Option, page 29-27

Decoded DHCP Packet Data Items

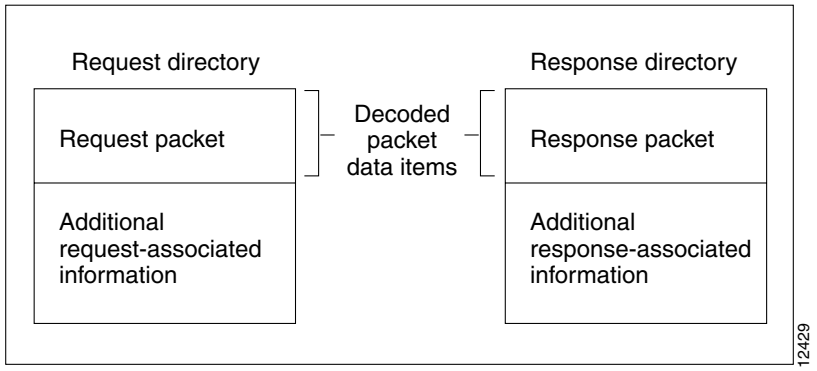
The DHCP protocol is a request-response UDP-based protocol and, thus, the stimulation for a DHCP server operation is usually a DHCP request from a client. The result is usually a DHCP response sent back to that client.

The DHCP extension facility makes the information input in the DHCP request available to extensions at most of the extension points, and the information to be sent as a response to a DHCP request available at the **pre-packet-encode** extension point (see the "pre-packet-encode" section on page 29-39).

In addition to this DHCP packet-based information, there is additional data that the DHCP server uses when processing DHCP requests. This data becomes associated with either the DHCP request or the DHCP response as part of the architecture of the server. Much of this data is also made available to extensions, and much of it can be both read and written—in many cases altering the processing algorithms of the DHCP server.

The request and response dictionaries, therefore, contain two classes of data in each dictionary. They contain decoded packet data items as well as other request or response associated data items. The decoded packet data items are those data items directly contained in or derived from the DHCP request or DHCP response. Access to the decoded packet data items allows you to read and, in some cases, rewrite the DHCP request and DHCP response packet. Figure 29-3 shows the relationship between the request and the response dictionaries.

Figure 29-3 Extensions Request and Response Dictionaries



You can access information from the DHCP request packet, such as the *giaddr*, *ciaddr*, and all the incoming DHCP options by using the decoded packet data items in the request dictionary. Similarly, you can set the *giaddr* and *ciaddr*, and add and remove DHCP options in the outgoing DHCP response by accessing the decoded packet data items in the response dictionary.

It is important to realize that access to the packet information provided by the decoded packet data items is not all available to you. The specific data items available to that extension point are listed in the description of each extension point. Because the decoded packet data items are always accessible as a group, they are listed as a group. See Table B-2 on page B-3.

You access DHCP options by name. If the option is not present, the server returns no data for that option. If you place an option into the decoded request or decoded response, it replaces any option with the same name already in the decoded request or decoded response, unless, in the put operation, you want the data specifically appended to existing data.

Some DHCP options can have multiple values. For example, the routers option can have one or more IP addresses associated with it. Access to these multiple values is by indexed operations on the option name.



Tip A **clear** operation on the request or response dictionary removes all the options in the decoded packet.

Using Parameter List Option

There is one option, *dhcp-parameter-request-list*, that the DHCP server specially handles in two ways, available as either a:

- Multiple-valued option of bytes under the name *dhcp-parameter-request-list*.
- Blob (sequence of bytes) option under the name *dhcp-parameter-request-list-blob*.

You can get or put the option using either name. The DHCP server handles the *dhcp-parameter-request-list* (and its *-blob* variant as well) differently in the response dictionary than in the request dictionary. When you access this option in the request dictionary, it is just another DHCP option in the request dictionary. In the response dictionary, however, special processing takes place.

You can use the *dhcp-parameter-request-list* option in the response dictionary to control the order of the options returned to the DHCP or BOOTP client. When you put the option in the response dictionary, the DHCP server reorders the existing options so that the ones listed in the option are first and in the order that they appear in the list. Then, the remaining options appear in their current order after the last ones that were in the list. The DHCP server retains the list, and uses it to order any future options that it puts into the response, until it replaces the list with a new one.

When an extension does a get operation for the *dhcp-parameter-request-list* in the response dictionary, it does not look in the decoded response packet to find an option. Instead, the DHCP server synthesizes one that contains the list of all options currently in the decoded response packet.

Extension Point Descriptions

The following sections describe each extension point, their actions, and data items. For all the extension points, you can read the **extension-point** and set the *trace-level* data item values in the environment dictionary. For most extension points, you can also tell the server to drop the packet.

See Also

init-entry, page 29-28
pre-packet-decode, page 29-28
post-packet-decode, page 29-29
post-class-lookup, page 29-30
pre-client-lookup, page 29-30

post-client-lookup, page 29-34
generate-lease, page 29-35
check-lease-acceptable, page 29-36
lease-state-change, page 29-37
pre-packet-encode, page 29-39
post-packet-encode, page 29-41
pre-dns-add-forward, page 29-42
post-send-packet, page 29-42
environment-destructor, page 29-42

init-entry

The **init-entry** extension point is an additional one that the DHCP server calls when it configures or unconfigures the extension, which occurs when starting, stopping, or reloading the server. This entry point has the same signature as the others for the extension, but you can use only the environment dictionary. You do not configure the **init-entry** extension with **dhcp attachExtension** in the CLI, but you do so implicitly by defining an **init-entry** on an already configured extension.

Note You must supply an **init-entry** extension point to enable extension points for DHCPv6 (or disable them for DHCPv4).

In addition to configuring an **init-entry** with the name of the entry point, you can also configure a string of arguments that the DHCP server loads in the environment dictionary under the string *arguments* before calling the **init-entry** point. Using *arguments*, you can create a customized extension by giving it different initialization arguments and thus not require a change to the code to elicit different behavior.

You configure arguments by setting *init-args* on an existing extension point. These arguments are present for both the configure and unconfigure calls of the **init-entry** entry point. The extension point name for the configure call is **initialize**, and for the unconfigure call is **uninitialize**.

Note The order in which the server calls extensions at the **init-entry** extension point can be different from reload to reload or release to release.

Caution An extension, when called to uninitialize, must terminate any threads it creates and clean up after itself before returning. Once the extension returns, the DHCP server unloads the extension from memory, which could result in a server failure if a thread an extension created is left running.

pre-packet-decode

The dictionaries available for **pre-packet-decode** are request and environment.

This extension point is the first one the DHCP server encounters when a request arrives. The server calls it after receiving a packet but before it decodes the packet (at the **post-packet-decode** extension point). An extension can use this extension point to examine a packet and alter it before the server decodes it, or cause the server to drop it.

Two new data items are in the request dictionary for use with this extension point:

- **client-packet**—Read-only item and valid only for **getBytes**. The server returns the address and length of the client portion of the packet. For DHCPv4, this is the complete packet. For DHCPv6, this is the client message.
- **packet**—Read/write item and valid only for **getBytes** and **putBytes**. For **getBytes**, it returns the address and length of the entire packet. For **putBytes**, it replaces the existing packet with a new, decoded one. For DHCPv4, **packet** is the same as **client-packet**. For DHCPv6, **packet** is the full packet if relayed, or the same as **client-packet** if not relayed.

An extension can directly alter the bytes of the packet by writing to the packet. However, an extension must use **putBytes** to adjust the length of the packet (and the operation can fail if the packet is too big). For DHCPv6, adjusting the length of the client portion of the packet, if relayed, requires updating the lengths in the Relay Message options in the packet.

It is up to an extension to handle parsing the packet to locate what it needs and properly alter the packet, if that is the intent.

Because the server has not yet decoded the received packet, most request dictionary data items are not available (as the server normally fills them in from the received packet). Thus, this extension point must extract data directly from the packet. The extension must also properly handle incorrectly formatted packets.

Caution The request dictionary *client-packet* and *packet* data items used for **pre-packet-decode** are available at any extension point that has a request dictionary. However, you should not directly alter or replace the packet at any extension point other than **pre-packet-decode**, because doing so can have unexpected side effects. For example, the server might never pick up the changes to the packet, or options data can change unexpectedly during processing.

Note that if you enable receive packet logging, the server logs the packet after it calls the extensions registered for this extension point.

post-packet-decode

The dictionaries available for **post-packet-decode** are request and environment.

This extension point immediately follows the decoding of the input packet and precedes any processing on the data in the packet. The primary activity for an extension at this point is to read information from an input packet and do something with it. For example, you might use it to rewrite the input packet.

The **post-packet-decode** extension point is one of the easiest extension points to use. If you can express the change in server behavior as a rewrite of the input DHCP or BOOTP packet, you should use this extension point. Because the packet was decoded, but not processed in any way, the number of side effects are very limited.

The **post-packet-decode** extension point is the only one at which you can make modifications to the decoded input packet and ensure that the server recognizes all the modifications. If the extension decides to drop the packet and terminate further processing, it can do so by using the **drop** data item in the environment dictionary.

Table 29-4 on page 29-30 lists the data items that are available in the **post-packet-decode** request dictionary.

Table 29-4 post-packet-decode Request Dictionary Data Items

Request Dictionary Data Item	Value	Operation
client-ipaddress	IP address	read/write
client-port	int	read/write
os-type	string	read/write
reply-to-client-address	int	read/write
transaction-time	int	read-only

post-class-lookup

The dictionaries available for **post-class-lookup** are request and environment.

The server calls this extension point only if there is a *client-class-lookup-id*; otherwise, it is similar to a **post-packet-decode**. The server calls the **post-class-lookup** extension point after evaluating the *client-class-lookup-id* and setting the client-class data for this client. It has the same dictionaries as the **post-packet-decode** extension point.

On input to this extension point, the environment dictionary has the *drop* data item set to true or false. You can change this setting by extension to drop the packet (or not drop it), and the server recognizes the change. The server also looks at the *log-drop-message* to decide whether to log the drop.

The extension point can also set the *client-class-name* in the environment dictionary, which sets the named client-class for this packet, regardless of the previous client-class. This setting has an effect only if the *drop* environment dictionary data item value is false on exiting the extension point.

pre-client-lookup

The dictionaries available for **pre-client-lookup** are request and environment.

You can use this extension point only if you enabled client-class processing for your DHCP server. This extension point allows an extension to perform any or all of these actions:

- Modify the client that the server looks up during client-class processing.
- Specify individual data items to override those found from the client entry or client-class it specifies.
- Instruct the server to skip the client lookup altogether. In this case, the only client data used is data that the extension supplied in the environment dictionary.

Although the request dictionary is available to make decisions about the operation of an extension running at this extension point, the environment dictionary controls all the operations.

See Also

Environment Dictionary for pre-client-lookup, page 29-31
Client-Class Data Input to pre-client-lookup, page 29-31
Request Dictionary for pre-client-lookup, page 29-33

Environment Dictionary for pre-client-lookup

The data items in Table 29-5 are available at **pre-client-lookup** for client-class control.

Table 29-5 pre-client-lookup Environment Dictionary Data Items

Environment Dictionary Data Item	Description	Operation
client-specifier	Name of the client the client-class processing code looks up, in MCD or LDAP. If you change the name at this extension point, the DHCP server looks up whatever client you specify.	read/write
default-client-class-name	Instructs the server to use the value associated with the *default-client-class-name* option as the *class-name* if: • The *client-specifier* data item was not specified in the **pre-client-lookup** script. • The server could not locate the specific client entry. The *default-client-class-name* data item then assumes precedence over the *class-name* associated with the default client.	read/write
release-by-ip	Instructs the server to release the lease by the IP address if it cannot retrieve the lease by the *client-id* (derived from the DHCPRELEASE request).	read/write
skip-client-lookup	If you set this item to **true**, the DHCP server skips the normal client lookup that it would have performed immediately upon exit from this extension. In this case, the only data items used to describe this client are those placed in the environment dictionary.	read/write
user-defined-data	Set with the *user-defined-data* attribute of a lease stored with the lease before processing the request. If set to null, the server ignores the *user-defined-data*.	read/write

Client-Class Data Input to pre-client-lookup

If you set the environment dictionary data items in Table 29-6 on page 29-32, their values override those determined from the client lookup (either in the internal database or from LDAP). If you do not add anything to the dictionary, then the server uses what is in the client value, or key.

Table 29-6 pre-client-lookup Environment Dictionary Client Data Items

Client-Class Data Item	Description	Operation
action	Convert this string to a number and use the result as the action. The numbers you can use are **0** (for none) and **1** (for exclude).	read/write
authenticate-until	Absolute time, measured in seconds, from January 1, 1970. Use to indicate the time at which the client authentication expires. When the client authentication expires, the DHCP server uses the values in the client *unauthenticated-client-class* option instead of its client-class to fill in missing data items in the client entry.	read/write
client-class-name	Use the client-class specified by this data item to fill in the missing information in the client entry. If there is no client-class corresponding to the name specified, the DHCP server logs a warning and continues processing. If you specify **none**, the DHCP server acts as if the client entry did not include the client-class name.	read/write
domain-name	Use this domain name for the client DNS operations in preference to the one specified in the scope. Note that the DNS server shown as the primary server for the domain in the scope must also be the primary server for the domain you specified. If there is no override for the domain name in the client or client-class entry, the DHCP server uses the domain name from the scope. If the client entry or the extension contains the word **none**, the DHCP server uses the domain name from the scope.	read/write
host-name	Use this for the client in preference to the host-name options specified in the input packet, or any data from the client or client-class entry. If you set this to **none**, the DHCP server does not use any information from the client or client-class entry, but uses the name from the client request.	read/write
policy-name	Use this policy as the policy specified for the client entry, overriding any policy specified by that client entry.	read/write

Table 29-6 pre-client-lookup Environment Dictionary Client Data Items (continued)

Client-Class Data Item	Description	Operation
selection-criteria	List of comma-separated strings, each specifying (for this input packet) scope-selection criteria for this client. Any scope this client uses must have all of these scope-selection tags. Use this data item to override any criteria specified in the client or client-class entry. If you do, the DHCP server does not use the client entry scope-selection criteria, independent of whether they were stored in the local or LDAP database. If you set this data item to **none**, the DHCP server does not use scope-selection tags for this packet. If you set this to a null string, the DHCP server treats it as if it were not set and uses the scope-selection criteria from the client or client-class entry.	read/write
selection-criteria-excluded	List of comma-separated strings, each specifying (for this input) exclusion criteria for this client. Any scope this client uses must not have any of these selection tags. Use this data item to override any specified client or client-class entries. If you do, the DHCP server does not use the client entry scope-exclusion criteria, independent of whether they were stored in the local or LDAP database. If you set this to **none**, the DHCP server does not use any scope-exclusion tags for this packet. If you set this data item to a null string, the DHCP server treats is as if it were not set and uses the scope-exclusion criteria from the client or client-class entry.	read/write
unauthenticated-client-class-name	Name of the client-class to use if the server does not authenticate the client. If you want to indicate not specifying the **unauthenticated-client-class-name**, use an invalid client-class name as the value of this data item. You can use the value **none** or any name that is not a client-class name. The DHCP server logs an error that the client-class is not present.	read/write
user-defined-data	Set with the *user-defined-data* attribute of a lease stored with the lease before processing the request. If set to null, the server ignores the *user-defined-data*.	read/write

Request Dictionary for pre-client-lookup

You can use all of the decoded packet data items specified in Table 29-7 on page 29-34. The request information data items available for the **pre-client-lookup** extension point are **client-ipaddress**, **client-port**, and **transaction-time** (see Table 29-4 on page 29-30). Table 29-7 on page 29-34 describes the client information data items, and Table 29-8 on page 29-34 describes the client understanding data items available at this extension point.

Table 29-7 pre-client-lookup Request Dictionary Client Data Items

Client Information Data Item	Value	Operation
client-id	blob	read/write
client-id-created-from-mac-address	int	read-only
client-mac-address	blob	read/write (DHCPv4 only)
client-os-type	string	read/write

Table 29-8 pre-client-lookup Request Dictionary Client Understanding Data Items

Client Understanding Data Item	Value	Operation
client-wants-nulls-in-strings	int	read/write
import-packet	int	read/write
reply-to-client-address	int	read/write

post-client-lookup

The dictionaries available for **post-client-lookup** are request and environment.

You can use this extension point to examine the results of the entire client-class processing operation, and take an action based on those results. You might want to use it to rewrite some of the results, or to drop the packet. If you want to override the hostname in the packet returned from the client-class processing from an extension running at the **post-client-lookup** extension point, set the hostname to the **client-requested-host-name** data item in the request dictionary. This causes Network Registrar to look to the server as though the packet came in with whatever string you specified in that data item.

You also can use this extension point to place some data items in the environment dictionary to affect the processing of an extension running at the **pre-packet-encode** extension point (see the "pre-packet-encode" section on page 29-39), where it might load different options into the response packet or take other actions.

See Also

Environment Dictionary for post-client-lookup
Request Dictionary for post-client-lookup, page 29-35

Environment Dictionary for post-client-lookup

The environment dictionary data items included in Table 29-9 on page 29-35 are available at **post-client-lookup**.

Table 29-9 post-client-lookup Environment Dictionary Data Items

Environment Dictionary Data Item	Description	Operation
client-specifier	Name of the client that the client-class processing looked up.	read-only
cnr-ldap-query-failed	The DHCP server sets this attribute to ease recovery from LDAP server failures so that a post-client-lookup script can respond to an LDAP server failure. The DHCP server, after a client lookup, sets this flag to **true** if the LDAP query failed because of an LDAP server error. If the server received a response from the LDAP server, one of two conditions occurs: • It sets the flag to **false**. • The *cnr-ldap-query-failed* attribute does not appear in the environment dictionary.	read-only
user-defined-data	Set with the *user-defined-data* attribute of a lease stored with the lease before the request is processed. If set to null, the server ignores the *user-defined-data*.	read/write

Request Dictionary for post-client-lookup

You can use all the decoded packet data items described in Table 29-10. The request information data items available for the **post-client-lookup** extension point are **client-ipaddress**, **client-port**, and **transaction-time** (see Table 29-4 on page 29-30). The client information data items are the same as those for the **pre-client-lookup** extension point (see Table 29-7 on page 29-34).

Table 29-10 post-client-lookup Request Dictionary Client Understanding Data Items

Client Understanding Data Item	Value	Operation
client-class-name	string	read-only
client-class-policy	string	read/write
client-domain-name	string	read/write
client-host-name	string	read/write
client-policy	string	read/write
client-requested-host-name	string	read/write
client-wants-nulls-in-strings	int	read/write
import-packet	int	read/write
reply-to-client-address	int	read/write
selection-criteria	string	read/write
selection-criteria-excluded	string	read/write

generate-lease

The dictionaries available for **generate-lease** are request, response, and environment. This extension point is available for DHCPv6 only.

You can use this extension point to generate a DHCPv6 address or prefix and allow the extension to control the address or prefix. The server calls **generate-lease** only if the prefix is configured to allow extensions to be called during address allocation or prefix delegation—the extension flag must be set in the *allocation-algorithms* attribute for the prefix. When the server calls the **generate-lease** extension:

- The server sets the prefix context for the response dictionary to the prefix on which the lease is to be created. (Calling **setObject** with DEX_PREFIX and DEX_INITIAL will return to this context.)
- No lease context exists, because the server has not yet created a lease. However, lease-binding data items, in particular *lease-binding-type* and *lease-binding-iaid* are available. (Calling **setObject** with DEX_LEASE and DEX_INITIAL returns to this context and also sets the prefix, because a lease context sets three contexts: lease, binding, and prefix.)
- The server sets the *skip-lease* environment dictionary data item to false.
- The server sets the (read-only) *attempts* environment dictionary data item with the number of times (starting with 1) it called the extension to create this lease.
- For prefix delegation, the following environment dictionary data items are available:
  - ***prefix-length***—Prefix length (requested or default prefix length).
  - ***default-prefix-length***—Default prefix length (from policies).
  - ***longest-prefix-length***—Longest allowable prefix (from policies).
  - ***shortest-prefix-length***—Shortest allowable prefix (from policies).

When the extension returns, it can:

- Request an explicit address (for stateful address assignment) by setting the address on the *generated-address* environment dictionary data item. Note that if the address is not available for the client, the server might call this extension again.
- Request an explicit prefix (for prefix delegation assignment) by setting the prefix on the *generated-prefix* environment dictionary data item. Note that if the prefix is not available for the client, the server might call this extension again.
- Cause the server not to assign a lease by setting the *skip-lease* environment dictionary data item to true.
- Allow normal address assignment or prefix delegation to occur by not setting any of the above.

The server calls the extension point at most 500 times for each lease (this limit is the same one that currently applies when the server randomly generates leases). The server calls an extension multiple times only if the extension supplies an unusable address or delegated prefix (that is not in range for the prefix or already exists).

Note You cannot request the server to drop the packet at this extension point.

check-lease-acceptable

The dictionaries available for **check-lease-acceptable** are request, response, and environment.

This extension point comes immediately after the server determines whether the current lease is acceptable for this client. You can use this extension to examine the results of that operation, and to cause the routine to return different results. See the "Determining Lease Acceptability" section on page 29-17.

The *acceptable* data item is available in the environment dictionary at this extension point. This is a read/write data item that the DHCP server initializes depending on if the lease is acceptable for this client. You can read and change this result in an extension. Setting the acceptable data item to true indicates that it is acceptable; setting it to false indicates that it is unacceptable.

All the request dictionary data items available for **pre-packet-encode** are available for **check-lease-acceptable**. (See the "pre-packet-encode" section on page 29-39.)

All the response dictionary data items available for **pre-packet-encode** are available for **check-lease-acceptable**, with the addition of the *client-os-type* data item. This is a read/write data item that you can read and change in an extension. However, you can set the *client-os-type* data item only by changing the **os-type** data item in the **post-packet-decode** request dictionary.

lease-state-change

The dictionaries available for **lease-state-change** are response and environment.

The server calls this extension point when a lease changes its state. The existing state is in the *lease-state* response dictionary lease information data item (see Table 29-12 on page 29-38). The new state is in the environment dictionary data item *new-state*. The server never calls the extension point if the new state matches the existing one.

Use this extension point mainly for read-only purposes, although you can place data in the environment dictionary so that other extension points can get it later.

The **lease-state-change** can also have a different environment dictionary, such as for lease expirations.

See Also

Response Dictionary for lease-state-change
Environment Dictionary for lease-state-change, page 29-38

Response Dictionary for lease-state-change

The **lease-state-change** extension point has the same response dictionary data items as **pre-packet-encode** (see the "pre-packet-encode" section on page 29-39), although only the client information and lease information data items are useful (see Table 29-11 and Table 29-12 on page 29-38). For this extension point, do not access any options in the response dictionary or any encoded packet data items (such as *giaddr* and *ciaddr*), because there is often no response packet transmitted.

Table 29-11 lease-state-change Response Dictionary Client Data Items

Client Information Data Item	Value	Operation
client-domain-name	string	read/write
client-expiration-time	blob	read-only
client-host-name	string	read/write
client-id	blob	read/write
client-id-created-from-mac-address	int	read-only
client-last-transaction-time	int	read-only
client-mac-address	blob	read/write (DHCPv4 only)

Table 29-11 lease-state-change Response Dictionary Client Data Items (continued)

Client Information Data Item	Value	Operation
client-requested-host-name	string	read/write
domain-name-changed	int	read/write
host-name-changed	int	read/write
host-name-in-dns	int	read/write
reverse-name-in-dns	int	read/write

Table 29-12 lease-state-change Response Dictionary Lease Data Items

Lease Information Data Item	Value	Operation
lease-deactivated	int	read-only
lease-ipaddress	IP address	read-only
lease-reserved	int	read-only
lease-state	string	read-only
response-source	string	read-only; values returned are that the server is processing: *unknown*—None of the following *client*—Client packet *failover*—Binding update from the failover partner *timeout*—Lease expiration or grace period end *operator*—Request from a user interface *one-lease-per-client*—One lease per client (removing a client from an old lease because of a new one)
start-time-of-state	int	read-only

Environment Dictionary for lease-state-change

The current state is in the *lease-state* lease information data item in the response dictionary (see Table 29-12 on page 29-38), and the state being changed to is in the environment dictionary under the *new-state* data item. This is all read-only.

In the initialization entry point for a script attached to this extension point, if on exit the environment dictionary contains an attribute *exiting-state*, the server saves the value of that attribute and calls the script only when a lease is transitioning away from that state. Thus, you might set *exiting-state* to **leased**, and the server calls the extension point only when the existing state was leased and the new state was something else. You would do this because you caught other lease transitions in other ways (such as in the **post-send-packet** extension point) and you did not want to incur overhead for calling this extension point except for transitions that you really care about. The valid lease states are:

- available
- offered
- leased
- expired
- unavailable
- released

- other-available
- pending-available

There is no strict state transition table for these states. In a failover environment, the server that receives a binding update message typically sets the state to whatever its partner tells it to be, without requiring any specific state transitions.

pre-packet-encode

The dictionaries available for **pre-packet-encode** are request, response, and environment.

Note For DHCPv6 Reconfigure messages, there is no request dictionary (because Reconfigure is a server-initiated message). Thus, enabled extensions should check the response *msg-type* for ADVERTISE or REPLY or use **isValid** on the request to ensure that the Reconfigure message exists.

See Also

Request Dictionary for pre-packet-encode
Response Dictionary for pre-packet-encode

Request Dictionary for pre-packet-encode

You can use all the decoded packet data items described in Table 29-13. The request information data items available for the **pre-packet-encode** extension point are *client-ipaddress*, *client-port*, and *transaction-time* (see Table 29-4 on page 29-30). Table 29-13 describes the client information data items. The client understanding data items are the same as for the **post-client-lookup** extension point (see Table 29-10 on page 29-35).

Table 29-13 pre-packet-encode Client Data Items

Client Information Data Item	Value	Operation
client-id	blob	read/write
client-id-created-from-mac-address	int	read-only
client-mac-address	blob	read/write (DHCPv4 only)
reply-to-client-address	int	read/write

Response Dictionary for pre-packet-encode

The **pre-packet-encode** extension point has these response dictionary data items:

- **General**—See Table 29-14 on page 29-40
- **Client information**—See Table 29-15 on page 29-40
- **Lease information**—See Table 29-16 on page 29-40
- **Scope address information**—See Table 29-17 on page 29-41
- **Scope acceptability information**—See Table 29-18 on page 29-41
- **Scope DNS information**—See Table 29-19 on page 29-41

Table 29-14 pre-packet-encode General Response Dictionary Data Items

Response Dictionary Data Item	Value	Operation
reply-ipaddress	IP address	read/write
reply-port	int	read/write
scope-ping-clients	int	read-only
scope-renew-only	int	read-only
scope-renew-only-expire-time	int	read-only
scope-selection-tags	string	read-only
scope-send-ack-first	int	read-only
transaction-time	int	read-only

Table 29-15 pre-packet-encode Client Data Items

Client Information Data Item	Value	Operation
client-domain-name	string	read-only
client-expiration-time	blob	read-only
client-host-name	string	read-only
client-id	blob	read-only
client-id-created-from-mac-address	int	read-only
client-last-transaction-time	int	read-only
client-mac-address	blob	read-only (DHCPv4 only)
client-requested-host-name	string	read-only
domain-name-changed	int	read-only
host-name-changed	int	read-only
host-name-in-dns	int	read-only
reverse-name-in-dns	int	read-only

Table 29-16 pre-packet-encode Lease Data Items

Lease Information Data Item	Value	Operation
lease-deactivated	int	read-only
lease-ipaddress	IP address	read-only
lease-reserved	int	read-only
lease-state	string	read-only
lease-start-time-of-state	int	read-only

Table 29-17 pre-packet-encode Scope Address Data Items

Scope Address Information Data Item	Value	Operation
scope-network-number	IP address	read-only
scope-primary-network-number	IP address	read-only
scope-primary-subnet-mask	IP address	read-only
scope-subnet-mask	IP address	read-only

Table 29-18 pre-packet-encode Scope Acceptability Data Items

Scope Acceptability Information Data Item	Value	Operation
scope-allow-bootp	int	read-only
scope-allow-dhcp	int	read-only
scope-allow-dynamic-bootp	int	read-only
scope-available-leases	int	read-only
scope-deactivated	int	read-only

Table 29-19 pre-packet-encode Scope DNS Update Data Items

Scope DNS Information Data Item	Value	Operation
scope-dns-forward-server-address	IP address	read-only
scope-dns-forward-zone-name	string	read-only
scope-dns-number-of-host-bytes	int	read-only
scope-dns-reverse-server-address	IP address	read-only
scope-dns-reverse-zone-name	string	read-only
scope-update-dns-enabled	int	read-only
scope-update-dns-for-bootp	int	read-only

post-packet-encode

The dictionaries available for **post-packet-encode** are request, response, and environment.

Note For DHCPv6 Reconfigure messages, there is no request dictionary (because Reconfigure is a server-initiated message). Thus, enabled extensions should check the response *msg-type* for ADVERTISE or REPLY or use **isValid** on the request to ensure that the Reconfigure message exists.

The server calls this extension point after encoding a packet, but before sending it to the client. The server can thereby examine and alter the packet before it sends the packet to the client, or the extension can cause the server to drop the packet (although the server might have made changes to its internal and on-disk data that will not be backed out if the packet is dropped).

The *client-packet* and *packet* data items were added to the response dictionary with similar behavior as described for the request dictionary in the "pre-packet-decode" section on page 29-28. Note that this extension point is the only one that can request the response *client-packet* or *packet* data items, because no packet exists at any other extension point. Also, the server does not process the changes made to the packet; the server simply sends the altered packet to the client.

If you enable sent packet logging, the server logs the packet after calling the extensions registered for this extension point. If tracing is configured with X>=3, the server also logs the packet before calling the extensions registered for this extension point (but only if at least one extension is registered).

pre-dns-add-forward

Note The **pre-dns-add-forward** extension point is deprecated and documentation for it removed. A future Network Registrar release may remove the extension point completely. Instead, use an earlier request extension point (such as **post-client-lookup**) to set the required options, such as the *client-fqdn* option.

post-send-packet

Use the **post-send-packet** extension point for any processing that you want to perform outside of the serious time constraints of the DHCP request-response cycle. After the server sends the packet to the client, it calls this extension point.

Note For DHCPv6 Reconfigure messages, there is no request dictionary (because Reconfigure is a server-initiated message). Thus, enabled extensions should check the response *msg-type* for ADVERTISE or REPLY or use **isValid** on the request to ensure that the Reconfigure message exists.

You can use all the request dictionary data items used for the **pre-packet-encode** extension point, as described in Table 29-13 on page 29-39, except that all the operations are read-only. You can use all the response dictionary data items used for the **pre-packet-encode** extension point, as described in the "Response Dictionary for pre-packet-encode" section on page 29-39.

environment-destructor

The **environment-destructor** extension point allows an extension to clean up any context that it might be holding. The only dictionary available for this extension point is environment.

The environment dictionary is available for all extension points called for a single client request. Because some extensions need to maintain context information between the multiple extension points called for a single client request, and because the server might drop requests at several places during processing, an extension cannot reliably release context that it might have created for that request. In these cases, the server calls the **environment-destructor** extension point whenever it is about to destroy the environment dictionary (provided that it created one in the first place).

Note The server calls all extensions attached to the **environment-destructor** extension point, even if the server did not call each extension at any other attachment point.

Handling Objects and Options

The following sections describe specialized ways of handling DHCP objects and options in extensions.

See Also

Using Object and Option Handling Methods
Options and Suboptions in C/C++

Using Object and Option Handling Methods

Extensions can call methods to set DHCP objects, and get, move to, put, and remove DHCP options. The methods are **setObject**, **getOption**, **moveToOption**, **putOption**, and **removeOption** methods in Tcl and C/C++.

These new callback methods are introduced in Network Registrar 7.0 primarily to provide support for DHCPv6. However, you can use the option-related functions for DHCPv4. In fact, it is recommended to use these methods for DHCPv4, because they provide richer access to options than the original **get**[**Bytes**], **get**[**Bytes**]**ByType**, **put**[**Bytes**], **put**[**Bytes**]**ByType**, and **remove**[**ByType**] methods.

Tip See the "DEX Request and Response Dictionary Methods" section on page C-27 for the different usages of some of these methods in C/C++.

For DHCPv6, you must use the **setObject**, **getOption**, **moveToOption**, **putOption**, and **removeOption** methods to access options. The **setObject** method was introduced for DHCPv6, because there can be many leases, prefixes, and messages (client or multiple relay) that an extension might want to access. So, **setObject** serves to set the context for subsequent calls to get request and response dictionary data items and options. When the server calls an extension, the context is set to the current lease (if applicable), prefix (if applicable), and client message. For example, when the server calls the **pre-packet-encode** extension point, only the request and response dictionary message context is valid, and set to the corresponding client message, because there is no lease or prefix associated with this extension point. However, when the server calls the **lease-state-change** extension point, it sets the response dictionary lease context to the lease on which the state has changed, sets the response dictionary prefix context to the prefix for the lease, and sets the request and response dictionary message context to the corresponding client message.

Options and Suboptions in C/C++

Some C/C++ extensions provide specialized argument type values to handle DHCP options and suboptions. The DEX_OPTION_* argument type specifies to use the standard DHCPv4 or DHCPv6 option definition set and not the definitions under an option (or suboption). So, DEX_OPTION_* means that the server looks up the option name or number in the standard DHCPv4 or DHCPv6 option definition set, whereas DEX_SUBOPTION_* means that the server looks up the suboption name or number of the current option definition (if any).

Thus, when you access options in DHCPv6, you often use DEX_OPTION_* followed by DEX_OPTION_* when options are encapsulated. You would use DEX_SUBOPTION when looking at vendor options. For DHCPv4, you would use DEX_OPTION at the client packet level, and then DEX_SUBOPTION perhaps one or more times, depending on the nesting level. Generally, only options have enterprise numbers or vendor names, but there is no prohibition on this. The option definition sets

determine what is valid (although one can walk off definitions, at which point everything is treated as binary bytes and thus it limits what is possible, and you cannot use the option or suboption names, but must use numbers).

The option ordering rules for the **getOption**, **moveToOption**, **putOption**, and **removeOption** methods are similar to the **request** expression syntax (see Table 25-1 on page 25-6). The ordering generally consists of:

- Preamble clause ([**parent** | **home**])
- Option clause (**option** [**vendor** | **enterprise**] [**instance**])
- Suboption clause (**suboption** [**vendor** | **enterprise**] [**instance**])
- End clause ([**instance-count** | **index-count** | [**index**] [**more**] **end**)

You can construct calls by using a preamble clause, followed by zero or more option clauses, followed by zero or more suboption clauses (which may themselves be followed by option and suboption clauses), followed by an end clause. Note that some things are possible only through a **get** method (such as **instance-count**, **index-count**, and **more**), and **move-to** can appear anywhere to move the context to the current option or suboption.

The option definition determines its data format, which can differ from what the older functions return for a specific option. To handle specific options:

- For the vendor class options (*v-i-vendor-class* [124] for DHCPv4 and *vendor-class* [16] for DHCPv6), if you ask for a specific instance of the option (instead of by enterprise-id or name), the only way to get the enterprise-id is to ask for the raw data (DEX_INDEX with DEX_RAW).
- For the DHCPv4 vendor options (*v-i-vendor-class* [124] and *v-i-vendor-opts* [125]), operating on the raw data (DEX_INDEX with DEX_RAW) only applies to an instance (preset value 0) of that option, not the entire option. There is no way to get the entire data for this option, which means that you cannot use **putOption** for the entire data. This is not an issue with the DHCPv6 vendor options, because these are separate options.
- If one of the DHCPv4 vendor options (124 or 125) is not formatted properly, the entire data is returned as a blob (if you asked for instance 0 and did not specify a particular enterprise-id). However, if an extension tries to use **putOption**, depending on the operation, that data might be appended to the existing data, and the result will be formatted incorrectly.
- For the vendor options, if there is no option, **putOption(pDict, "01:02", DEX_OPTION_NUMBER, 124, DEX_END)** fails because no enterprise-id is available. However, **putOption(pDict, "00:00:00:09:04:03:65:66:67", DEX_OPTION_NUMBER, 124, DEX_END)** will work because it is assumed that 00:00:00:09 is the enterprise-id and the bytes following it starting with 04 are the length of the option data of that enterprise-id. Note that the length byte is validated in this case, and **putOption** fails if it does not have the correct length. The recommended way to add this data is to use **putOption(pDict, "65:66:67", DEX_OPTION_NUMBER, 124, DEX_ENTERPRISE_ID, 9, DEX_END)**.



PART 6

Appendices, Glossary, and Index



APPENDIX A

Resource Records

Resource records comprise the data within a DNS zone. There is no fixed limit to the number of resource records a zone can own. In general, there can be zero, one, or more resource records of a given type. However, there are constraints on the number of certain types of records a zone can have.

All resource records have these required entries:

- **Name**—Name (host) that owns the record, such as example.com.
- **Class (not required for all formats)**—DNS supports only the IN (Internet) class of record.
- **TTL (time to live)**—Amount of time to store the record in cache, in seconds. If you do not include a TTL, Network Registrar uses the zone default TTL, defined in the SOA resource record.
- **Type**—Type of the record, such as A, NS, SOA, and MX. There are many types that various RFCs define, although ten or fewer are in common use.
- **Record data**—Data types whose format and meaning varies with record type.

Table A-1 lists all the resource record types Network Registrar supports. It provides the field syntax and the field descriptions, as well as how the fields are represented in the Network Registrar GUI.

Table A-1 Resource Records

Record	No.	Name	Syntax and Description	RFC
A	1	Host Address—Name-to-address mapping for the zone	*name ttl class* A *address* **Web UI**: Add or Edit Host for Zone page: Hostname, IP Address or Resource Records for Zone page: Name, TTL, Type, Data `nrcmd> zone example.com addRR host123 3600 IN A 192.168.40.123`	1035

Table A-1 Resource Records (continued)

Record	No.	Name	Syntax and Description	RFC
A6	38	IPv6 Address—(Experimental; use AAAA records instead)	*name ttl class* A6 *address* In the data, the suffix address is an IPv6 address encoded in network order (high-order octet first). There must be exactly enough octets in this field to contain a number of bits equal to 128 minus prefix length, with 0 to 7 leading pad bits to make this field an integral number of octets. Pad bits, if present, must be set to zero when loading a zone file and ignored on reception. For example: `2001:0:734c:c0::` **Web UI**: Resource Records for Zone page: Name, TTL, Type=A6, Data=*prefixlength suffixaddr prefixname*, with data in the form: `0 2345:00c1:ca11:0001:1234:5678:9abc:def0` `nrcmd> zone example.com addRR host456 A6 0 1345:c1:ca11:1:1234:5678:9abc:def0`	2874
AAAA	28	IPv6 Address—	*name ttl class* AAAA *address* Data is the IPv6 address format of eight sets of four hexadecimal digits, separated by colons. The first set of four digits is the high-order 16 bits of the address. You can omit leading zeros in sets and omit a value in a set if the value of the set is zero. **Web UI**: Resource Records for Zone page: Name, TTL, Type=AAAA, Data=*address* `nrcmd> zone example.com addRR host456 AAAA 1345:c1:ca11:1:1234:5678:9abc:def0`	1884
AFSDB	18	Andrew File System (AFS) Data Base—	*name ttl class* AFSDB *subtype hostname* Subtype is either 1—AFS cell database server, or 2—DCE authentication name server. Hostname is the domain name of host that has a server for the cell named by the owner. **Web UI**: Resource Records for Zone page: Name, TTL, Type=AFSDB, Data=*subtype hostname* `nrcmd> zone example.com addRR host4 AFSDB 1 AFSDBhost.example.com.`	1183

Table A-1 Resource Records (continued)

Record	No.	Name	Syntax and Description	RFC
CNAME	5	Canonical Name—Aliases or nicknames	*alias ttl class* CNAME *canonicalname* You cannot have any other resource records associated with a CNAME. Aliases are useful when you want the outside world to know a single, easily remembered name. You can also use aliases when a host changes its name. In that case, ensure that you have a CNAME pointer so that when people use the original name, it can be resolved to the newer one. **Web UI**: Resource Records for Zone page: Name=*alias*, TTL, Type=CNAME, Type, Data=*canonicalname* `nrcmd> zone example.com addRR host456 CNAME host1234`	1035
DHCID	49	Dynamic Host Configuration Identifier—(RFC 4701)	*name ttl class* DHCID *data* The DNS server uses this RR to allow DHCP clients and servers to update DNS automatically. This RR is not user-configurable. The data is the result of a one-way hash computation of the client message and the domain name. Sample RR output for an IPv6 address: `chi6.example.com IN DHCID ( AAIBY2/AuCccgoJbaxcQc9TUapptP691OjxfNuVAA2kjEA= )`	1035
HINFO	13	Host Info—Hardware and software information for the host	*name ttl class* HINFO *cpu os* Data is the hardware (CPU) and operating system. **Web UI**: Resource Records for Zone page: Name, TTL, Type=HINFO, Data=*cpu os* `nrcmd> zone example.com addRR host5 HINFO CPU1 OS2`	1035
ISDN	20	Integrated Services Digital Network (ISDN) Address—	*name ttl class* ISDN *ISDNnumber* [*subaddr*] Data is the ISDN number of the owner and Direct Dial In, if any, and an optional ISDN subaddress string **Web UI**: Resource Records for Zone page: Name, TTL, Type=ISDN, Data=*ISDNnumber* [*subaddr*] `nrcmd> zone example.com addRR host6 ISDN ISDN88888`	1183
MB	7	Mailbox Domain Name—	*name ttl class* MB *mbox* Data is the domain name of the host with the specified mailbox. **Web UI**: Resource Records for Zone page: Name, TTL, Type=MB, Data=*mbox* `nrcmd> zone example.com addRR host7 MB mailbox.example.com.`	1035

Table A-1 Resource Records (continued)

Record	No.	Name	Syntax and Description	RFC
MG	8	Mail Group Member—	*name ttl class* MG *mgroup* Data is the domain name of the mailbox group (mailing list). **Web UI**: Resource Records for Zone page: Name, TTL, Type=MG, Data=*mgroup* `nrcmd> zone example.com addRR host7 MG mbgroup.example.com.`	1035
MINFO	14	Mailbox Info—	*name ttl class* MINFO *respmbox errormbox* Data is the mailbox responsible for the mailing list, and the mailbox to receive error messages. **Web UI**: Resource Records for Zone page: Name, TTL, Type=MINFO, Data=*respmbox errormbox* `nrcmd> zone example.com addRR host7 MINFO resp.example.com. error.example.com.`	1035
MR	9	Mail Rename—	*name ttl class* MR *newmbox* Data is the mailbox name to rename the owner mailbox. **Web UI**: Resource Records for Zone page: Name, TTL, Type=MR, Data=*newmbox* `nrcmd> zone example.com addRR host7 MR renamemb.example.com.`	1035
MX	15	Mail Exchanger— Where to deliver the mail for a domain name	*name ttl class* MX *pref mxname* Data is the preference value (16-bit integer for the preference for the record, with lower values having preference), and the domain name of the mail exchanger for the owner. **Web UI**: Resource Records for Zone page: Name, TTL, Type=MX, Data=*pref mxname* `nrcmd> zone example.com addRR host8 MX 10 exchanger.example.com.`	1035

Table A-1 Resource Records (continued)

Record	No.	Name	Syntax and Description	RFC
NAPTR	35	Naming Authority Pointer—Produces a new domain label or Universal Resource Identifier (URI). You can then use DNS to look up services for many resource names that are not in domain name syntax.	*name ttl class* NAPTR *order pref flags serv regexp replace* • *order*—16-bit integer for the order in which to process the NAPTR records to ensure the correct ordering of rules, with low numbers processed before high numbers. • *pref*—16-bit unsigned integer for the order in which to process NAPTR records with equal *order* values, with low numbers processed before high numbers. • *flags*—Character-string containing flags to control aspects of rewriting and interpreting fields, single characters from the set [A-Z0-9] (not case-sensitive); the S, A and U flags denote a terminal lookup, the P flag says that the remainder of the application-side algorithm should be carried out protocol-specific. • *serv*—Valid protocols or services. • *regexp*—String containing a substitution expression applied to the original string held by the client to construct the next domain name to look up. (For common regex usage, see Table 5-4 on page 5-22). • *replace*—Next FQDN to query for NAPTR, SRV, or address records, depending on the value of the *flags* field. **Web UI**: Resource Records for Zone page: Name, State, TTL, Type=NAPTR, Data=*order pref flags service regexp replace* `nrcmd> zone 8.6.4.e164.arpa addRR 4.3.2.1.6.7.9 naptr 100 10 u sip+E2U /^.*$/sip:info@tele2.se/ .`	2915
NS	2	Name Server—Authoritative server for the zone	*name ttl class* NS *nameserver* Machines that provide name service must not reside in the owner domain. For each domain, you must have at least one NS record. NS records for a domain must exist in both the zone that delegates the domain and in the domain itself. NS record names must have an equivalent A record (they cannot point to an alias). **Web UI**: Add or Edit Zone page Nameservers: NS TTL, Add Nameserver `nrcmd> zone example.com addRR @ NS DNSserv2.example.com.`	1035

Table A-1 Resource Records (continued)

Record	No.	Name	Syntax and Description	RFC
NSAP	22	Network Service Access Point (NSAP) Address—	*name ttl class* NASP *NSAPaddr* Data is the *NSAPaddr*—Octet values assigned by the assigning authority, a character string of the type used in TXT and HINFO records (see RFC 1706). **Web UI**: Resource Records for Zone page: Name, TTL, Type=NSAP, Data=*NSAPaddr* `nrcmd> zone example.com addRR host10 NSAP 39840f80005a0000000001e13708002010726e00`	1706
PTR	12	Pointer—Reverse mapping	*name ttl class* PTR *dname* Data is the domain name of host having the reverse record indicated by the owner. PTR records are used for reverse mapping, specifically in the in-addr.arpa zones for translation of addresses to names. PTRs use official names, not aliases. The name in a PTR record is the local IP address portion of the reverse name. **Web UI**: Resource Records for Zone page: Name, State, TTL, Type=PTR, Data=*dname* `nrcmd> zone example.com addRR 45.40.168.192.in-addr.arpa. PTR host1234`	1035
RP	17	Responsible Person—	*name ttl class* RP *mbox txthost* Data is the domain name of the mailbox for the responsible person, and the domain name of host where TXT records exist. **Web UI**: Resource Records for Zone page: Name, TTL, Type=RP, Data=*mbox txthost* `nrcmd> zone example.com addRR host7 RP resp.example.com. text.example.com.`	1183
RT	21	Route Through—	*name ttl class* RT *pref intermediatehost* Data is the *pref*—16-bit integer for preference to give to this record among others of the same owner, and *intermediatehost*—domain name of the host serving as intermediate to reach the owner. **Web UI**: Resource Records for Zone page: Name, TTL, Type=RT, Data=*pref intermediatehost* `nrcmd> zone example.com addRR host7 RT 10 routthru.example.com.`	1183

Table A-1 Resource Records (continued)

Record	No.	Name	Syntax and Description	RFC
SOA	6	Start of Authority—Every zone must have a single SOA record	*name ttl class* SOA *primeserver hostmaster* (*serial refresh retry expire minimum*) **Web UI**: Add or Edit Zone page SOA Attributes: Serial Number, SOA TTL, Nameserver, Contact E-Mail, Secondary Refresh, Secondary Retry, Secondary Expire, Minimum TTL `nrcmd> zone example.com addRR @ 172800 IN SOA ns hostmaster 1 10800 3600 604800 86400`	1035
SRV	33	Service Location—	*name ttl class* SRV *priority weight port target* • *priority*—16-bit priority to give the record among the owner SRV records. • *weight*—16-bit load to give the record at the same priority level. • *port*—16-bit port on which to run the service. • *target*—Domain name of host running on the specified port. Administrators can use several servers for a single domain, move services between hosts with little difficulty, and designate some hosts as primary servers for a service and others as backups. Clients ask for a specific service or protocol for a domain and receive the names of any available servers. See Chapter 28, "Configuring DNS Update" for how this record affects Windows 2000 servers. **Web UI**: Resource Records for Zone page: Name, TTL, Type=SRV, Data=*priority weight port target* `nrcmd> zone example.com addRR host2 SRV 10 1 60 host7.example.com.`	2782
TXT	16	Text—	*name ttl class* TXT *textstring* Data is one or more text character strings that can contain any type of information. **Web UI**: Resource Records for Zone page: Name, TTL, Type=TXT, Data=*textstring* `nrcmd> zone example.com addRR host2 TXT "this message"`	1035

Table A-1 Resource Records (continued)

Record	No.	Name	Syntax and Description	RFC
WKS	11	Well Known Services—	*name ttl class* WKS *addr protocol servicelist* • *addr*—32-bit IP address. • *protocol*—8-bit IP protocol number, which can be TCP or UDP. • *servicelist*—Variable-length bit map in 8-bit multiples of services, which can be TIME, TELNET, FTP, or SMTP. **Web UI**: Resource Records for Zone page: Name, TTL, Type=WKS, Data=*addr protocol servicelist* `nrcmd> zone example.com addRR host8 WKS 192.168.40.56 TCP TELNET`	1035
X25	19	X.25 Address—	*name ttl class* X25 *PSDNaddr* Data is the character string of the Public Switch Data Network (PSDN) address in the X.121 numbering plan associated with the owner. **Web UI**: Resource Records for Zone page: Name, TTL, Type=X25, Data=*PSDNaddr* `nrcmd> zone example.com addRR host9 IN X25 311061700956`	1183



APPENDIX B

DHCP Options

DHCP provides a framework for passing configuration information to hosts on a TCP/IP network. Configuration parameters and other control information are carried in tagged data items that are stored in the options field of the DHCP message. The data items themselves are also called options.

This appendix contains DHCP options and BOOTP vendor extensions from RFC 2132, and includes the validation type for each option, as indicated in Table B-10 on page B-14.

This appendix also contains the standard Microsoft client options and several tables displaying the options sorted by categories.

Option Descriptions

The following sections describe the DHCP options in detail:

- RFC 1497 Vendor Extensions, page B-1
- IP Layer Parameters Per Host, page B-3
- IP Layer Parameters Per Interface, page B-4
- Link Layer Parameters Per Interface, page B-4
- TCP Parameters, page B-5
- Application and Service Parameters, page B-5
- DHCPv4 Extension Options, page B-8
- DHCPv6 Options, page B-10
- Microsoft Client Options, page B-13
- Options by Number, page B-14
- Options by Network Registrar Name, page B-18
- Option Validation Types, page B-24

RFC 1497 Vendor Extensions

Table B-1 on page B-2 lists the vendor extensions as defined in RFC 1497.

Table B-1 RFC 1497 Vendor Extension Options

Option Name	No.	Length	Description
Pad	0	1 octet	Causes the subsequent fields to align on word boundaries.
End	255	1 octet	End of valid information in the vendor field. Subsequent octets should be filled with the Pad options.
Subnet Mask	1	4 octets	Client subnet mask, as per RFC 950. If both the Subnet Mask and the Router option are specified in a DHCP reply, the Subnet Mask option must be first.
Time Offset	2	4 octets	Offset of the client subnet, in seconds, from Universal Time (UT). The offset is expressed as a twos-complement 32-bit integer. A positive offset indicates a location east of the zero meridian and a negative offset indicates a location west of the zero meridian.
Router	3	4 octet minimum; multiples of 4	List of IP addresses for routers on the client subnet. Routers should be in order of preference.
Time Server	4	4 octet minimum; multiples of 4	List of RFC 868 compliant time servers available to the client. Servers should be in order of preference.
Name Server Option	5	4 octet minimum; multiples of 4	List of IEN 116 name servers available to the client. Servers should be in order of preference.
Domain Name Server	6	4 octet minimum; multiples of 4	List of Domain Name System (STD 13, RFC 1035) name servers available to the client. Servers should be in order of preference.
Log Server	7	4 octet minimum; multiples of 4	List of MIT-LCS UDP log servers available to the client. Servers should be in order of preference.
Cookie Server	8	4 octet minimum; multiples of 4	List of RFC 865-compliant cookie servers available to the client. Servers should be in order of preference.
LPR Server	9	4 octet minimum; multiples of 4	List of RFC 1179-compliant line printer servers available to the client. Servers should be in order of preference.
Impress Server	10	4 octet minimum; multiples of 4	List of Imagen Impress servers available to the client. Servers should be in order of preference.
Resource Location Server	11	4 octet minimum; multiples of 4	List of RFC 887-compliant resource location servers available to the client. Servers should be in order of preference.
Host Name	12	1 octet minimum	Name of the client. The name may or may not be qualified with the local domain name. See RFC 1035 for the character set restrictions.
Boot File Size	13	2 octets	Number of 512-octet blocks in the default boot file.
Merit Dump File	14	1 octet minimum	Path name of a file to which the client core image should be placed in the event the client crashes. The path is formatted as a character string consisting of characters from the NVT ASCII character set.
Domain Name	15	1 octet minimum	Domain name that the client should use when resolving hostnames through the Domain Name System.
Swap Server	16	4 octets	IP address of the client swap server.

Table B-1 RFC 1497 Vendor Extension Options (continued)

Option Name	No.	Length	Description
Root Path	17	1 octet minimum	Path name that contains the client root disk. The path is formatted as a character string consisting of characters from the NVT ASCII character set.
Extensions Path	18	1 octet minimum	Uses a string to specify a file, retrievable through TFTP. The file contains information that can be interpreted in the same way as the 64-octet vendor-extension field within the BOOTP response, with these exceptions: the length of the file is unconstrained, and all references to instances of this option in the file are ignored.

IP Layer Parameters Per Host

Table B-2 lists the options that affect the operation of the IP layer on a per-host basis.

Table B-2 IP Layer Parameters Per Host Options

Option Name	No.	Length	Description
IP Forwarding Enable/Disable	19	1 octet	Specifies whether the client should configure its IP layer for packet forwarding. Values: 0=disable; 1=enable
Non-Local Source Routing Enable/Disable	20	1 octet	Specifies whether the client should configure its IP layer to allow forwarding of datagrams with non-local source routes. Values: 0=disable; 1=enable
Policy Filter	21	8 octet minimum; multiples of 8	Policy filters for non-local source routing. The filters consist of a list of IP addresses and masks that specify destination/mask pairs with which to filter incoming source routes. Any source-routed datagram whose next-hop address does not match one of the filters should be discarded by the client.
Maximum Datagram Reassembly Size	22	2 octets	Maximum size datagram that the client should be prepared to reassemble. Value: 576 minimum
Default IP Time-to-Live	23	1 octet	Default TTL that the client should use on outgoing datagrams. Values: 1 to 255
Path MTU Aging Timeout	24	4 octets	Timeout (in seconds) to use when aging Path MTU values (defined in RFC 1191).
Path MTU Plateau Table	25	2 octets minimum; multiples of 2	Table of MTU sizes to use when performing Path MTU Discovery as defined in RFC 1191. The table is formatted as a list of 16-bit unsigned integers, ordered from smallest to largest. Value: 68 minimum

IP Layer Parameters Per Interface

Table B-3 lists the options that affect the operation of the IP layer on a per-interface basis. A client can issue multiple requests, one per interface, to configure interfaces with their specific parameters.

Table B-3 IP Layer Parameters Per Interface Options

Option Name	No.	Length	Description
Interface MTU	26	2 octets	Maximum time to live to use on this interface.
All Subnets Are Local	27	1 octet	Specifies whether or not the client can assume that all subnets of the IP network to which the client is connected use the same MTU as the subnet of that network to which the client is directly connected. Values: 1=all subnets share same MTU; 0=some directly-connected subnets can have smaller MTUs
Broadcast Address	28	4 octets	Broadcast address in use on the client subnet.
Perform Mask Discovery	29	1 octet	Specifies whether or not the client should perform subnet mask discovery using ICMP. Values: 0=disable; 1=enable
Mask Supplier	30	1 octet	Specifies whether or not the client should respond to subnet mask requests using ICMP. Values: 0=do not respond; 1=respond
Perform Router Discovery	31	1 octet	Specifies whether or not the client should solicit routers using the Router Discovery mechanism defined in RFC 1256. Values: 0=disable; 1=enable
Router Solicitation Address	32	4 octets	Address to which the client should transmit router solicitation requests.
Static Route	33	8 octet minimum; multiples of 8	List of static routes that the client should install in its routing cache. If multiple routes to the same destination are specified, they are in descending order of priority. The routes consist of a list of IP address pairs. The first address is the destination address, and the second address is the router for the destination. The default route (0.0.0.0) is an illegal destination for a static route.

Link Layer Parameters Per Interface

Table B-4 lists the options that affect the operation of the data link layer on a per-interface basis.

Table B-4 Link Layer Parameters Per Interface Options

Option Name	No.	Length	Description
Trailer Encapsulation	34	1 octet	Specifies whether or not the client should negotiate the use of trailers (RFC 893) when using the ARP protocol. Values: 0=do not use; 1=use

Table B-4 Link Layer Parameters Per Interface Options (continued)

Option Name	No.	Length	Description
ARP Cache Timeout	35	4 octets	Timeout in seconds for ARP cache entries.
Ethernet Encapsulation	36	1 octet	Specifies whether or not the client should use Ethernet Version 2 (RFC 894) or IEEE 802.3 (RFC 1042) encapsulation if the interface is an Ethernet. Value: 0=use RFC 894 encapsulation; 1=use RFC 1042 encapsulation

TCP Parameters

Table B-5 lists the options that affect the operation of the TCP layer on a per-interface basis.

Table B-5 TCP Parameter Options

Option Name	No.	Length	Description
TCP Default TTL	37	1 octet	Default TTL that the client should use when sending TCP segments. Value: minimum 1
TCP Keepalive Interval	38	4 octets	Interval (in seconds) that the client TCP should wait before sending a keepalive message on a TCP connection. The time is specified as a 32-bit unsigned integer. A value of zero indicates that the client should not generate keepalive messages on connections unless specifically requested by an application. Value: 32-bit unsigned; 0=do not generate keepalive messages unless specifically requested.
TCP Keepalive Garbage	39	1 octet	Specifies the whether or not the client should send TCP keep-alive messages with an octet of garbage for compatibility with older implementations. Values: 0=do not send; 1=send

Application and Service Parameters

Table B-6 lists some miscellaneous options used to configure miscellaneous applications and services.

Table B-6 Application and Service Parameter Options

Option Name	No.	Length	Description
Network Information Service (NIS) Domain	40	1 octet minimum	Name of the client NIS domain. The domain is formatted as a character string consisting of characters from the NVT ASCII character set.
Network Information Service (NIS) Servers	41	4 octet minimum; multiples of 4	List of IP addresses indicating NIS servers available to the client. Servers should be in order of preference.

Table B-6 Application and Service Parameter Options (continued)

Option Name	No.	Length	Description
Network Time Protocol Servers	42	4 octet minimum; multiples of 4	List of IP addresses indicating NTP servers that are available to the client. Servers should be in order of preference.
Vendor-Specific Information	43	1 octet minimum	This option is used by clients and servers to exchange vendor-specific information. The information is an opaque object of *n* octets, presumably interpreted by vendor-specific code on the clients and servers. The definition of this information is vendor specific. The vendor is indicated in the *dhcp-class-identifier* option. Servers not equipped to interpret the vendor-specific information sent by a client must ignore it (although it can be reported). Clients that do not receive desired vendor-specific information should make an attempt to operate without it, although they can do so (and announce they are doing so) in a degraded mode. If a vendor potentially encodes more than one item of information in this option, then the vendor should encode the option using encapsulated vendor-specific options as described here. The encapsulated vendor-specific options field should be encoded as a sequence of code/length/value fields of identical syntax to the DHCP options field with these exceptions: • There should not be a magic cookie field in the encapsulated vendor-specific extensions field. • Codes other than 0 or 255 can be redefined by the vendor within the encapsulated vendor-specific extensions field, but should conform to the tag-length-value syntax defined in section 2. Code 255 (END), if present, signifies the end of the encapsulated vendor extensions, not the end of the vendor extensions field. If no code 255 is present, then the end of the enclosing vendor-specific information field is taken as the end of the encapsulated vendor-specific extensions field.
NetBIOS over TCP/IP Name Server	44	4 octet minimum; multiples of 4	List of RFC 1001/1002 NBNS name servers in order of preference.
NetBIOS over TCP/IP Datagram Distribution Server	45	4 octet minimum; multiples of 4	List of RFC 1001/1002 NBDD servers in order of preference.

Table B-6 Application and Service Parameter Options (continued)

Option Name	No.	Length	Description
NetBIOS over TCP/IP Node Type	46	1 octet	Allows NetBIOS over TCP/IP client, which are configured as described in RFC 1001/1002. Values: Single hexadecimal octet that identifies the client type: • 0x1=B-node (broadcast node) • 0x2=P-node (point-to-point node) • 0x4=M-node (mixed node) • 0x8=H-node
NetBIOS over TCP/IP Scope	47	1 octet minimum	NetBIOS over TCP/IP scope parameter for the client as specified in RFC 1001/1002.
X Window System Font Server	48	4 octet minimum; multiples of 4	List of X Window System Font servers available to the client. Servers should be in order of preference.
X Window System Display Manager	49	4 octet minimum; multiples of 4	List of IP addresses of systems that are running the X Window System Display Manager and are available to the client. Addresses should be in order of preference.
Network Information Service (NIS+) Domain	64	1 octet minimum	Name of the client NIS+ domain. The domain is formatted as a character string consisting of characters from the NVT ASCII character set.
Network Information Service (NIS+) Servers	65	4 octet minimum; multiples of 4	List of IP addresses indicating NIS+ servers available to the client. Servers should be in order of preference.
Mobile IP Home Agent	68	0 octets minimum; multiples of 4; expected, 4 octets (single home agent address)	List of IP addresses indicating mobile IP home agents available to the client. Agents should be in order of preference. Value: 32-bit address; 0=no home agents available
Simple Mail Transport Protocol (SMTP) Server	69	4 octet minimum; multiples of 4	List of SMTP servers available to the client. Servers should be in order of preference.
Post Office Protocol (POP3) Server	70	4 octet minimum; multiples of 4	List of POP3 servers available to the client. Servers should be in order of preference.
Network News Transport Protocol (NNTP) Server	71	4 octet minimum; multiples of 4	List of NNTP servers available to the client. Servers should be in order of preference.
World Wide Web (WWW) Server	72	4 octet minimum; multiples of 4	List of World Wide Web (WWW) servers available to the client. Servers should be in order of preference.

Table B-6 Application and Service Parameter Options (continued)

Option Name	No.	Length	Description
Finger Server	73	4 octet minimum; multiples of 4	List of Finger servers available to the client. Servers should be in order of preference.
Internet Relay Chat Server	74	4 octet minimum; multiples of 4	List of IRC servers available to the client. Servers should be in order of preference.
StreetTalk Server	75	4 octet minimum; multiples of 4	List of StreetTalk servers available to the client. Servers should be in order of preference.
StreetTalk Directory Assistance (STDA) Server	76	4 octet minimum; multiples of 4	List of STDA servers available to the client. Servers should be in order of preference.

DHCPv4 Extension Options

Table B-7 lists the DHCPv4 extension options.

Table B-7 DHCPv4 Extensions

Option Name	No.	Length	Description
Requested IP Address	50	4 octets	Used in a client request (DHCPDISCOVER) to allow the client to request that a particular IP address be assigned.
IP Address Lease Time	51	4 octets	Used in a client request (DHCPDISCOVER or DHCPREQUEST) to allow the client to request a lease time for the IP address. In a server reply (DHCPOFFER), a DHCP server uses this option to specify the lease time it is willing to offer. Value: seconds, as 32-bit unsigned integer
Option Overload	52	1 octet	Indicates that the DHCP sname or file fields are being overloaded by using them to carry DHCP options. A DHCP server inserts this option if the returned parameters will exceed the usual space allotted for options. If this option is present, the client interprets the specified additional fields after it concludes interpretation of the standard option fields. Values: 1=file field is used to hold options; 2=sname field is used to hold options; 3=both fields are used to hold options
DHCP Message Type	53	1 octet	Used to convey the type of DHCP message. The preset value is 1 (DHCPDISCOVER). Values: 1=DHCPDISCOVER; 2=DHCPOFFER; 3=DHCPREQUEST; 4=DHCPDECLINE; 5=DHCPACK; 6=DHCPNAK; 7=DHCPRELEASE; 8=DHCPINFORM; 13=LEASEQUERY

Table B-7 DHCPv4 Extensions (continued)

Option Name	No.	Length	Description
Server Identifier	54	4 octets	Used in DHCPOFFER and DHCPREQUEST messages, and can optionally be included in the DHCPACK and DHCPNAK messages. DHCP servers include this option in the DHCPOFFER in order to allow the client to distinguish between lease offers. DHCP clients use the contents of the server identifier field as the destination address for any DHCP messages unicast to the DHCP server. DHCP clients also indicate which of several lease offers is being accepted by including this option in a DHCPREQUEST message. The identifier is the IP address of the selected server.
Parameter Request List	55	1 octet minimum	Used by a DHCP client to request values for specified configuration parameters. The list of requested parameters is specified as n octets, where each octet is a valid DHCP option code as defined in this document. The client can list the options in order of preference. The DHCP server does not have to return the options in the requested order, but must try to insert the options in the order that the client requested.
Message	56	1 octet minimum	Used by a DHCP server to provide an error message to a DHCP client in a DHCPNAK message in the event of a failure. A client can use this option in a DHCPDECLINE message to indicate why the client declined the offered parameters. The message consists of *n* octets of NVT ASCII text, which the client can display on an available output device.
Maximum DHCP Message Size	57	2 octets	Maximum-length DHCP message that a server is willing to accept. The length is specified as an unsigned 16-bit integer. A client can use the maximum DHCP message size option in DHCPDISCOVER or DHCPREQUEST messages, but should not use the option in DHCPDECLINE messages. Value: 576 minimum
Renewal (T1) Time Value	58	4 octets	Time interval from address assignment until the client transitions to RENEWING state. Value: seconds, as 32-bit unsigned integer
Rebinding (T2) Time Value	59	4 octets	Time interval from address assignment until the client transitions to REBINDING state. Value: seconds, as 32-bit unsigned integer
Vendor Class Identifier	60	1 octet minimum	Used by DHCP clients to optionally identify the vendor type and configuration of a DHCP client. The information is a string of *n* octets, interpreted by servers. Vendors can choose to define specific vendor class identifiers to convey particular configuration or other identification information about a client. For example, the identifier can encode the client hardware configuration. Servers not equipped to interpret the class-specific information sent by a client must ignore it (although it can be reported). Servers that respond should only use option 43 to return the vendor-specific information to the client.

Table B-7 DHCPv4 Extensions (continued)

Option Name	No.	Length	Description
Client-Identifier	61	2 octet minimum	Used by DHCP clients to specify their unique identifier. DHCP servers use this value to index their database of address bindings. This value is expected to be unique for all clients in an administrative domain. DHCP servers should treat identifiers as opaque objects. The client identifier can consist of type-value pairs similar to the *htype*/*chaddr* fields. For instance, it can consist of a hardware type and hardware address. In this case, the type field should be one of the ARP hardware types defined in STD2. A hardware type of 0 (zero) should be used when the value field contains an identifier other than a hardware address (for example, a fully qualified domain name). For correct identification of clients, each client-identifier must be unique among the client-identifiers used on the subnet to which the client is attached. Vendors and system administrators are responsible for choosing client-identifiers that meet this requirement for uniqueness.
TFTP Server Name	66	1 octet minimum	Identifies a TFTP server when the *sname* field in the DHCP header has been used for DHCP options.
Bootfile Name	67	1 octet minimum	Identifies a bootfile when the file field is the DHCP header that has been used for DHCP options.
Relay Agent Information	82		Identifies the DHCP relay agent information.
iSNS	83	14 bytes minimum	Identifies the Internet Storage Name Service (see RFC 4174)
BCMS Controller Domain	88	Variable	List of Broadcast and Multicast Service (BCMS) controller domains (see RFC 4280)
BCMS Address	89	4 octets minimum	List of IP addresses for the BCMS controller (see RFC 4280)
Lease Query Client Last Transaction Time	91	4 octets	Time of the most recent access of the client sending a DHCPLEASEQUERY (see RFC 4388).
Lease Query Associated IP Addresses	92	4 octets minimum	All IP addresses associated with the client specified in a particular DHCPLEASEQUERY message (see RFC 4388).

DHCPv6 Options

Table B-8 on page B-11 lists the DHCPv6 options, along with their defined data types. All the option packets include at least an option length (option-len) and a variable length data field. There can also be additional parameter settings, as described in the table. **Mod** in the Description column indicates that the option is modifiable. Many of these options are described in RFC 3315.

Table B-8 DHCPv6 Options

Network Registrar Name (Type)	No.	Description (Mod=modifiable)
client-identifier AT_BLOB	1	DUID identifying a client between a client and a server.
server-identifier AT_BLOB	2	DUID identifying a server between a client and a server.
ia-na AT_BLOB	3	Nontemporary Addresses option with the associated parameters and addresses. Parameters are the unique ID, time the client contacts the addresses in the IA to extend the lifetime, and time the client contacts any available server to extend the lifetime of the addresses.
ia-ta AT_BLOB	4	Temporary Addresses option with the associated parameters and addresses.
iaaddr AT_BLOB	5	IPv6 addresses associated with an IA_NA or IA_TA. (The IAADRR must be encapsulated in the options field of an IA_NA or IA_TA option.) The IAADDR option includes preferred and valid lifetime fields, and the options field that encapsulates the options specific to this address.
oro AT_SHORT	6	Option Request option (ORO) that identifies a list of options in a message between a client and a server. A client can include this option in a Solicit, Request, Renew, Rebind, Confirm, or Information-request message to inform the server about options the client wants from the server. A server can include this option in a Reconfigure message to indicate which option updates the client should request.
preference AT_INT8	7	A server sends this option to a client to affect what server the client selects (Mod).
elapsed-time AT_SHORT	8	A client sends this option to a server to indicate how long the client has been trying to complete a message exchange (Mod).
relay-message AT_BLOB	9	DHCP message in a Relay-forward or Relay-reply message.
auth AT_BLOB	11	Authenticates the identity and contents of a DHCP message. The parameters are the authentication protocol, the authentication algorithm, the replay detection method (RDM), and the authentication information.
server-unicast AT_IP6ADDR	12	The server sends this option to a client to indicate that the client can unicast messages to the server.
status-code AT_BLOB	13	Returns a status indication related to the DHCP message or option in which it appears. The parameters are the status code and status message.
rapid-commit AT_ZEROSIZE	14	Signals use of the two-message exchange for address assignment.
user-class AT_TYPECNT	15	Clients use this option to identify the type or category of user or applications it represents. A zero type count value field followed by user data (as a blob).
vendor-class AT_VENDOR_CLASS	16	Clients use this option to identify the vendor that manufactured the hardware on which they are running.

Table B-8 DHCPv6 Options (continued)

Network Registrar Name (Type)	No.	Description (Mod=modifiable)
vendor-opts AT_VENDOR_OPTS	17	Clients and servers use this option to exchange vendor-specific information. The enterprise ID for the CableLabs vendor is 4491; the suboptions for CableLabs are listed in Table C-4 on page C-7.
interface-id AT_BLOB	18	Relay agents use this option to identify the interface on which the client message is received.
reconfigure-message AT_INT8	19	The server includes this in a Reconfigure message to indicate whether the client should respond with a Renew or Information-request message.
reconfigure-accept AT_ZEROSIZE	20	Clients use this option to announce to the server whether the client is willing to accept Reconfigure messages.
sip-servers-name AT_DNSNAME	21	Domain names of the SIP outbound proxy servers for the client (Mod). See RFC 3319.
sip-servers-address AT_IP6ADDR	22	IPv6 addresses of the SIP outbound proxy servers for the client (Mod).
dns-servers AT_IP6ADDR	23	IPv6 addresses of DNS recursive name servers (Mod).
domain-list AT_DNSNAME	24	Domain names in the domain search list (Mod).
ia-pd AT_BLOB	25	IPv6 prefix delegation identity association and its associated parameters and prefixes. Parameters are the unique ID, time the client contacts the addresses in the IA to extend the lifetime, and time the client contacts any available server to extend the lifetime of the addresses.
iaprefix AT_BLOB	26	IPv6 prefixes associated with an IA_PD. The prefix must be encapsulated in the options field of an IA_PD option. Parameters are the valid and preferred lifetimes, prefix length, and the prefix.
nis-servers AT_IP6ADDR	27	List of IPv6 addresses of Network Information Service (NIS) servers available to the client (see RFC 3898) (Mod).
nisp-servers AT_IP6ADDR	28	List of IPv6 addresses of NIS+ servers available to the client (Mod).
nis-domain-name AT_DNSNAME	29	Conveys the NIS domain name to the client (Mod).
nisp-domain-name AT_DNSNAME	30	Conveys the NIS+ domain name to the client (Mod).
sntp-servers AT_IP6ADDR	31	List of Simple Network Time Protocol (SNTP) servers available to the client (see RFC 4075) (Mod).
info-refresh-time AT_TIME	32	Sets an upper bound for how long a client should wait before refreshing DHCPv6 information (see RFC 4242) (Mod).
bcms-server-d AT_DNSNAME	33	List of BCMS controller domains (see RFC 4280) (Mod).
bcms-server-a AT_IP6ADDR	34	List of IPv6 addresses for the Broadcast and Multicast Service (BCMS) controller (see RFC 4280) (Mod).

Table B-8 DHCPv6 Options (continued)

Network Registrar Name (Type)	No.	Description (Mod=modifiable)
geoconf-civic AT_BLOB	36	DHCP civic addresses configuration (Mod).
remote-id AT_BLOB	37	Relay agents that terminate switched or permanent circuits can add this option to identify remote hosts (see RFC 4649) (Mod).
relay-agent-subscriber-id AT_BLOB	38	Allows assignment and activation of subscriber-specific actions (see RFC 4580) (Mod).
client-fqdn AT_BLOB	39	DHCP client FQDN (Mod).
new-posix-timezone AT_BLOB	41	POSIX time zone, for example, EST5EDT4, M3.2.0/02:00,M11.1.0/02:00.
new-tzdb-timezone AT_BLOB	42	POSIX time zone database name, for example, Europe/Zurich.
ero AT_BLOB	43	Relay agent Echo Request option to inform the server of the list of relay agent options to echo back.
lq-query AT_BLOB	44	Used only in a LEASEQUERY message; identifies the query being performed. The option includes the query type, link-address (or 0::0), and options to provide data needed for the query.
client-data AT_BLOB	45	Encapsulates the data for a single client on a single link in a LEASEQUERY-REPLY message.
clt-time AT_TIME	46	Client last transaction time encapsulated in the *client-data* option; identifies how long ago the server last communicated with the client (in seconds).
lq-relay-data AT_BLOB	47	Used only in a LEASEQUERY-REPLY message; provides the relay agent data used when the client last communicated with the server.
lq-client-link AT_IP6ADDR	48	Used only in a LEASEQUERY-REPLY message; identifies the links on which the client has one or more bindings. It is used in reply to a query when no link-address was specified and the client is found to be on more than one link.

Microsoft Client Options

Table B-9 lists the standard Microsoft client options.

Table B-9 Microsoft DHCP Client Options

Option Name	No.	Description
dhcp-lease-time	51	14 days
domain-name	15	A domain name such as cisco.com
domain-name-servers	6	IP address of the name servers
netbios-name-servers	44	WINS server address

Table B-9 Microsoft DHCP Client Options (continued)

Option Name	No.	Description
netbios-node-type	46	Identifies the NetBIOS client type; note that Network Registrar displays a warning if it is not present
routers	3	IP address of the router for this subnet

Option Tables

The following tables display the DHCP options in various ways. They show the options sorted numerically, by Network Registrar name, and by category.

DHCP options have a prescribed format and allowed values for their option parameters. Table B-10 lists each DHCP option and parameter type (in the Validation column). The parameter formats and allowed values come from the DHCP and Internet RFCs. All the DHCP options appear, but clients control only some, and the CLI only others.

Options by Number

Table B-10 shows the DHCPv4 options sorted by option number, and includes the validation type. (See Table B-12 on page B-24 for details on the option validation types found in the Validation column.) A **0+** in the Comments column means a repeat count of zero or more occurrences, **1+** means one or more occurrences, **2n** means multiple occurrences in multiples of 2. Comments also indicate whether the option includes suboptions, and, if so, how many, and whether the option is nonmodifiable (**NM**).

Tip For the syntax for adding more complex option data values for suboptions, see the "Adding Complex Values for Suboptions" section on page 21-7.

Table B-10 DHCPv4 Options by Number

No.	Network Registrar Name	Protocol Name	Validation	Comments
0	pad	Pad	AT_NOLEN	NM
1	subnet-mask	Subnet Mask	AT_IPADDR	
2	time-offset	Time Offset	AT_STIME	Replaced by tz-options (RFC 4833)
3	routers	Router	AT_IPADDR	1+
4	time-servers	Time Server	AT_IPADDR	1+
5	name-servers	Name Server	AT_IPADDR	1+
6	domain-name-servers	Domain Server	AT_IPADDR	1+
7	log-servers	Log Server	AT_IPADDR	1+
8	cookie-servers	Quotes Server	AT_IPADDR	1+
9	lpr-servers	LPR Server	AT_IPADDR	1+
10	impress-servers	Impress Server	AT_IPADDR	1+

Table B-10 DHCPv4 Options by Number (continued)

No.	Network Registrar Name	Protocol Name	Validation	Comments
11	resource-location-servers	RLP Server	AT_IPADDR	1+
12	host-name	Host Name	AT_NSTRING	
13	boot-size	Boot File Size	AT_SHORT	
14	merit-dump	Merit Dump File	AT_NSTRING	
15	domain-name	Domain Name	AT_NSTRING	
16	swap-server	Swap Server	AT_IPADDR	
17	root-path	Root Path	AT_NSTRING	
18	extensions-path	Extension File	AT_NSTRING	
19	ip-forwarding	Forward On/Off	AT_BOOL	
20	non-local-source-routing	SrcRte On/Off	AT_BOOL	
21	policy-filters	Policy Filter	AT_IPADDR	2n
22	max-dgram-reassembly	Maximum DG Assembly	AT_SHORT	
23	Default-ip-ttl	Default IP TTL	AT_RANGEBYTE	
24	path-mtu-aging-timeout	MTU Timeout	AT_TIME	
25	path-mtu-plateau-tables	MTU Plateau	AT_RANGESHORT	2n
26	interface-mtu	MTU Interface	AT_RANGESHORT	
27	all-subnets-local	MTU Subnet	AT_BOOL	
28	broadcast-address	Broadcast Address	AT_IPADDR	
29	perform-mask-discovery	Mask Discovery	AT_BOOL	
30	mask-supplier	Mask Supplier	AT_BOOL	
31	router-discovery	Router Discovery	AT_BOOL	
32	router-solicitation-address	Router Request	AT_IPADDR	
33	static-routes	Static Route	AT_IPADDR	2n
34	trailer-encapsulation	Trailers	AT_BOOL	
35	arp-cache-timeout	ARP Timeout	AT_TIME	
36	ieee802.3-encapsulation	Ethernet	AT_BOOL	
37	default-tcp-ttl	Default TCP TTL	AT_RANGEBYTE	
38	tcp-keepalive-interval	Keepalive Time	AT_TIME	
39	tcp-keepalive-garbage	Keepalive Data	AT_BOOL	
40	nis-domain	NIS Domain	AT_STRING	
41	nis-servers	NIS Servers	AT_IPADDR	1+
42	ntp-servers	NTP Servers	AT_IPADDR	1+
43	vendor-encapsulated-options	Vendor Specific	AT_BLOB	NM
44	netbios-name-servers	NetBIOS Name Server	AT_IPADDR	1+

Table B-10 DHCPv4 Options by Number (continued)

No.	Network Registrar Name	Protocol Name	Validation	Comments
45	netbios-dd-servers	NetBIOS Distribution Server	AT_IPADDR	1+
46	netbios-node-type	NetBIOS Node Type	AT_RANGEBYTE	
47	netbios-scope	NetBIOS Scope	AT_NSTRING	
48	font-servers	X Window Font	AT_IPADDR	1+
49	x-display-managers	X Window Manager	AT_IPADDR	1+
50	dhcp-requested-address	Address Request	AT_IPADDR	
51	dhcp-lease-time	Address Time	AT_TIME	NM
52	dhcp-option-overload	Overload	AT_OVERLOAD	
53	dhcp-message-type	DHCP Message Type	AT_MESSAGE	NM (See the DHCP Message Type option in Table B-7 on page B-8)
54	dhcp-server-identifier	DHCP Server ID	AT_IPADDR	
55	dhcp-parameter-request-list	Parameter List	AT_INT8	0+
56	dhcp-message	DHCP Message	AT_NSTRING	NM
57	dhcp-max-message-size	DHCP Maximum Message Size	AT_SHORT	NM
58	dhcp-renewal-time	Renewing Time	AT_TIME	NM
59	dhcp-rebinding-time	Rebinding Time	AT_TIME	NM
60	dhcp-class-identifier	Class Identifier	AT_NSTRING	
61	dhcp-client-identifier	Client Identifier	AT_BLOB	
62	netwareip-domain	NetWare/IP Domain	AT_NSTRING	
63	netwareip-information	NetWare/IP Option	AT_BLOB	
64	nis+-domain	NIS Domain Name	AT_NSTRING	
65	nis+-servers	NIS Server Address	AT_IPADDR	1+
66	tftp-server	TFTP Server Name	AT_NSTRING	
67	boot-file	Bootfile Name	AT_NSTRING	
68	mobile-ip-home-agents	Mobile IP Home Agent	AT_IPADDR	0+
69	smtp-servers	SMTP Server	AT_IPADDR	1+
70	pop3-servers	POP3 Server	AT_IPADDR	1+
71	nntp-servers	NNTP Server	AT_IPADDR	1+
72	www-servers	WWW Server	AT_IPADDR	1+
73	finger-servers	Finger Server	AT_IPADDR	1+
74	irc-servers	IRC Server	AT_IPADDR	1+
75	streettalk-servers	StreetTalk Server	AT_IPADDR	1+

Table B-10 DHCPv4 Options by Number (continued)

No.	Network Registrar Name	Protocol Name	Validation	Comments
76	streettalk-directory-assistance-servers	STDA Server	AT_IPADDR	1+
77	dhcp-user-class-id	User Class ID	AT_TYPECNT	Suboptions (2)
78	slp-directory-agent	Service Location Protocol Directory Agent	AT_BLOB	Suboptions (2)
79	slp-service-scope	SLP Service Scope	AT_BLOB	Suboptions (2)
80	rapid-commit	Rapid Commit	AT_ZEROSIZE	
81	client-fqdn	Client FQDN	AT_BLOB	Suboptions (4)
82	relay-agent-info	Relay Agent Information	AT_BLOB	For suboptions, see Table C-3 on page C-2
83	iSNS	Internet Storage Name Service (RFC4174)	AT_BLOB	Suboptions (7)
85	nds-servers	NDS Servers	AT_IPADDR	1+
86	nds-tree	NDS Tree Name	AT_NSTRING	
87	nds-context	NDS Context	AT_NSTRING	
88	bcms-servers-d	BCMS Controller Domain (RFC 4280)	AT_DNSNAME	1+
89	bcms-servers-a	BCMS Address	AT_IPADDR	1+
90	authentication	Authentication	AT_BLOB	Suboptions (5)
91	client-last-transaction-time	Lease Query Client Last Transaction Time	AT_TIME	
92	associated-ip	Lease Query Associated IP Addresses	AT_IPADDR	1+
93	pxe-client-arch		AT_SHORT	
94	pxe-client-network-id		AT_BLOB	Suboptions (2)
95	ldap-url		AT_NSTRING	
97	pxe-client-machine-id		AT_BLOB	Suboptions (2)
98	user-auth		AT_NSTRING	
99	geoconf-civic	Civic Addresses Configuration	AT_BLOB	
100	tz-posix	IEEE 1003.1 String	AT_NSTRING	
101	tz-database	Time Zone Database	AT_NSTRING	
112	netinfo-parent-server-addr		AT_IPADDR	
113	netinfo-parent-server-tag		AT_NSTRING	
114	initial-url		AT_NSTRING	
116	auto-configure	Autoconfiguration	AT_RANGEBYTE	

Table B-10 DHCPv4 Options by Number (continued)

No.	Network Registrar Name	Protocol Name	Validation	Comments
117	name-service-search	Name Service Search	AT_SHORT	1+
118	subnet-selection	Subnet Selection	AT_IPADDR	
119	domain-search	Domain Search	AT_BLOB	
120	sip-servers	SIP Servers	AT_BLOB	Suboptions (2)
121	classless-static-route	Classless Static Route	AT_BLOB	
122	cablelabs-client-configuration)	CableLabs Client Configuration	AT_BLOB	Suboptions (10) (see cablelabs-client-configuration, page C-3)
123	geo-conf	GeoConf Option	AT_BLOB	
124	v-i-vendor-class	Vendor-Identifying Vendor Class	AT_VENDOR_CLASS	NM
125	v-i-vendor-info	Vendor-Identifying Vendor-Specific Info	AT_VENDOR_OPTS	See also the cablelabs-125 suboptions in Table C-3 on page C-2
128	mcns-security-server	--	AT_IPADDR	
161	cisco-leased-ip	Cisco	AT_IPADDR	
162	cisco-client-requested-host-name	Cisco	AT_NSTRING	
163	cisco-client-last-transaction-time	Cisco	AT_INT	
185	vpn-id	VPN Identifier	AT_BLOB	NM: Suboptions (2)
220	subnet-alloc	Subnet Allocation	AT_BLOB	Suboptions (5)
221	cisco-vpn-id	Cisco VPN Identifier	AT_BLOB	Suboptions (2)
251	cisco-auto-configure	Cisco Autoconfiguration	AT_RANGEBYTE	
255	end	End	AT_NOLEN	NM

Options by Network Registrar Name

Table B-11 lists the DHCP options by Network Registrar name. (For each option validation type, cross-reference it by number to Table B-10 and check the Validation column.)

Table B-11 DHCP Options by Network Registrar Name

Network Registrar Name	No.	Option Name	Category
all-subnets-local	27	All Subnets Are Local	Interface
arp-cache-timeout	35	ARP Cache Timeout	Interface
associated-ip	92	Lease Query Associated IP	DHCPv4
auth	11	Authentication	DHCPv6
authentication	90	Authentication	--
auto-configuration	116	Auto-Configuration	DHCPv4
bcms-server-a	34	BCMS Address v6	DHCPv6
bcms-servers-a	89	BCMS Address	DHCPv4
bcms-server-d	33	BCMS Controller Domain v6	DHCPv6
bcms-servers-d	88	BCMS Controller Domain	DHCPv4
boot-file	67	Bootfile Name	BOOTP
boot-size	13	Boot File Size	BOOTP
broadcast-address	28	Broadcast Address	Interface
cablelabs-client-configuration	122	CableLabs Client Configuration	Interface
cisco-autoconfigure	251	Cisco Autoconfiguration	DHCPv4
cisco-client-last-transaction-time	163	Cisco Client Last Transaction Time	DHCPv4
cisco-client-requested-host-name	162	Cisco Client Requested Host Name	DHCPv4
cisco-leased-ip	161	Cisco Leased IP Address	DHCPv4
cisco-vpn-id	221	Cisco VPN Identifier	DHCPv4
classless-static-route	121	Classless Static Route	DHCPv4
client-data	45	Leasequery Reply Client Data	DHCPv6
client-fqdn	81	DHCP Client FQDN	DHCPv4
client-fqdn	39	DHCP Client FQDN	DHCPv6
client-identifier	1	Client Identifier	DHCPv6
client-last-transaction-time	91	Leasequery Client Last Transaction Time	DHCPv4
clt-time	46	Leasequery Client Last Transaction Time	DHCPv6
cookie-servers	8	Cookie Server	BOOTP
default-ip-ttl	23	Default IP Time-to-Live	Host IP
default-tcp-ttl	37	TCP Default TTL	Interface
dhcp-class-identifier	60	Vendor Class Identifier	DHCPv4
dhcp-client-identifier	61	Client-Identifier	Basic
dhcp-lease-time	51	IP Address Lease Time	Lease Information, MS DHCP Client

Table B-11 DHCP Options by Network Registrar Name (continued)

Network Registrar Name	No.	Option Name	Category
dhcp-max-message-size	57	Maximum DHCP Message Size	DHCPv4
dhcp-message-type	53	DHCP Message Type	DHCPv4
dhcp-message	56	Message	DHCPv4
dhcp-option-overload	52	Option Overload	DHCPv4
dhcp-parameter-request-list	55	Parameter Request List	DHCPv4
dhcp-rebinding-time	59	Rebinding (T2) Time Value	Lease Information, MS DHCP Client
dhcp-renewal-time	58	Renewing (T1) Time Value	Lease Information, MS DHCP Client
dhcp-requested-address	50	Requested IP Address	DHCPv4
dhcp-server-identifier	54	Server Identifier	DHCPv4
dhcp-user-class-id	77	User Class ID	DHCPv4
dns-servers	23	DNS Recursive Name Server	DHCPv6
domain-list	24	Domain Search List	DHCPv6
domain-name	15	Domain Name	Basic, MS DHCP Client
domain-name-servers	6	Domain Name Server	Basic, MS DHCP Client
domain-search	119	Domain Search	DHCPv4
elapsed-time	8	Elapsed Time	DHCPv6
end	255	End	--
ero	43	Relay Agent Echo Request Option	DHCPv6
extensions-path	18	Extensions Path	BOOTP
finger-servers	73	Finger Server	Servers
font-servers	48	X Window System Font Server	Servers
geo-conf	123	GeoConf	DHCPv4
geoconf-civic	99	Civic Addresses Configuration	DHCPv4
geoconf-civic	36	Civic Addresses Configuration	DHCPv6
host-name	12	Host Name	Basic
ia-na	3	Identity Association for Nontemporary Addresses	DHCPv6
ia-pd	25	Prefix Delegation	DHCPv6
ia-ta	4	Identity Association for Temporary Addresses	DHCPv6
iaaddr	5	IA Address	DHCPv6
iaprefix	26	IA Prefix	DHCPv6
ieee802.3-encapsulation	36	Ethernet Encapsulation	Interface
impress-servers	10	Impress Server	BOOTP

Table B-11 DHCP Options by Network Registrar Name (continued)

Network Registrar Name	No.	Option Name	Category
info-refresh-time	32	Information Refresh Time	DHCPv6
interface-id	18	Interface Identifier	DHCPv6
interface-mtu	26	Interface MTU	Interface
ip-forwarding	19	IP Forwarding Enable/Disable	Host IP
irc-servers	74	IRC Server	Servers
isns	83	iSNS	DHCPv4
log-servers	7	Log Server	Servers
lpr-servers	9	LPR Server	Servers
lq-client-link	48	Leasequery Client Link Reply	DHCPv6
lq-query	44	Leasequery	DHCPv6
lq-relay-data	47	Leasequery Relay Agent Reply	DHCPv6
mask-supplier	30	Mask Supplier	Interface
max-dgram-reassembly	22	Maximum Datagram Reassembly Size	Host IP
mcns-security-server	128	--	Servers
merit-dump	14	Merit Dump File	BOOTP
mobile-ip-home-agents	68	Mobile IP Home Agent	Servers
name-servers	5	Name Server	BOOTP
name-service-search	117	Name Service Search	DHCPv4
nds-context	87	NDS Context	NetWare Client
nds-servers	85	NDS Servers	NetWare Client
nds-tree	86	NDS Tree Name	NetWare Client
netbios-dd-servers	45	NetBIOS over TCP/IP Datagram Distribution Server	WINS/NetBIOS
netbios-name-servers	44	NetBIOS over TCP/IP Name Server	WINS/NetBIOS, MS DHCP Client
netbios-node-type	46	NetBIOS over TCP/IP Node Type	WINS/NetBIOS, MS DHCP Client
netbios-scope	47	NetBIOS over TCP/IP Scope	WINS/NetBIOS, MS DHCP Client
netwareip-domain	62	NetWare/IP Domain Name	NetWare Client
netwareip-information	63	NetWare/IP Information	NetWare Client
new-posix-timezone	41	POSIX time zone string	DHCPv6
new-tzdb-timezone	42	POSIX time zone database name	DHCPv6
nis+-domain	64	NIS+ Domain	Servers
nis+-servers	65	Network Information Service (NIS+) Servers	Servers

Table B-11 DHCP Options by Network Registrar Name (continued)

Network Registrar Name	No.	Option Name	Category
nis-domain	40	NIS Domain	Servers
nis-domain-name	29	NIS Domain Name	DHCPv6
nis-servers	41	Network Information Service (NIS) Servers	Servers
nis-servers	27	NIS Servers	DHCPv6
nisp-domain-name	30	NIS+ Domain Name	DHCPv6
nisp-servers	28	NIS+ Servers	DHCPv6
nntp-servers	71	NNTP Server	Servers
non-local-source-routing	20	Non-Local Source Routing	Host IP
ntp-servers	42	NTP Servers	Servers
option-time	8	Option Time	DHCPv6
oro	6	Option Request Option	DHCPv6
pad	0	Pad	--
path-mtu-aging-timeout	24	Path MTU Aging Timeout	Host IP
path-mtu-plateau-tables	25	Path MTU Plateau Table	Host IP
perform-mask-discovery	29	Perform Mask Discovery	Interface
policy-filters	21	Policy Filter	Host IP
pop3-servers	70	POP3 Server	Servers
preference	7	Preference	DHCPv6
rapid-commit	80	Rapid Commit	DHCPv4
rapid-commit	14	Rapid Commit	DHCPv6
reconfigure-accept	20	Reconfigure Accept	DHCPv6
reconfigure-message	19	Reconfigure Message	DHCPv6
relay-agent-info	82	DHCP Relay Agent Information	DHCPv4
relay-agent-subscriber-id	38	Relay Agent Subscriber ID	DHCPv6
relay-message	9	Relay Message	DHCPv6
remote-id	37	Relay Agent Remote ID	DHCPv6
resource-location-servers	11	Resource Location Server	BOOTP
root-path	17	Root Path	BOOTP
router-discovery	31	Perform Router Discovery	Interface
router-solicitation-address	32	Router Solicitation Address	Interface
routers	3	Router	Basic, MS DHCP Client
server-identifier	2	DHCPv6 Server Identifier	DHCPv6
server-unicast	12	Server Unicast	DHCPv6
sip-servers	120	SIP Servers	DHCPv4
sip-servers-name	21	SIP Servers Domain Name List	DHCPv6

Table B-11 DHCP Options by Network Registrar Name (continued)

Network Registrar Name	No.	Option Name	Category
sip-servers-address	22	SIP Servers IPv6 Address List	DHCPv6
slp-directory agent	78	SLP Directory Agent	DHCPv4
slp-service-scope	79	SLP Service Scope	DHCPv4
smtp-servers	69	SMTP Server	Servers
sntp-servers	31	SNTP Configuration	DHCPv6
static-route	33	Static Route	Interface
status-code	13	Status Code	DHCPv6
streettalk-directory-assistance-servers	76	STDA Server	Servers
streettalk-servers	75	StreetTalk Server	Servers
subnet-mask	1	Subnet Mask	Basic
subnet-selection	118	Subnet Selection	DHCPv4
swap-server	16	Swap Server	BOOTP
tcp-keepalive-garbage	39	TCP Keepalive Garbage	Interface
tcp-keepalive-interval	38	TCP Keepalive Interval	Interface
tftp-server	66	TFTP Server Name	Servers
time-offset	2	Time Offset	BOOTP
time-servers	4	Time Server	BOOTP
trailer-encapsulation	34	Trailer Encapsulation	Interface
tz-database	101	TZ Database String	DHCPv4
tz-posix	100	IEEE 1003.1 String	DHCPv4
user-auth	98	User Authentication	DHCPv4
user-class	15	User Class	DHCPv6
vendor-class	16	Vendor Class	DHCPv6
vendor-encapsulated-options	43	Vendor Specific Information	DHCPv4
vendor-opts	17	Vendor Specific Information	DHCPv6
v-i-vendor-class	124	Vendor Identifying Vendor Class	DHCPv4
v-i-vendor-opts	125	Vendor Identifying Vendor Options	DHCPv4
vpn-id	185	VPN Identifier	DHCPv4
www-servers	72	WWW Server	Servers
x-display-managers	49	X Window System Display Manager	Servers

Option Validation Types

Table B-12 defines the DHCP option validation types. Note that you cannot use some of them to define custom options.

Table B-12 Validation Types

Validation	Description—Web UI Equivalent)
AT_BLOB	List of binary bytes—binary
AT_BOOL	Boolean—boolean
AT_DATE	Bytes representing a date—date
AT_DNSNAME	DNS name—DNS name
AT_INT	Unsigned 32-bit integer—unsigned 32-bit
AT_INT8	8-bit integer—unsigned 8-bit
AT_INTI	Unsigned 32-bit integer (Intel)—unsigned 32-bit (Intel)
AT_IPADDR	32-bit IP address—IP address
AT_IP6ADDR	32-bit IPv6 address—IPv6 address
AT_MACADDR	Bytes representing a MAC address—MAC address
AT_MESSAGE	Unsigned 8-bit message (not usable for custom options)
AT_NOLEN	No length (used for PAD and END only)
AT_NSTRING	Sequence of ASCII characters—string
AT_OVERLOAD	Overload bytes (not usable for custom options)
AT_RANGEBYTE	Range of bytes (not usable for custom options)
AT_RANGESHORT	Range of shorts (not usable for custom options)
AT_RDNSNAME	Relative DNS name—relative DNS name
AT_SHORT	Unsigned 16-bit integer—unsigned 16-bit
AT_SHRTI	Unsigned 16-bit integer (Intel)—unsigned 16-bit (Intel)
AT_SINT	Signed 32-bit integer—signed 32-bit
AT_SINT8	8-bit integer—signed 8-bit
AT_SINTI	Signed 32-bit integer (Intel)—signed 32-bit (Intel)
AT_SSHORT	Signed 16-bit integer—signed 16-bit
AT_SSHRTI	Signed 16-bit integer (Intel)—signed 16-bit (Intel)
AT_STIME	Signed 32-bit signed integer representing time—signed time
AT_STRING	Unrestricted sequence of ASCII characters—string
AT_TIME	Unsigned 32-bit integer representing time—unsigned time

Table B-12 Validation Types (continued)

Validation	Description—Web UI Equivalent)
AT_TYPECNT	Type requiring two child definition: size of the type field, and type of data—counted-type: For the DHCPv4 dhcp-user-class-id option (77), the repeating pattern is: `[ len (1 byte) ] [ data, of single type ]` For the DHCPv6 user-class option (15), the repeating pattern is: `[ len (2 byte) ] [ data, of single type ]`
AT_VENDOR_CLASS	Vendor-class option (enterprise ID followed by opaque data; if DHCPv4, enterprise ID is followed by EID length)—vendor-class
AT_VENDOR_OPTS	Vendor-specific options data (enterprise ID followed by TLVs of vendor-specific data; if DHCPv4, enterprise ID is followed by EID length)—vendor-opts
AT_ZEROSIZE	32 bits of zero size (no longer used for PAD and END)



APPENDIX C

DHCP Extension Dictionary

This appendix describes the DHCP extension dictionary entries and the application program interface (API) to the extension dictionary. It describes the data items available in the request and response dictionaries, and the calls to use when accessing dictionaries from Tcl extensions and shared libraries.

Extension Dictionary Entries

A dictionary is a data structure that contains key-value pairs. There are two types of dictionaries: the attribute dictionaries that the request and response dictionaries use, and the environment dictionary. This section describes the request and response dictionaries; the environment dictionary entries are described in the "Tcl Environment Dictionary Methods" section on page C-25.

Decoded DHCP Packet Data Items

The decoded DHCPv4 packet data items represent the information in the DHCP packet, and are available in both the request and response dictionaries. These dictionaries provide access to considerably more internal server data structures than just the decoded request and decoded response.

All of the options followed by an asterisk (*) are multiple, which means that there can be more than one value associated with each option. In the DHCP/BOOTP packet, all of these data items appear in the same option. However, in the extension interface, these multiple data items are accessible through indexing.

You can access options that do not have names in Table C-3 on page C-2 as option–*n*, where *n* is the option number. All fields are read/write. Table C-1 describes the field values for the DHCPv4 options; Table C-2 on page C-2 describes the field values for the DHCPv6 options.

Table C-1 DHCPv4 and BOOTP Fields

Name	Value
chaddr	blob (sequence of bytes)
ciaddr	IP address
file	string
flags	16-bit unsigned integer
giaddr	IP address
hlen	8-bit unsigned integer

Table C-1 DHCPv4 and BOOTP Fields (continued)

Name	Value
hops	8-bit unsigned integer
htype	8-bit unsigned integer
op	8-bit unsigned integer
secs	16-bit unsigned integer
siaddr	IP address
sname	string
xid	32-bit unsigned integer
yiaddr	IP address

Table C-2 DHCPv6 and BOOTP Fields

Name	Value
hop-count	8-bit unsigned integer
link-address	IPv6 address
msg-type	8-bit unsigned integer
peer-address	IPv6 address
xid	32-bit unsigned integer

Table C-3 lists the DHCP and BOOTP options for DHCPv4.

Table C-3 DHCPv4 and BOOTP Options

Name (*=multivalue)	Number	Value
all-subnets-local	27	byte-valued boolean
authentication	90	blob (sequence of bytes); 5 fields
auto-configure	116	8-bit unsigned integer
arp-cache-timeout	35	unsigned time
bcmcs-servers-a*	89	IP address
bcmcs-servers-d*	88	DNS name
boot-file	67	string
boot-size	13	16-bit unsigned integer
broadcast-address	28	IP address
cablelabs-125 (v-i-vendor-info ID: 4491)	125 suboption:	binary
oro	1	Option request, 8-bit unsigned integer
tftp-servers	2	IP addresses of TFTP servers
erouter-container	3	Relay agent CMTS capabilities
packetcable-mib-env	4	MIB environment

Table C-3 DHCPv4 and BOOTP Options (continued)

Name (*=multivalue)	Number	Value
modem-capabilities	5	Modem capabilities
cablelabs-client-configuration	122 suboption:	blob (sequence of bytes)
[ccc-]primary-dhcp-server	1	IP address
[ccc-]secondary-dhcp-server	2	IP address
[ccc-]provisioning-server	3	blob (the first byte must be the type byte, with 0 for RFC 1035 encoding, and 1 for IP address encoding, for which the address must be in network order)
[ccc-]as-backoff-retry-blob	4	12-byte blob (3 unsigned 4-byte integers, which must be in network order); configures the Kerberos AS-REQ/AS-REP timeout, back-off, and retry mechanism
[ccc-]ap-backoff-retry-blob	5	12-byte blob (3 unsigned 4-byte integers, which must be in network order); configures the Kerberos AP-REQ/AP-REP timeout, back-off, and retry mechanism
[ccc-]kerberos-realm	6	variable-length blob (an RFC 1035 style name); a Kerberos realm name is required
[ccc-]use-tgt	7	1-byte unsigned integer boolean; indicates whether to use a Ticket Granting Ticket (TGT) when obtaining a service ticket for one of the application servers
[ccc-]provisioning-timer	8	1-byte unsigned integer; defines the maximum time allowed for the provisioning process to finish
[ccc-]ticket-control-mask	9	2-byte unsigned integer, in host order
[ccc-]kdc-addresses-blob	10	variable-length (multiple of 4) IP address, in network order
cisco-autoconfigure	251	bounded byte
cisco-client-last-transaction-time	163	unsigned 32-bit integer
cisco-client-requested-host-name	162	string
cisco-leased-ip	161	IP address
cisco-vpn-id	221	blob (structured)
classless-static-route	121	blob (structured)
client-fqdn	81	blob (sequence of bytes); 4 fields: flags, rcode-1, rcode-2, and domain-name
cookie-servers*	8	IP address
default-ip-ttl	23	8-bit unsigned integer
default-tcp-ttl	37	8-bit unsigned integer
dhcp-class-identifier	60	string

Table C-3 DHCPv4 and BOOTP Options (continued)

Name (*=multivalue)	Number	Value
dhcp-client-identifier	61	blob (sequence of bytes)
dhcp-lease-time	51	unsigned time
dhcp-max-message-size	57	16-bit unsigned integer
dhcp-message	56	string
dhcp-message-type	53	8-bit unsigned integer
dhcp-option-overload	52	8-bit unsigned integer
dhcp-parameter-request-list*	55	8-bit unsigned integer
dhcp-parameter-request-list-blob*	55	blob (sequence of bytes)
dhcp-rebinding-time	59	unsigned time
dhcp-renewal-time	58	unsigned time
dhcp-requested-address	50	IP address
dhcp-server-identifier	54	IP address
dhcp-user-class-id	77	set of counted len byte arrays; 2 fields: typcnt-size and user-data
domain-name	15	string
domain-name-servers*	6	IP address
domain-search	119	blob (sequence of bytes)
extensions-path	18	string
finger-servers*	73	IP address
font-servers*	48	IP address
geo-conf	123	blob (sequence of bytes)
geoconf-civic	99	blob (sequence of bytes)
host-name	12	string
ieee802.3-encapsulation	36	byte-valued boolean
impress-servers*	10	IP address
initial-url	114	string
interface-mtu	26	16-bit unsigned integer
ip-forwarding	19	byte-valued boolean
irc-servers*	74	IP address
iSNS	83	blob (sequence of bytes); 7 fields
ldap-url	95	string
log-servers*	7	IP address
lpr-servers*	9	IP address
lq-associated-ip*	92	IP address
lq-client-last-transaction-time	91	unsigned time

Table C-3 DHCPv4 and BOOTP Options (continued)

Name (*=multivalue)	Number	Value
mask-supplier	30	byte-valued boolean
max-dgram-reassembly	22	16-bit unsigned integer
mcns-security-server	128	IP address
merit-dump	14	string
mobile-ip-home-agents*	68	IP address
name-servers*	5	IP address
name-service-search*	117	16-bit unsigned integer
nds-servers*	85	IP address
nds-tree	86	string
nds-context	87	string
netbios-dd-servers*	45	IP address
netbios-name-servers*	44	IP address
netbios-node-type	46	8-bit unsigned integer
netbios-scope	47	string
netinfo-parent-server-addr	112	IP address
netinfo-parent-server-tag	113	string
netwareip-domain	62	string
netwareip-information	63	blob (sequence of bytes)
nis+-servers*	65	IP address
nis+domain	64	string
nis-domain	40	string
nis-servers*	41	IP address
nntp-servers*	71	IP address
non-local-source-routing	20	byte-valued boolean
ntp-servers*	42	IP address
path-mtu-aging-timeout	24	unsigned time
path-mtu-plateau-tables*	25	16-bit unsigned integer
perform-mask-discovery	29	byte-valued boolean
policy-filters*	21	IP address (there can be two policy filters, each one having its own IP address)
pop3-servers*	70	IP address
pxe-client-arch	93	16-bit unsigned integer
pxe-client-machine-id	97	blob (sequence of bytes); 2 fields: type-flag and uuid
pxe-client-network-id	94	blob (sequence of bytes); 2 fields: type-flag and version
rapid-commit	80	null-length

Table C-3 DHCPv4 and BOOTP Options (continued)

Name (*=multivalue)	Number	Value
relay-agent-info suboptions:	82 suboption:	blob (sequence of bytes)
relay-agent-circuit-id-data	1	blob (sequence of bytes)
relay-agent-remote-id-data	2	blob (sequence of bytes)
relay-agent-device-class-data	4	4-byte unsigned integer
relay-agent-subnet-selection-data	5	IP address
subscriber-id	6	string identifying the network client or subscriber
radius-attributes	7	supported attributes are user, class, and framed-pool
authentication	8	binary
v-i-vendor-opts	9	vendor options
cisco-subnet-selection	150	IP address
cisco-vpn-id	151	binary
cisco-server-id-override	152	IP address
relay-agent-vpn-id-data	181	string
relay-agent-server-id-override-data	182	IP address

Note The relay-agent-circuit-id, relay-agent-remote-id, and relay-agent-device-class suboptions, which returned the two bytes (suboption code and data length) preceding the suboption data, are deprecated, but still available.

resource-location-servers*	11	IP address
root-path	17	string
router-discovery	31	byte-valued boolean
router-solicitation-address	32	IP address
routers*	3	IP address
sip-servers	120	blob (sequence of bytes); 2 fields: flag and sip-server-list
slp-directory-agent*	78	blob (sequence of bytes); 2 fields: mandatory and agent-ip-list
slp-service-scope*	79	blob (sequence of bytes); 2 fields: mandatory and slp-scope-list
smtp-servers*	69	IP address
static-routes*	33	IP address
streettalk-directory-assistance-servers*	76	IP address
streettalk-servers*	75	IP address

Table C-3 DHCPv4 and BOOTP Options (continued)

Name (*=multivalue)	Number	Value
subnet-alloc	220	blob (sequence of bytes); 5 fields: flags, subnet-request, subnet-info, subnet-name, and subnet-suggested-lease-time
subnet-mask	1	IP address
subnet-selection	118	IP address
swap-server	16	IP address
tcp-keepalive-internal	38	unsigned time
tcp-keepalive-garbage	39	byte-valued boolean
tftp-server	66	string
time-offset	2	signed time
time-servers*	4	IP address
trailer-encapsulation	34	byte-valued boolean
user-auth	98	string
vendor-encapsulated-options	43	blob (sequence of bytes)
v-i-vendor-class	124	blob (sequence of bytes)
v-i-vendor-info	125	blob (sequence of bytes)
vpn-id	185	blob (structured); 2 fields: flag and vpn-id
www-servers*	72	IP address
x-display-managers*	49	IP address

Table C-4 lists the DHCPv6 options.



Note Access to these options is available using the **putOption**, **getOption**, and **removeOption** methods only.

Table C-4 DHCPv6 Options

Name (*=multivalue)	Number	Value
auth	11	binary; 5 fields: protocol, algorithm, replay-detection-method, replay-detection, and auth-info
bcmcs-server-a*	34	IPv6 address
bcmcs-server-d*	33	DNS name
cablelabs-17 (vendor-opts ID: 4491)	17 suboption:	vendor-opts; 20 suboptions
oro	1	16-bit unsigned integer
device-type	2	string
embedded-components-list	3	string
device-serial-number	4	string

Table C-4 DHCPv6 Options (continued)

Name (*=multivalue)	Number	Value
hardware-version-number	5	string
software-version-number	6	string
boot-rom-version	7	string
vendor-oui	8	string
model-number	9	string
vendor-name	10	string
ecm-cfg-encaps	15	string
tftp-servers	32	IPv6 address
config-file-name	33	string
syslog-servers	34	IPv6 address
modem-capabilities	35	binary
device-id	36	binary
rfc868-servers	37	IPv6 address
time-offset	38	unsigned time
cmts-capabilities	1025	binary; 1 suboption: docsis-version
cm-mac-address	1026	binary
erouter-container	1027	binary
cablelabs-client-configuration	2170	IPv6 address; 10 suboptions (various data types)
client-fqdn	39	binary; 2 fields: flags and domain-name
client-identifier	1	binary
dns-servers*	23	IPv6 address
domain-list*	24	DNS name
elapsed-time	8	unsigned 16-bit
geoconf-civic	36	binary
ia-na	3	binary; 3 fields: iaid, t1, and t2
ia-pd	25	binary; 3 fields: iaid, t1, and t2
ia-prefix	26	binary; 4 fields: preferred-lifetime, valid-lifetime, prefix-length, and prefix
ia-ta	4	binary; 1 suboption: iaid
iaaddr	5	binary; 3 fields: address, preferred-lifetime, and valid-lifetime
info-refresh-time	32	unsigned time
interface-id	18	binary
nis-domain-name*	29	DNS name
nis-servers*	27	IP address
nisp-domain-name*	30	DNS name

Table C-4 DHCPv6 Options (continued)

Name (*=multivalue)	Number	Value
nisp-servers*	28	IP address
oro*	6	unsigned 16-bit
preference	7	unsigned 8-bit
rapid-commit	14	zero size
reconfigure-accept	20	zero size
reconfigure-message	19	unsigned 8-bit
relay-agent-subscriber-id	38	binary
relay-message	9	binary
remote-id	37	binary; 2 fields: enterprise-id and remote-id
server-identifier	2	binary (AT_BLOB)
server-unicast	12	IPv6 address
sip-servers-address*	22	IPv6 address
sip-servers-name*	21	DNS name
sntp-servers*	31	IP address
status-code	13	binary; 2 fields: status-code and status-message
user-class*	15	counted-type; 2 fields: typecnt-size and user-data
vendor-class	16	vendor-class
vendor-opts	17	vendor-opts (see also cablelabs-17)

Request Dictionary

Table C-5 lists the data items that you can set in the request dictionary at any time. The DHCP server reads them at various times. Unless indicated otherwise, all operations are read/write.

Table C-5 Request Dictionary Specific Data Items

Data Item	Value (Protocol: v4=DHCPv4, v6=DHCPv6)
allow-bootp	int (v4)
If set to 1, allows BOOTP for any scope for this request. Read during scope selection and while checking for lease acceptability.	
allow-dhcp	int (v4)
If set to a 1, allows DHCP for any scope for this request. Read during scope selection and while checking for lease acceptability.	
allow-dynamic-bootp	int (v4)
If set to a 1, allows dynamic BOOTP for any scope for this request. Read during scope selection and while checking for lease acceptability.	
bootp-reply-options	blob (v4)
Overrides any *v4-bootp-reply-options* in any policy. Read when gathering data for the output packet.	

Table C-5 Request Dictionary Specific Data Items (continued)

Data Item	Value (Protocol: v4=DHCPv4, v6=DHCPv6)
client-class-name	string (v4, v6)
Name of the client-class used to complete the client information (if any). Read-only.	
client-class-policy	string (v4, v6)
Name of the policy that is associated with the client-class. If set, it must be with the name of a policy that was already configured in the server.	
client-domain-name	string (v4, v6)
Domain name that the client wants to use. It might not exist, in which case the DHCP server uses the domain name specified in the scope. Read when queuing the request for DNS update just prior to the update of stable storage. For DHCPv6, overrides the client-fqdn value and used for DNS updates.	
client-host-name	string (v4, v6)
Hostname for the client in DNS; read when queuing in the request for a DNS update just before updating stable storage. Places the actual name in DNS when that operation finishes. For DHCPv6, overrides the client-fqdn value and used for DNS updates.	
client-id	blob (v4, v6)
Client identification that the server uses to track the client. Can be the *client-id* sent with a request or internally generated from the MAC address. See *client-id-created-from-mac-address*.	
client-id-created-from-mac-address	int (v4)
If set to 1, the *client-id* must be created for internal use from the client-supplied MAC address and should not be used in reporting.	
client-ipaddress	IP address (v4)
IP address from which the client sent its packet. Note that it could be zero if the client does not yet have an IP address.	
client-limitation-id	blob (v4, v6)
Limitation ID for the client.	
client-lookup-id	blob (v4, v6)
Client lookup ID calculated by the *client-lookup-id* expression of the client-class.	
client-mac-address	blob (v4)
MAC address stored in the client object associated with the request dictionary. Has the same format (and was created from) the *mac-address*.	
client-os-type	int (v4)
Change the client entry of the request packet by setting this at the **pre-client-lookup** or **post-client-lookup** extension points. Can also be read at **check-lease-acceptable**, but cannot be set there. To set the value, you must first set the *os-type* in the **post-packet-decode** request dictionary.	
client-packet	blob (v4, v6, read-only)
Valid only for getBytes. Returns the address and length of the client portion of the packet. For DHCPv4, this is the complete packet. For DHCPv6, this is the client message.	
client-policy	string (v4, v6)
Name of the policy that is associated with the client entry. If set, must be the name of a preconfigured policy in the DHCP server.	
client-port	int (v4, v6)

Table C-5 Request Dictionary Specific Data Items (continued)

Data Item	Value (Protocol: v4=DHCPv4, v6=DHCPv6)
Port from which the client sent its request.	
client-requested-host-name	string (v4)
Hostname that the client requested be used for the DNS update. The DHCP server saves this information so that a change can be detected.	
client-unicast	boolean (v6, read-only)
True if the packet is a client unicast packet.	
client-wants-nulls-in-strings	int (v4)
Determines whether the DHCP server returns strings to the client terminated with a null. If set to 1, the server terminates strings with a null. If set to 0, it does not terminate strings with a null. Set before **post-packet-decode** and read when encoding the response packet after **pre-packet-encode**.	
derived-vpn-id	int (v4, v6, read-only)
VPN identifier. See *vpn-name* for details.	
destination-address	IP address (v6, read-only)
Destination IPv6 address of the packet.	
dhcp-reply-options	blob (v4, v6)
Overrides any dhcp-reply-options specified in a policy; read when gathering data for the output packet.	
dump-packet	int (v4, v6, write-only)
When set to 1, Network Registrar dumps the current decoded DHCP/BOOTP packet to the log file. An extension can put the value 1 into this data item at multiple points in its execution. This might be useful when debugging extensions.	
import-packet	int (v4)
Determines whether the server treats the packet as if it came from an import client. If set to 1, the server treats the client as an import client and performs all DNS operations on it before sending an ACK. Read when checking the server import mode (right after **post-packet-decode**), getting ready for DNS processing, and when setting the reply address.	
limitation-count	int (v4)
Number of simultaneous users allowed with the same *limitation-id*.	
limitation-id	blob (v4)
Calculated by the *limitation-id* expression (if any) for the client-class in which this request falls.	
limitation-id-null	int (v4)
Set to 1(TRUE) if the *limitation-id* is null, 0 (FALSE) if another value.	
log-client-criteria-processing	int (v4, v6)
If set to a 1, logs the criteria processing for the client for this request. Read when trying to acquire a new lease for a client that does not have one, and when checking for lease acceptability.	
log-client-detail	int (v4, v6)
If set to a 1, logs the client-class processing for this request. Read at the end of client-class processing, after **post-client-lookup**.	
log-dns-update-detail	int (v4, v6)
If set to a 1, logs DNS update details for this request.	

Table C-5 Request Dictionary Specific Data Items (continued)

Data Item	Value (Protocol: v4=DHCPv4, v6=DHCPv6)
log-dropped-bootp-packets	int (v4)
If set to a 1, logs dropped BOOTP packets for this request.	
log-dropped-dhcp-packets	int (v4, v6)
If set to a 1, logs dropped DHCP packets for this request.	
log-dropped-waiting-packets	int (v4, v6)
If set to a 1, logs dropped waiting packets for this request.	
log-failover-detail	int (v4)
If set to a 1, logs a more detailed level of failover activity, such as all failover state changes.	
log-incoming-packet-detail	int (v4, v6)
If set to a 1, checks whether detailed incoming packet tracing occurred for this request, so that you do not need to put a separate trace on it. Read before packet decoding and the first extension point.	
log-incoming-packets	int (v4, v6)
If set to a 1, logs the incoming packets for this request. Read after **post-decode-packet**.	
log-ldap-create-detail	int (v4)
If set to a 1, logs messages whenever the DHCP server initiates a lease state entry creation to, receives a response from, or retrieves a result or error message from an LDAP server.	
log-ldap-query-detail	int (v4, v6)
If set to a 1, logs messages whenever the DHCP server initiates a query to, receives a response from, or retrieves a query result or an error message from an LDAP server.	
log-ldap-update-detail	int (v4)
If set to a 1, logs messages whenever the DHCP server initiates an update lease state to, receives a response from, or a retrieves a result or error message from an LDAP server.	
log-leasequery	int (v4, v6)
If set to a 1, logs messages when leasequery packets are processed without internal errors and result in an ACK or a NAK.	
log-missing-options	int (v4, v6)
If set to a 1, logs missing options (those a client requests but the DHCP server cannot return). Read while gathering data for the response.	
log-outgoing-packet-detail	int (v4, v6)
If set to a 1, logs a detailed dump of the outgoing packet for this request. Read after **pre-packet-encode** and just before sending the packet to the DHCP client.	
log-success-messages	int (v4, v6)
If set to a 1, logs the success messages.	
log-unknown-criteria	int (v4, v6)
If set to a 1, logs any unknown criteria specified in the client inclusion or exclusion criteria for this request. Read when acquiring a new client lease or checking lease acceptability for an existing client.	
log-v6-lease-detail	int (v6)
If set to 1, logs individual messages about DHCPv6 leasing activity.	

Table C-5 Request Dictionary Specific Data Items (continued)

Data Item	Value (Protocol: v4=DHCPv4, v6=DHCPv6)
mac-address	blob (v4)
MAC address that came in the client packet. The first byte is the hardware type, the second is the hardware length, and the remaining (up to 16) is the information from the *chaddr* read just after **post-packet-decode**. This is a useful aggregation of the *htype*, *hlen*, and *chaddr* fields of the DHCP packet. When read it is constructed from these fields; when written it is placed into these fields.	
max-client-lookups	integer (v4, v6))
Maximum number of client database lookups allowed. Usually a small integer such as 2; the preset value is 1.	
override-client-id	blob (v4, v6)
Blob used for the current client-id value. Replaces any client-id from the incoming packet (although both values are kept in the lease state database).	
packet	string (v4, v6)
Valid only for getBytes and putBytes. For getBytes, it returns the address and length of the entire packet. For putBytes, it replaces the existing packet with a new packet (which the server decodes). For DHCPv4, this is the same as *client-packet*. For DHCPv6, this is the full packet if relayed or the same as *client-packet* if not relayed.	
ping-clients	int (v4)
If set to a 1, performs a ping before offering a lease for this request. Read just before determining if a lease is acceptable for a client.	
relay-agent-circuit-id	blob (v4, v6)
Contents of the circuit-id suboption of option 82.	
relay-agent-circuit-id-data	blob (v4, v6)
Contents of just the data part of the circuit-id suboption of option 82.	
relay-agent-device-class-data	blob (v4, v6)
Contents of the device-class suboption of option 82.	
relay-agent-radius-attributes	blob (v4)
Contents of the radius suboption of option 82.	
relay-agent-radius-class	string (v4)
Encapsulated class attribute of the radius suboption of option 82.	
relay-agent-radius-pool-name	string (v4)
Encapsulated framed-pool attribute of the radius suboption of option 82.	
relay-agent-radius-user	string (v4)
Encapsulated user attribute of the radius suboption of option 82.	
relay-agent-remote-id	blob (v4, v6)
Contents of the remote-id suboption of option 82.	
relay-agent-remote-id	blob (v4, v6)
Contents of just the data part of the remote-id suboption of option 82.	

Table C-5 Request Dictionary Specific Data Items (continued)

Data Item	Value (Protocol: v4=DHCPv4, v6=DHCPv6)
relay-agent-server-id-override-data	IPv6 address (v4, v6)
Contents of the server-id suboption of option 82. If the IANA suboption 182 is in the packet, that value appears; otherwise, the Cisco suboption 152 value appears.	
relay-agent-subscriber-id	string (v4)
Contents of the subscriber-id suboption of option 82.	
relay-count	int (v6, read-only)
Number of DHCPv6 relay hops.	
reply-options	blob
Overrides any DHCPv4 reply options specified in any policy. Read when gathering data for the output packet.	
reply-to-client-address	int (v4, v6)
For v4, if set to 1, the server sends the response packet to the *client-ipaddress* and the *client-port* instead of using the RFC-mandated algorithm. For v6, if set to 1, the server sends the response packet back to the address and port of the sender (client or relay agent).	
selection-criteria	string (v4, v6)
Comma-separated string that contains the scope selection criteria.	
selection-criteria-excluded	string (v4, v6)
Comma-separated string that contains the scope exclusion criteria.	
send-ack-first	int (v4, v6)
If set to a 1, updates DNS after the ACK for DHCP requests. Read just before initiating the DNS operation.	
source-ipaddress	IPv6 address (v6, read-only)
IPv6 source address of the packet.	
trace-id	string (v4, v6, read-only)
ID used by the system to trace the packet.	
transaction-time	int (v4, v6)
Time, in seconds since 1970, that the input packet was decoded.	
update-dns	string (v4, v6)
Requests partial, full, or no dynamic DNS updates on a per-request packet basis. Input and output values are: 1=update-all, 2=update-fwd-only, 3=update-rev-only, and 0=update-none.	
update-dns-for-bootp	int (v4)
If set to a 1, updates DNS for BOOTP requests for this request. Read just before initializing the DNS operation for BOOTP.	
verbose-logging	int (v4, v6)
If set to a 1, logs verbose messages for this request. Read at various times during processing.	
vpn-description	string (v4, v6, read-only)
Description for the VPN. See *vpn-name* for details.	

Table C-5 Request Dictionary Specific Data Items (continued)

Data Item	Value (Protocol: v4=DHCPv4, v6=DHCPv6)
vpn-name	string (v4, v6, read-only)
Name of the VPN. The request dictionary does not have valid values for these items at **post-packet-decode**, but does at all other extension points, because the VPN has not yet been determined. This is so that a script can change the *derived-vpn-id* option or suboption at **post-packet-decode** and thereby affect the VPN used for a lease.	
vpn-vpn-id	blob, typically 7 bytes (v4, v6, read-only)
Virtual private network identifier. See *vpn-name* for details.	
vpn-vrf-name	string (v4, v6, read-only)
Virtual routing and forwarding table identifier for the VPN. See *vpn-name* for details.	

Response Dictionary

Table C-6 lists the data items you can set in the response dictionary at any time. The DHCP server reads them at various times. Unless indicated otherwise, the operation is read/write.

Table C-6 Response Dictionary Specific Data Items

Data Item	Value (Protocol: v4=DHCPv4, v6=DHCPv6)
client-active-lease-count	int (v6, read-only)
Number of active leases on the DHCPv6 client.	
client-creation-time	int (v6, read-only)
Creation time of the IPv6 client.	
client-domain-name	string (v4, read-only)
From the client information in the lease, the domain name that the client wants to use. It might not exist, in which case the DHCP server uses the domain name specified in the scope. Read when queuing the request for DNS update just prior to the update of stable storage.	
client-expiration-time	int (v4, read-only)
Absolute time value of the lease expiration time last given to the client.	
client-host-name	string (v4, read-only)
From the client information in the lease, the hostname that the DHCP server puts into DNS. Read when queueing the request for a DNS update just before updating stable storage.	
client-id	blob (v4, v6, read-only)
From the client information in the lease, the client identification that the server used to keep track of the client. This might be the *client-id* sent with a request or a *client-id* internally generated from the MAC address.	
client-id-created-from-mac-address	int (v4, read-only)
From the client information in the lease. If set to 1, the *client-id* must be created from the MAC address and should not be used in reporting.	
client-last-transaction-time	int (v4, v6, read-only)
Time, in seconds, since 1970, that the DHCP server last heard from this client.	

Table C-6 Response Dictionary Specific Data Items (continued)

Data Item	Value (Protocol: v4=DHCPv4, v6=DHCPv6)
client-limitation-id	blob (v4, read-only)
Limitation identifier of the client associated with the current lease.	
client-mac-address	blob (v4, read-only)
From the client information in the lease, the MAC address stored in the client object associated with the request dictionary. Has the same format as (and was created from) the *mac-address*.	
client-os-type	int (v4)
Change the client entry of the request packet by setting this at the **pre-client-lookup** or **post-client-lookup** extension points. Can also be read at **check-lease-acceptable**, but cannot be set there. To set the value, you must first set the *os-type* in the **post-packet-decode** request dictionary.	
client-override-client-id	blob (v4, v6, read-only)
Blob used for the current client-id value. Replaces any client-id from the incoming packet (although both values are kept in the lease state database).	
client-packet	blob (v4, v6, read-only)
Valid only for getBytes. Returns the address and length of the client portion of the packet. For DHCPv4, this is the complete packet. For DHCPv6, this is the client message.	
client-reconfigure-key	string (v6)
Returns the *client-reconfigure-key* attribute value of the DHCPv6 lease.	
client-reconfigure-key-generation-time	string (v6)
Returns the *client-reconfigure-key-generation-time* attribute value of the DHCPv6 lease.	
client-relay-address	IPv6 address (v6, read-only)
Source IPv6 address for the (last) relay.	
client-relay-message	string (v6, read-only)
Last relayed DHCPv6 message, excluding the client message.	
client-requested-host-name	string (v4)
From the client information in the lease, the hostname that the client requested for the DNS update.	
client-user-defined-data	string (v4, v6)
Returns the value previously or currently associated with the client. It returns the previously associated value if requested in a **check-lease-acceptable** or **lease-state-change** extension point. It returns the current value if requested in a **pre-packet-encode** or **post-send-packet** extension point.	
client-vendor-class	string (v4, v6)
Returns the *client-vendor-class* attribute value of the DHCPv4 or DHCPv6 lease.	
client-vendor-info	string (v4, v6)
Returns the *client-vendor-info* attribute value of the DHCPv4 or DHCPv6 lease.	
client-write-sequence	int (v6, read-only)
Write sequence of the client IPv6 request.	
client-write-time	int (v6, read-only)
Time of the client IPv6 write request.	

Table C-6 Response Dictionary Specific Data Items (continued)

Data Item	Value (Protocol: v4=DHCPv4, v6=DHCPv6)
derived-vpn-id	int (v4, v6, read-only)
VPN identifier.	
domain-name-changed	int (v4)
If set to 1, the domain name in the current packet differs from the domain name used in the DNS update. Read after **check-lease-acceptable** and before **pre-packet-encode**.	
dump-packet	int (v4, v6, write-only)
When set to 1, Network Registrar dumps the current decoded DHCP/BOOTP packet to the log file. An extension can put the value 1 into this data item at multiple points in its execution. This might be useful when debugging extensions.	
host-name-changed	int (v4)
If set to 1, the hostname in the current packet differs from that used in the DNS update. Read after **check-lease-acceptable** and before **pre-packet-encode**.	
host-name-in-dns	int (v4, v6)
If set to 1, the hostname is in DNS. Read after **check-lease-acceptable** and before **pre-packet-encode**. Written after the hostname goes into DNS.	
lease-binding-iaid	int (v6, read-only)
IPv6 lease binding IAID.	
lease-binding-rebinding-time	int (v6, read-only)
IPv6 lease binding rebinding time.	
lease-binding-renewal-time	int (v6, read-only)
IPv6 lease binding renewal time.	
lease-binding-type	string (v6, read-only)
IPv6 lease binding type: "IA_NA", "IA_TA", or "IA_PD".	
lease-creation-time	string (v6, read-only)
IPv6 lease creation time.	
lease-deactivated	int (v4, v6, read-only)
If set to 1, reports that the lease is deactivated.	
lease-giaddr	IP address (v4, read-only)
Lease *giaddr*.	
lease-ipaddress	IPv4 or IPv6 address (v4, v6, read-only)
Address of the lease that the DHCP server uses in processing.	
lease-preferred-lifetime	int (v6, read-only)
Preferred lifetime of the IPv6 lease.	
lease-prefix-name	string (v6, read-only)
Prefix name of the IPv6 lease.	
lease-relay-agent-info	blob (v4, v6)
Entire contents of option 82.	

Table C-6 Response Dictionary Specific Data Items (continued)

Data Item	Value (Protocol: v4=DHCPv4, v6=DHCPv6)
lease-relay-agent-circuit-id	blob (v4)
Accesses and manipulates the relay agent circuit ID as stored with the lease of a response. Requires the suboption number 1 as the first byte. Deprecated in favor of the *lease-relay-agent-circuit-id-data* item.	
lease-relay-agent-circuit-id-data	blob (v4, use instead of deprecated *lease-relay-agent-circuit-id*)
Accesses and manipulates the *relay-agent-circuit-id-data* as stored with the lease of a response. Relevant only if the *save-relay-agent-data* DHCP server attribute is enabled.	
lease-relay-agent-device-class-data	blob (v4, v6)
Contents of the device-class suboption of option 82.	
lease-relay-agent-radius-attributes	blob (v4)
Contents of the radius suboption of option 82.	
lease-relay-agent-radius-class	string (v4)
Encapsulated class attribute of the radius suboption of option 82.	
lease-relay-agent-radius-pool-name	string (v4)
Encapsulated framed-pool attribute of the radius suboption of option 82.	
lease-relay-agent-radius-user	string (v4)
Encapsulated user attribute of the radius suboption of option 82.	
lease-relay-agent-remote-id	blob (v4)
Accesses and manipulates the *relay-agent-remote-id* data as stored with the lease of a response. Requires suboption number 2 as the first byte. Deprecated in favor of the *lease-relay-agent-remote-id-data* item.	
lease-relay-agent-remote-id-data	blob (v4, use instead of *lease-relay-agent-remote-id* item)
Accesses and manipulates the *relay-agent-remote-id-data* as stored with the lease of a response. Relevant only if the *save-relay-agent-data* DHCP server attribute is enabled.	
lease-relay-agent-server-id-override-data	IP address (v4)
Accesses and manipulates the *relay-agent-server-id-override-data* as stored with the lease of a response. Relevant only if the *save-relay-agent-data* DHCP server attribute is enabled.	
lease-relay-agent-subnet-selection-data	IP address (v4)
Accesses and manipulates the *relay-agent-subnet-selection-data* as stored with the lease of a response. Relevant only if the *save-relay-agent-data* DHCP server attribute is enabled.	
lease-relay-agent-subscriber-id	string (v4)
Contents of the subscriber-id suboption of option 82.	
lease-relay-agent-vpn-id-data	blob (v4)
Accesses and manipulates the *relay-agent-vpn-id* data as stored with the lease of a response. Relevant only if the *save-relay-agent-data* DHCP server attribute is enabled.	
lease-reserved	int (v4, v6, read-only)
If set to 1, reports if the lease is reserved.	

Table C-6 Response Dictionary Specific Data Items (continued)

Data Item	Value (Protocol: v4=DHCPv4, v6=DHCPv6)
lease-start-time-of-state	int (v4, v6, read-only)
Time, in seconds since 1970, that this lease was first placed into its current state.	
lease-state	string (v4, v6, read-only)
State of the lease, which can be *available*, *offered*, *expired*, *leased*, or *unavailable*.	
lease-state-expiration-time	int (v6, read-only)
Expiration time of the IPv6 lease state.	
lease-status	string (v4, v6, read-only)
Returns "nonexistent," "owned-by-client," or "exists." Used to determine if a lease exists and if the current client owns it. If "exists" is returned, the lease exists but the current owner does not own it (limited information on the lease is available).	
lease-valid-lifetime	int (v6, read-only)
Valid lifetime of the IPv6 lease.	
lease-vpn-description	string (v4, v6, read-only)
Description for the VPN stored with the lease of a response.	
lease-vpn-id	int (v4, v6, read-only)
Identifier for the VPN stored with the lease of a response.	
lease-vpn-name	string (v4, v6, read-only)
Name of the VPN stored with the lease of a response.	
lease-vpn-vpn-id	blob, typically 7 bytes (v4, v6, read-only)
Virtual private network (VPN) identifier stored with the lease of a response.	
lease-vpn-vrf-name	string (v4, v6, read-only)
Virtual routing and forwarding table identifier for the VPN stored with the lease of a response.	
mac-address	blob (v4)
MAC address that came in the client packet. The first byte is the hardware type, the second is the hardware length, and the remaining (up to 16) is the information from the *chaddr* read just after **post-packet-decode**. This is a useful aggregation of the *htype*, *hlen*, and *chaddr* fields of the DHCP packet. When read it is constructed from these fields; when written it is placed into these fields.	
packet	string (v4, v6, read-only except at **post-packet-encode**)
Valid only for getBytes and putBytes. For getBytes, it returns the address and length of the entire packet. For putBytes, it replaces the existing packet with a new packet (which the server decodes). For DHCPv4, this is the same as *client-packet*. For DHCPv6, this is the full packet if relayed or the same as *client-packet* if not relayed.	
ping-clients	int (v4)
If set to 1, performs a ping before offering a lease for this request. Read just before determining a client lease acceptability.	
prefix-allocate-via-best-fit	int (v6, read-only)
Prefix allocated via the best fit.	
prefix-allocate-via-client-request	int (v6, read-only)
Prefix allocated via client request.	

Table C-6 Response Dictionary Specific Data Items (continued)

Data Item	Value (Protocol: v4=DHCPv4, v6=DHCPv6)
prefix-allocate-via-extension	int (v6, read-only)
Prefix allocated via an extension.	
prefix-allocate-via-reservation	int (v6, read-only)
Prefix allocated via a reservation.	
prefix-allocate-via-interface-identifier	int (v6, read-only)
Prefix allocated via an interface identifier.	
prefix-allocate-random	int (v6, read-only)
Prefix randomly allocated.	
prefix-address	IPv6 address (v6, read-only)
Prefix address (17 bytes).	
prefix-deactivated	int (v6, read-only)
Indicates if the prefix is deactivated.	
prefix-dhcp-type	string (v6, read-only)
Prefix DHCP type.	
prefix-expiration-time	string (v6, read-only)
Expiration time of the prefix.	
prefix-link-name	string (v6, read-only)
Link of the prefix.	
prefix-name	string (v6, read-only)
Name of the prefix.	
prefix-range	IPv6 address (v6, read-only)
IPv6 address range of the prefix.	
prefix-selection-tags	string (v6, read-only)
Selection tags of the prefix.	
relay-count	int (v6, read-only)
Number of DHCPv6 relay hops.	
reply-ipaddress	IPv4 or IPv6 address (v4, v6)
IP address to use when replying to the DHCP client. Read just after **pre-packet-encode**. If you change its value in a **pre-packet-encode**, the IP address you place in it should be for a system that can respond to ARP queries (unless it is a broadcast address). Even if unicast is enabled and the broadcast flag is not set in the DHCP request, the local ARP cache is not set with a mapping from a new *reply-ipaddress* in the **pre-packet-encode** to the MAC address in the DHCP request.	
reply-port	int (v4, v6)
Port to use when replying to the DHCP client. Read just after **pre-packet encode**.	
response-source	string (v4, v6, read-only)
Source of the lease state change; used by any extension where a response is valid. Output values are: unknown, client, failover, timeout, operator, and one-lease-per-client. (See Table 29-12 on page 29-38.)	

Table C-6 Response Dictionary Specific Data Items (continued)

Data Item	Value (Protocol: v4=DHCPv4, v6=DHCPv6)
reverse-name-in-dns	int (v4, v6)
If equal to 1, the reverse name is in DNS. Read before initializing a DNS operation.	
scope-allow-bootp	int (v4, read-only)
If set to 1, the scope allows BOOTP. Written after a DNS operation finishes.	
scope-allow-dhcp	int (v4, read-only)
If set to 1, the scope allows DHCP.	
scope-allow-dynamic-bootp	int (v4, read-only)
If set to 1, the scope allows dynamic BOOTP.	
scope-available-leases	int (v4, read-only)
Number of available leases on the current scope.	
scope-deactivated	int (v4, read-only)
If set to 1, the scope is deactivated.	
scope-dns-forward-server-address	IP address (v4, read-only)
DNS server to use for the DNS forward address.	
scope-dns-forward-zone-name	string (v4, v6, read-only)
Forward zone name configured in the scope.	
scope-dns-number-of-host-bytes	int (v4, read-only)
Number of host bytes used by the DHCP server code that handles DNS updates.	
scope-dns-reverse-server-address	IP address (v4, read-only)
DNS server to use for the DNS reverse address.	
scope-dns-reverse-zone-name	string (v4, read-only)
Reverse zone name configured in the scope.	
scope-network-number	IP address (v4, read-only)
Network number of the scope that contains the lease the DHCP server is processing.	
scope-ping-clients	int (v4, read-only)
If set to 1, the scope associated with the current lease was configured to support a ping operation prior to offering a lease.	
scope-primary-network-number	IP address (v4, read-only)
Network number of this primary scope.	
scope-primary-subnet-mask	IP address (v4, read-only)
Subnet mask of this primary scope.	
scope-renew-only	int (v4, read-only)
If set to 1, the scope is renew-only.	
scope-renew-only-expire-time	int (v4, read-only)
Absolute time, in seconds since January 1, 1970, at which a renew-only scope should cease to be renew-only.	

Table C-6 Response Dictionary Specific Data Items (continued)

Data Item	Value (Protocol: v4=DHCPv4, v6=DHCPv6)
scope-selection-tags	string (v4, read-only)
Comma-separated string that contains the scope selection criteria. Use this data item for decisions based on scopes.	
scope-send-ack-first	int (v4, read-only)
If set to 1, the scope sends an ACK before performing the rest of the processing.	
scope-subnet-mask	IP address (v4, read-only)
Subnet mask of the scope that contains the lease the DHCP server is processing.	
scope-update-dns	string (v4, read-only)
DNS updates for forward or reverse zones. Output values are: 1=update-all, 2=update-fwd-only, 3=update-rev-only, and 0=update-none. Introduced in Network Registrar 6.1.1.	
scope-update-dns-enabled	boolean (v4, read-only)
If set to 1, the scope has update DNS enabled for forward and reverse zones. Deprecated in favor of *scope-update-dns.*	
scope-update-dns-for-bootp	int (v4, read-only)
If set to 1, the scope has update DNS enabled for BOOTP.	
trace-id	string (v4, v6, read-only)
ID used by the system to trace the packet.	
transaction-time	int (v4, v6, read-only)
Time, in seconds since 1970, that the request was decoded.	
vpn-description	string (v4, v6, read-only)
Description for the VPN.	
vpn-name	string (v4, v6, read-only)
Name of the VPN.	
vpn-vpn-id	blob, typically 7 bytes (v4, v6, read-only)
Virtual private network (VPN) identifier.	
vpn-vrf-name	string (v4, v6, read-only)
Virtual routing and forwarding table (VRF) identifier for the VPN.	

Extension Dictionary API

This section contains the dictionary method calls to use when accessing dictionaries from Tcl extensions and shared libraries.

Tcl Attribute Dictionary API

In an attribute dictionary, the keys are constrained to be the names of attributes as defined in the Network Registrar DHCP server configuration. The values are the string representation of the legal values for that particular attribute. For example, IP addresses are specified by the dotted-decimal string representation of the address, and enumerated values are specified by the name of the enumeration. This means that numbers are specified by the string representation of the number.

Attribute dictionaries are unusual in that they can contain more than one instance of a key. These instances are ordered, with the first instance at index zero. Some of the attribute dictionary methods allow an index to indicate a particular instance or position in the list of instances to be referenced.

Tcl Request and Response Dictionary Methods

Attribute dictionaries use commands with which you can change and access the values in the dictionaries. Table C-7 lists the commands to use with the request and response dictionaries. In this case, you can define the *dict* variable as **request** or **response**.

See the *install-path*/examples/dhcp/tcl/tclextension.tcl file for examples.

Table C-7 Tcl Request and Response Dictionary Methods

Method	Syntax
get	**$***dict* **get** *attribute* [*index* [*bMore*]]
Returns the value of the attribute from the dictionary, represented as a string. If the dictionary does not contain the attribute, the empty string is returned instead. If you include the index value, this returns the *index*th instance of the attribute. Some attributes can appear more than once in the request or response packet. The index selects which instance to return. If you include the *bMore*, the **get** method sets *bMore* to TRUE if there are more attributes after the one returned, otherwise to FALSE. Use this to determine whether to make another call to **get** to retrieve other instances of the attribute.	
getOption	**$***dict* **getOption** *arg-type* [*arg-data*]
Gets the data for an option as a string. See Table C-8 on page C-25 for the *arg-type* values. If the next argument is a numeric value, it is assumed to be a number, otherwise a name. Note that this function always returns a pointer to a string, which can be zero length if the option does not exist or has length zero. For sample usage, see the "Handling Vendor Class Option Data" section on page C-37.	
isValid **isV4** **isV6**	**$***dict* **isValid** **$***dict* **isV4** **$***dict* **isV6**
The **isValid** method returns TRUE if there is a request or response (depending on the dictionary passed in); FALSE otherwise. Extensions such as **lease-state-change** can use this method to determine whether a dictionary is available. The **isV4** method returns TRUE if this extension is being called for a DHCPv4 packet; FALSE otherwise. Calling this method from an **init-entry** routine returns FALSE. The **isV6** method returns TRUE if this extension is being called for a DHCPv6 packet; FALSE otherwise. Calling this method from an **init-entry** routine returns FALSE.	

Table C-7 Tcl Request and Response Dictionary Methods (continued)

Method	Syntax
log	$*dict* **log** *level message …*
Puts a message into the DHCP server logging system. The level should be LOG_ERROR, LOG_WARNING, or LOG_INFO. The remaining arguments are concatenated and sent to the logging system at the specified level. **Note** Use the LOG_ERROR and LOG_WARNING levels sparingly, because the server flushes its log file with messages logged at these levels. Using these levels for messages that are likely to occur frequently (such as client requests) can have severe impact on disk I/O performance.	
moveToOption	$*dict* **moveToOption** *arg-type* [*arg-data*] ...
Sets the context for subsequent **get**, **put**, and **remove** option operations. See Table C-8 on page C-25 for the *arg-type* values. Note that the context can become invalid if the option is removed (such as by **removeOption**).	
put	$*dict* **put** *attribute value* [*index*]
Associates a value with the attribute in the dictionary. If you omit the index or set it to the special value REPLACE, this replaces any existing instances of the attribute with the single value. If you include the index value as the special value APPEND, this appends a new instance of the attribute to the end of the list of instances of the attribute. If you include the index value as a number, this inserts a new instance of the attribute at the position indicated. If you set the index value to the special value AUGMENT, this puts the attribute only if there is not one already.	
putOption	$*dict* **putOption** *data arg-type* [*arg-data*] ...
Adds an option and its data or modifies the data for an option. See Table C-8 on page C-25 for the *arg-type* values. For sample usage, see the "Handling Vendor Class Option Data" section on page C-37.	
remove	$*dict* **remove** *attribute* [*index*]
Removes the attribute from the dictionary. If you omit the index or set it to the special value REMOVE_ALL, this removes any existing instances of the attribute. If you include the index as a number, this removes the instance of the attribute at the position indicated. This method always returns 1, even if the dictionary does not contain that attribute at that index.	
removeOption	$*dict* **removeOption** *arg-type* [*arg-data*] ...
Removes an option. See Table C-8 on page C-25 for the *arg-type* values. For sample usage, see the "Handling Vendor Class Option Data" section on page C-37.	
setObject	$*dict* **setObject** *obj-type* [*data*]
(DHCPv6 only.) Sets the object for **get**, **put**, and **remove** methods, and alters the message on which the new option methods operate. See Table C-8 on page C-25 for the *obj-type* values. DHCPv6 extensions primarily use this method to access the leases and prefixes available for the client and link, or to get message header fields or options from relay packets. Unlike in DHCPv4, where one lease and scope are associated with a response, a DHCPv6 response can involve several leases and prefixes. Returns TRUE if the object exists; FALSE otherwise. For sample usage, see the "Handling Object Data" section on page C-37. **Note** For leases not associated with the current client, only minimal information is available.	
trace	$*dict* **trace** *level message …*
Returns a message in the DHCP server packet tracing system. At level 0, no tracing occurs. At level 1, it traces only that the server received the packet and sent a reply. At level 4, it traces everything. The remaining arguments are concatenated and sent to the tracing system at the specified level. The default tracing is set using the DHCP server *extension-trace-level* attribute.	

Table C-8 Tcl arg-type and obj-type Values

arg-type or obj-type	Description
enterprise-id *number/name*	Enterprise-id number or name for the option definition set for the option or suboption.
home	Requests that the context is reset to the "top" of the current client or relay message.
index *number/keyword*	Number or keyword (replace, append, augment, raw, or remove_all) for the array index on which to operate.
index-count	Returns the number of array index entries in the option.
instance *number*	Instance number of the option (primarily used for DHCPv6).
instance-count	Returns the number of times the option appears (if 0, the option is not present).
more *tcl-variable-name*	Name of a Tcl variable that is set to TRUE or FALSE, depending on whether more array index entries exist in the option data.
move-to	Requests that the context be set to the option.
option *number/name*	Option number or name to operate on.
parent	Requests that the context is moved up one option.
suboption *number/name*	Suboption number or name to operate on.
vendor *name*	Vendor name for the option definition set for the option or suboption.
lease initial | *index* | *address* | *prefix*	Used with **setObject**, sets the context for the lease, binding, and prefix data items in the response dictionary to the indicated lease. The **initial** keyword requests that the original context for when the extension was called is restored. The *index* requests that the numbered lease (starting at 0) is set and can be used to iterate through all of the leases for the client. The *address* or *prefix* requests that the lease with that address or prefix is set (if it exists).
message initial | *number*	Used with **setObject**, sets the context for the message data items and options in the request or response dictionary to the indicated message. The **initial** keyword sets the context to the client message. The *number* sets the context to the relay message, with 0 specifying the relay closest to the client.
prefix initial | *index* | *address* | *prefix* | *name*	Used with **setObject**, sets the context for the prefix data items in the response dictionary to the indicated prefix. The **initial** keyword requests that the original context for when the extension was called is restored. The *index* requests the numbered prefix (starting at 0) is set and can be used to iterate through all of the prefixes for the client on the link. The *address* or *prefix* requests that the prefix for the address or prefix is set (if found). The *name* requests that the named prefix is found. Note that only prefixes on the current link can be used.

Tcl Environment Dictionary Methods

Table C-9 describes the commands to use with the environment dictionary. In this case, you can define the *dict* variable as **environ**, as in the following procedure example:

```
proc tclhelloworld2 { request response environ } {
```

```
    $environ put trace-level 4
    $environ log LOG_INFO "Environment hello world"
}
```

Table C-9 Tcl Environment Dictionary Methods

Method	Syntax
clear	$*dict* **clear**
Removes all entries from the dictionary.	
containsKey	$*dict* **containsKey** *key*
Returns 1 if the dictionary contains the key, otherwise 0.	
firstKey	$*dict* **firstKey**
Returns the name of the first key in the dictionary. Note that the keys are not stored sorted by name. If a key does not exist, returns the empty string.	
get	$*dict* **get** *key*
Returns the value of the key from the dictionary. If a key does not exist, returns the empty string.	
isEmpty	$*dict* **isEmpty**
Returns 1 if the dictionary has no entries, otherwise 0.	
log	$*dict* **log** *level message …*
Returns a message in the DHCP server logging system. The *level* should be one of LOG_ERROR, LOG_WARNING, or LOG_INFO. The remaining arguments are concatenated and sent to the logging system at the specified level. **Note** Use the LOG_ERROR and LOG_WARNING levels sparingly, because the server flushes its log file with messages logged at these levels. Using these levels for messages that are likely to occur frequently (such as client requests) can have severe impact on disk I/O performance.	
nextKey	$*dict* **nextKey**
Returns the name of the next key in the dictionary that follows the key returned in the last call to **firstKey** or **nextKey**. If a key does not exist, returns the empty string.	
put	$*dict* **put** *key value*
Associates a value with the key, replacing an existing instance of the key with the new value.	
remove	$*dict* **remove** *key*
Removes the key from the dictionary. Always returns 1, even if the dictionary did not contain the key.	
size	$*dict* **size**
Returns the number of entries in the dictionary.	
trace	$*dict* **trace** *level message …*
Returns a message in the DHCP server packet tracing system. At level 0, no tracing occurs. At level 1, it traces only that the server received the packet and sent a reply. At level 4, it traces everything. The remaining arguments are concatenated and sent to the tracing system at the specified level. The default tracing is set using the DHCP server *extension-trace-level* attribute.	

DEX Attribute Dictionary API

When writing DEX extensions for C/C++, you can specify keys as the attribute name string representation or by type (a byte sequence defining the attribute). This mean that some of these access methods have four different variations that are the combinations of string or type for the key or value.

A basic DEX extension example might be:

```
int DEXAPI dexhelloworld( int iExtensionPoint,
        dex_AttributeDictionary_t *pRequest,
        dex_AttributeDictionary_t *pResponse,
        dex_EnvironmentDictionary_t *pEnviron )
{
pEnviron->log( pEnviron, DEX_LOG_INFO, "hello world" );
return DEX_OK;
}
```

See the *install-path*/examples/dhcp/dex/dexextension.c file or other files in that directory for examples.

DEX Request and Response Dictionary Methods

DEX attribute dictionaries use active commands, called methods, with which you can change and access values. Table C-10 on page C-27 lists the methods to use with the request and response dictionaries. In this case, you can define the *pDict* variable as **pRequest** or **pResponse**, as in:

```
pRequest->get( pRequest, "host-name", 0, 0 );
```

The *pszAttribute* is the **const char *** pointer to the attribute name that the application wants to access. The *pszValue* is the pointer to the **const char *** string that represents the data (returned for a **get** method, and stored in a **put** method). See Table C-11 on page C-31, Table C-12 on page C-32, and Table C-13 on page C-33 for the valid *iObjectType*, *iObjArgType*, and *iArgType* values, respectively.

Tip See also the "Differences in get, put, Option, Bytes, and OptionBytes Methods" section on page C-30 and the "Differences in get, put, remove, and ByType Methods" section on page C-31.

Table C-10 DEX Request and Response Dictionary Methods

Method	Syntax
allocateMemory	**void** **pDict*->**allocateMemory(dex_AttributeDictionary_t** *pDict*, **unsigned int** *iSize*)
Allocates memory in extensions that persists only for the lifetime of this request.	
get	**const char** **pDict*->**get(dex_AttributeDictionary_t** *pDict*, **const char** **pszAttribute*, **int** *iIndex*, **abool_t** **pbMore*)
Returns the value of the *iIndex*ed instance of the attribute from the dictionary, represented as a string. If the dictionary does not contain the attribute (or that many instances of it), the empty string is returned instead. If *pbMore* is nonzero, the **get** method sets *pbMore* to TRUE if there are more instances of the attribute after the one returned, otherwise to FALSE. Use this to determine whether to make another call to **get** to retrieve other instances of the attribute.	

Table C-10 DEX Request and Response Dictionary Methods (continued)

Method	Syntax
getBytes	**const abytes_t ****pDict***->getBytes(dex_AttributeDictionary_t ****pDict***, const char ****pszAttribute***, int** *iIndex***, abool_t ****pbMore* **)**

Returns the value of the *iIndex*ed instance of the attribute from the dictionary as a sequence of bytes. If the dictionary does not contain the attribute (or that many instances of it), returns 0 instead. If *pbMore* is nonzero, the **getBytes** method sets it to TRUE if there are more instances of the attribute after the one returned, otherwise to FALSE. Use this to determine whether to make another call to **getBytes** to retrieve other instances of the attribute.

getBytesByType	**const abytes_t ****pDict***-> getBytesByType(dex_AttributeDictionary_t ****pDict***, const abytes_t ****pszAttribute***, int** *iIndex***, abool_t ****pbMore* **)**

Returns the value of the *iIndex*ed instance of the attribute from the dictionary as a sequence of bytes. If the dictionary does not contain the attribute (or that many instances of it), 0 is returned instead. If *pbMore* is nonzero, sets the variable pointed to TRUE if there are more instances of the attribute after the one returned, otherwise to FALSE. Use this to determine whether to make another call to **get** to retrieve other instances of the attribute.

getByType	**const char ****pDict***->getByType(dex_AttributeDictionary_t ****pDict***, const abytes_t ****pszAttribute***, int** *iIndex***, abool_t ****pbMore* **)**

Returns the value of the *iIndex*ed instance of the attribute from the dictionary, represented as a string. If the dictionary does not contain the attribute (or that many instances of it), returns the empty string instead. If *pbMore* is nonzero, the **getByType** method sets *pbMore* to TRUE if there are more instances of the attribute after the one returned, otherwise to FALSE. Use this to determine whether to make another call to **getByType** to retrieve other instances.

getOption	**const char * getOption(dex_AttributeDictionary_t ****pDict***, int** *iArgType*, ... **)**

Gets the data for an option as a string. Note that this function always returns a pointer to a string, which can be zero length if the option does not exist or has length zero. To find out if the option exists, use **getOptionBytes** or specify DEX_INSTANCE_COUNT.

getOptionBytes	**const abytes_t * getOptionBytes(dex_AttributeDictionary_t ****pDict***, int** *iArgType*, ... **)**

Gets the data for an option as a sequence of bytes. Note that this function returns a null pointer if the option does not exist, and an **abytes_t** with a zero-length buffer if the option exists but is zero bytes long.

getType	**const abytes_t *** *pDict***->getType(dex_AttributeDictionary_t*** *pDict***, const char*** *pszAttribute* **)**

Returns a pointer to the byte sequence defining the attribute, if the attribute name matches a configured attribute, otherwise 0.

isValid	**abool_t isValid(dex_AttributeDictionary_t ****pDict* **)**
isV4	**abool_t isV4(dex_AttributeDictionary_t ****pDict* **)**
isV6	**abool_t isV6(dex_AttributeDictionary_t ****pDict* **)**

The **isValid** method returns TRUE if there is a request or response (depending on the dictionary passed in); FALSE otherwise. Extensions such as **lease-state-change** can use this method to determine whether a dictionary is available.

The **isV4** method returns TRUE if this extension is being called for a DHCPv4 packet; FALSE otherwise. Calling this method from an **init-entry** routine returns FALSE.

The **isV6** method returns TRUE if this extension is being called for a DHCPv6 packet; FALSE otherwise. Calling this method from an **init-entry** routine returns FALSE.

Table C-10 DEX Request and Response Dictionary Methods (continued)

Method	Syntax
log	**abool_t** *pDict*->**log(dex_AttributeDictionary_t** ***pDict*, int** *eLevel*, **const char** ***pszFormat*, ...)**

Returns a message in the DHCP server logging system. The *eLevel* should be one of DEX_LOG_ERROR, DEX_LOG_WARNING, or DEX_LOG_INFO. The *pszFormat* is treated as a printf style format string, and it, along with the remaining arguments, are formatted and sent to the logging system at the specified level.

Note Use the DEX_LOG_ERROR and DEX_LOG_WARNING levels sparingly, because the server flushes its log file with messages logged at these levels. Using these levels for messages that are likely to occur frequently (such as client requests) can have severe impact on disk I/O performance.

moveToOption	**abool_t moveToOption(dex_AttributeDictionary_t** ***pDict*, int** *iArgType*, ...)

Sets the context for subsequent **get**, **put**, and **remove** option operations. Note that the context can become invalid if the option is removed (such as with **removeOption**).

put	**abool_t** *pDict*->**put(dex_AttributeDictionary_t** ***pDict*, const char** ***pszAttribute*, const char** ***pszValue*, int** *iIndex*)

Converts *pszValue* to a sequence of bytes, according to the definition of *pszAttribute* in the server configuration. Associates that sequence of bytes with the attribute in the dictionary. If *iIndex* is the special value DEX_REPLACE, replaces any existing instances of the attribute with a single value. If the special value DEX_APPEND, it appends a new instance of the attribute to its list. If the special value DEX_AUGMENT, puts the attribute only if there is not one already. Otherwise, inserts a new instance at the position indicated. Returns TRUE unless the attribute name does not match any configured attributes or the value could not be converted to a legal value.

putBytes	**abool_t** *pDict*->**putBytes(dex_AttributeDictionary_t** ***pDict*, const char** ***pszAttribute*, const abytes_t** ***pszValue*, int** *iIndex*)

Associates *pszValue* with the *pszAttribute* in the dictionary. If *iIndex* is the special value DEX_REPLACE, replaces any existing instances of the attribute with a single new value. If the special value DEX_APPEND, appends a new instance of the attribute to its list. If the special value DEX_AUGMENT, puts the attribute only if there is not one already. Otherwise, inserts a new instance at the position indicated. Returns TRUE unless the attribute name does not match a configured one.

putBytesByType	**abool_t** *pDict*->**putBytesByType(dex_AttributeDictionary_t** ***pDict*, const abytes_t** ***pszAttribute*, const abytes_t** ***pszValue*, int** *iIndex*)

Associates *pszValue* with the *pszAttribute* in the dictionary. If *iIndex* is the special value DEX_REPLACE, replaces any existing instances of the attribute with the new value. If the special value DEX_APPEND, appends a new instance of the attribute to its list. If the special value DEX_AUGMENT, puts the attribute only if there is not one already. Otherwise, inserts a new instance of the attribute at the position indicated.

putByType	**abool_t** *pDict*->**putByType(dex_AttributeDictionary_t** ***pDict*, const abytes_t** ***pszAttribute*, const char** ***pszValue*, int** *iIndex*)

Converts *pszValue* to a sequence of bytes, according to the definition of *pszAttribute* in the server configuration. Associates that sequence of bytes with the attribute in the dictionary. If *iIndex* is the special value DEX_REPLACE, replaces any existing instances of the attribute with a single new value. If the special value DEX_APPEND, appends a new instance of the attribute to its list. If the special value DEX_AUGMENT, puts the attribute only if there is not one already. Otherwise, inserts a new instance at the position indicated.

Table C-10 DEX Request and Response Dictionary Methods (continued)

Method	Syntax
putOption	**abool_t putOption(dex_AttributeDictionary_t** ***pDict*****, const char** ***pszValue*****, int** *iArgType*, ...)
Adds an option and its data or modifies the data for an option.	
putOptionBytes	**abool_t putOptionBytes(dex_AttributeDictionary_t** ***pDict*****, const abytes_t** ***pValue*****, int** *iArgType*, ...)
Adds an option and its data or modifies the data for an option.	
remove	**abool_t** *pDict***->remove(dex_AttributeDictionary_t** ***pDict*****, const char** ***pszAttribute*****, int** *iIndex*)
Removes the attribute from the dictionary. If *iIndex* is the special value DEX_REMOVE_ALL, removes any existing instances of the attribute. Otherwise, removes the instance at the position indicated. Returns TRUE, even if the dictionary did not contain that attribute at that index, unless the attribute name does not match any configured one.	
removeByType	**abool_t** *pDict***->removeByType(dex_AttributeDictionary_t** ***pDict*****, const abytes_t** ***pszAttribute*****, int** *iIndex*)
Removes the attribute from the dictionary. If *iIndex* is the value DEX_REMOVE_ALL, removes any existing instances of the attribute. Otherwise, removes the instance at the position indicated. Always returns TRUE, even if the dictionary does not contain that attribute at that index.	
removeOption	**abool_t removeOption(dex_AttributeDictionary** ***pDict*****, int** *iArgType*, ...)
Removes an option. Note that if you omit DEX_INDEX, a DEX_INDEX of DEX_REMOVE_ALL is assumed (this removes the entire option).	
setObject	**abool_t setObject(dex_AttributeDictionary_t** ***pDict*****, int** *iObjectType*, **int** *iObjArgType*, ...)
Sets the object for **get**, **put**, and **remove** methods, and alters the message on which the new option methods operate. DHCPv6 extensions primarily use this method to access the leases and prefixes available for the client and link, or to get message header fields or options from relay packets. Unlike in DHCPv4, where one lease and scope are associated with a response, a DHCPv6 response can involve several leases and prefixes. Returns TRUE if the object exists; FALSE otherwise. For sample usage, see the "Handling Object Data" section on page C-37.	
Note For leases not associated with the current client, only minimal information is available.	
trace	**abool_t** *pDict***->trace(dex_AttributeDictionary_t** ***pDict*****, int** *iLevel*, **const char** ***pszFormat*****,** ...)
Returns a message in the DHCP server packet tracing system. At level 0, no tracing occurs. At level 1, it traces only that the server received the packet and sent a reply. At level 4, it traces everything. The remaining arguments are concatenated and sent to the tracing system at the specified level. The default tracing is set using the DHCP server *extension-trace-level* attribute.	

Differences in get, put, Option, Bytes, and OptionBytes Methods

There are differences among the following DEX extension methods:

- **get** and **put**
- **getOption** and **putOption**

- **getBytes** and **putBytes**
- **getOptionBytes** and **putOptionBytes**

The **get** and **getOption** methods return the requested information formatted as a string. The server converts the data to the string depending on the expected data type for the dictionary item. If the data type is unknown, the server returns the data in blob string format.

The **getBytes** and **getOptionBytes** methods return the requested information as the raw bytes (a pointer to a buffer and the size of that buffer). The server should have to read this buffer only, and it contains only the data from the option (no null terminator has been added, for example).

The **put** and **putOption** methods expect the data to be written as a formatted string. The server converts the data from the string depending on the expected data type for the dictionary item. If the data type is unknown, it is expected to be in blob string format.

The server passes raw bytes to the **putBytes** and **putOptionBytes** methods (a pointer to a buffer and the size of that buffer). The server only reads these bytes.

Differences in get, put, remove, and ByType Methods

There are differences among the following DEX extension methods:

- **get**, **put**, and **remove**
- **getByType**, **putByType**, and **removeByType**

The server passes the **get**, **put**, and **remove** methods the name of the desired data item as a string. This requires that the server map the string to its internal data tables.

The server passes the **getByType**, **putByType**, and **removeByType** methods an internal data table reference, which the server must have previously obtained (such as in the extension init-entry) by calling the **getType** method on the string. This speeds processing for extensions, which can be important in applications requiring high performance.

Note The internal data table that the **getType** method references is the same whether requested for the Request or Response dictionary. There is no need for separate **getType** calls on each dictionary for the same data item name.

Table C-11 DEX iObjectType Values

iObjectType	Description
General definition:	Object for which the context is to be changed. (See also Table C-12 on page C-32.)
DEX_LEASE	Changes the lease (and prefix) context. Response dictionary only. Allows *iObjTypeArg*: DEX_BY_IPV6ADDRESS DEX_BY_IPV6PREFIX DEX_BY_INSTANCE DEX_INITIAL

Table C-11 DEX iObjectType Values (continued)

iObjectType	Description
DEX_MESSAGE	Changes the message context to a relay message or the client message. Request and response dictionaries. Allows *iObjArgType*: DEX_INITIAL DEX_RELAY DEX_BY_NUMBER
DEX_PREFIX	Changes the prefix context, but does not change the lease context. Response dictionary only. Allows *iObjTypeArg*: DEX_BY_IPV6ADDRESS DEX_BY_IPV6PREFIX DEX_BY_INSTANCE DEX_BY_NAME DEX_INITIAL

Table C-12 DEX iObjArgType Values

iObjArgType	Description
General definition: By what means the context is to be changed. (See also Table C-11 on page C-31.)	
DEX_BY_INSTANCE	Used with DEX_LEASE or DEX_PREFIX *iObjectType*. Requires that **int** follows to specify the instance number (starting with 0). Used to walk through the list of all available objects, but only through the list of objects applicable to the current request or response: for DEX_LEASE, the leases for that client (if any); for DEX_PREFIX, the prefixes on the current link (if any). Used with DEX_MESSAGE, a synonym for DEX_RELAY.
DEX_BY_IPV6ADDRESS	Used with DEX_LEASE and DEX_PREFIX *iObjectType* only. Requires that **const unsigned char *** follows to specify the 16-byte address.
DEX_BY_IPV6PREFIX	Used with DEX_LEASE or DEX_PREFIX *iObjectType*. Requires that **const unsigned char *** follows to specify a 17-byte prefix buffer (16-byte address followed by a 1-byte prefix length).
DEX_BY_NAME	Used with the DEX_PREFIX *iObjectType* only. Requires that a **const char *** follows to specify the name of the desired object.
DEX_INITIAL	Resets the context back to the original for the request or response, and has no additional argument. Sets the lease and prefix (DEX_LEASE), prefix (DEX_PREFIX), or message (DEX_MESSAGE) to what it was when the extension was originally called (which can be none).
DEX_RELAY	Used with DEX_MESSAGE *iObjectType* only. Requires that **int** follows to specify the relay (0 specifies the relay closest to the client). To set the message context back to the client, use **`setObject( pDict, DEX_MESSAGE, DEX_INITIAL )`**.

Table C-13 DEX iArgType Values

iArgType	Description
General definition: Action and argument that follows the context. There can be any number of *iArgType* instances in the calls. (See also Table C-11 on page C-31 and Table C-12 on page C-32.)	
DEX_ARG_ARRAY	Requires that a pointer to an array of **dex_OptionsArgs_t** follow, and is an alternative to specifying the argument list. Each **dex_OptionsArgs_t** structure has two fields: • *iArgType*—One of the *iArgType* DEX values in this table. • *pData*—Data (integer), pointer to the data (for strings and other data types), or ignored (if the *iArgType* takes no arguments). Note that once the server encounters the DEX_ARG_ARRAY (in an argument list or in an array of **dex_OptionsArgs_t**), it ignores any subsequent arguments in the original list.
DEX_END	**Note** Required, has no additional argument, and marks the end of the argument list.
DEX_ENTERPRISE_NAME	Requires that **const char *** follow to specify the option definition set name, from which the server extracts the enterprise-id to get the vendor option data. Valid only for vendor-identifying options. Requires that the vendor option definition set exists.
DEX_ENTERPRISE_ID	Requires that **int** follow to specify the enterprise-id for the vendor.
DEX_HOME	Moves the context back to the client or relay message options. Has no additional argument. Always returns success. If used, must be the first *iArgType*. Valid only for **getOption**, **getOptionBytes**, and **moveToOption** methods.

Table C-13 DEX iArgType Values (continued)

iArgType	Description
DEX_INDEX	Requires that **int** follow with the index of the option data (if any array of data is to be acted on). If omitted, index 0 is assumed, except for **removeOption**, in which case DEX_REMOVE_ALL is assumed. Use the special value DEX_RAW to get, put, or remove the entire option data. However, for the DHCPv4 Vendor-Identifying Vendor Options (RFC 3925 and RFC 4243), DEX_RAW returns the data for only one vendor (based on the instance or enterprise-id) and not that for the entire option. The DEX_RAW special value accesses the entire option (or suboption) data. It provides consistent access to the data, regardless of what the option definitions might specify in terms of the data type and repeat counts of the data type. **Tip** DEX_INDEX is useful for applications that need to assure that they always access the same data, regardless of the option definitions (because administrators or other components may alter those definitions). It is recommended for general-purpose extensions. Use the special values DEX_REPLACE (replace a value), DEX_APPEND (add to end), and DEX_AUGMENT (add if no value currently exists) with **putOption** and **putOptionBytes** methods, which operate the same as the **put**, **putByType**, **putBytes**, and **putBytesByType** methods. Use DEX_REMOVE_ALL for **removeOption** to remove the option completely.
DEX_INDEX_COUNT	Results in an **int** value returned with the count of the number of indexed entries of the option, rather than the option data. Has no additional argument, and cannot be used with DEX_INDEX or DEX_INSTANCE_COUNT. DEX_END must follow. Valid only for **getOption** and **getOptionBytes**.
DEX_INSTANCE	Requires that **int** follow to specify the instance of the option (valid only for DHCPv6 options, which can have more than one instance). 0 specifies the first instance.
DEX_INSTANCE_COUNT	Results in an **int** value returned with the count of the number of instances of the option, rather than the option data. Has no additional argument and cannot be used with DEX_INSTANCE. DEX_END must follow. Valid only for **getOption** and **getOptionBytes**.
DEX_MORE	Requires that **abool_t *** follow to specify the location at which a **more** flag is to be written. This location is set to TRUE if more array items exist beyond the index that DEX_INDEX specified. Valid only for **getOption** and **getOptionBytes** methods.

Table C-13 DEX iArgType Values (continued)

iArgType	Description
DEX_MOVE_TO	Leaves the context at the option or suboption immediately preceding DEX_MOVE_TO. Has no additional argument. If omitted, the context does not change. Use **moveToOption** to move the context without getting any data. Valid only for **getOption** and **getOptionBytes** methods. **Note** An attempt to move to an option or suboption that does not exist logs an error. Use **moveToOption** if your extension did not previously confirm that the option exists.
DEX_OPTION_NAME	Requires that **const char *** follow to specify the desired option name. Option names should be in the **dhcpv4-config** or **dhcpv6-config** option definition set.
DEX_OPTION_NUMBER	Requires that **int** follow to specify the desired option number. Option numbers should be in the **dhcpv4-config** or **dhcpv6-config** option definition set, although there is no requirement that a definition exists. However, if the option does not exist, it is assumed to be a byte blob of data.
DEX_PARENT	Moves the context to the parent option. Has no additional argument. It does not move beyond the client or relay message and returns FALSE if the context does not change. If used, must be the first *iArgType*. Valid only for **getOption**, **getOptionBytes**, and **moveToOption** methods.
DEX_SUBOPTION_NAME	Requires that **const char *** follow to specify the name of the desired suboption. Suboptions must be in the current option definition.
DEX_SUBOPTION_NUMBER	Requires that **int** follow to specify the desired suboption number. Suboption numbers should be in the current option definition, although there is no requirement that a definition exists. However, if the suboption does not exist, it is assumed to be a byte blob of data.
DEX_VENDOR_NAME	Requires that **const char *** follow to specify the vendor string. The string serves only to find the appropriate option definition set.

DEX Environment Dictionary Methods

The environment dictionary uses active commands, called methods, with which you can change and access the dictionary values. Table C-14 lists the methods to use with the environment dictionary. In this case, you can define the *pDict* variable as **pEnviron**, as in:

```
pEnviron->log( pEnviron, DEX_LOG_INFO, "Environment hello world" );
```

Table C-14 DEX Environment Dictionary Methods

Method	Syntax
allocateMemory	**void** **pDict*->**allocateMemory(** **dex_EnvironmentDictionary_t** **pDict*, **unsigned int** *iSize* **)**
Allocates memory for extensions that persists only for the lifetime of this request.	

Table C-14 DEX Environment Dictionary Methods (continued)

Method	Syntax
clear	**void** *pDict*->**clear(dex_EnvironmentDictionary_t** ***pDict** **)**
Removes all entries from the dictionary.	
containsKey	**abool_t** *pDict*->**containsKey(dex_EnvironmentDictionary_t** ***pDict*, **const char** ***pszKey** **)**
Returns TRUE if the dictionary contains the key, otherwise FALSE.	
firstKey	**const char** ***pDict*->**firstKey(dex_EnvironmentDictionary_t** ***pDict** **)**
Returns the name of the first key in the dictionary. Note that the keys are not stored sorted by name. If a key does not exist, returns zero.	
get	**const char** ***pDict*->**get(dex_EnvironmentDictionary_t** ***pDict*, **const char** ***pszKey** **)**
Returns the value of the key from the dictionary. If a key does not exist, returns the empty string.	
isEmpty	**abool_t** *pDict*->**isEmpty(dex_EnvironmentDictionary_t** ***pDict** **)**
Returns TRUE if the dictionary has 0 entries, otherwise FALSE.	
log	**abool_t** *pDict*->**log(dex_EnvironmentDictionary_t** ***pDict*, **int** *eLevel*, **const char** ***pszFormat*, ... **)**
Returns a message in the DHCP server logging system. The *eLevel* should be one of DEX_LOG_ERROR, DEX_LOG_WARNING, or DEX_LOG_INFO. The *pszFormat* is treated as a printf style format string, and it, along with the remaining arguments, are formatted and sent to the logging system at the specified level.	
Note Use the DEX_LOG_ERROR and DEX_LOG_WARNING levels sparingly, because the server flushes its log file with messages logged at these levels. Using these levels for messages that are likely to occur frequently (such as client requests) can have severe impact on disk I/O performance.	
nextKey	**const char** ***pDict*->**nextKey(dex_EnvironmentDictionary_t** ***pDict** **)**
Returns the name of the next key in the dictionary that follows the key returned in the last call to **firstKey** or **nextKey**. If a key does not exist, returns zero.	
put	**abool_t** *pDict*->**put(dex_EnvironmentDictionary_t** ***pDict*, **const char** ***pszKey*, **const char*** *pszValue* **)**
Associates a value with the key, replacing an existing instance of the key with the new value.	
remove	**abool_t** *pDict*->**remove(dex_EnvironmentDictionary_t** ***pDict*, **const char** ***pszKey** **)**
Removes the key and the associated value from the dictionary. Always returns TRUE, even if the dictionary did not contain the key.	
size	**int** *pDict*->**size(dex_EnvironmentDictionary_t** ***pDict** **)**
Returns the number of entries in the dictionary.	
trace	**abool_t** *pDict*->**trace(dex_EnvironmentDictionary_t** ***pDict*, **int** *iLevel*, **const char** ***pszFormat*, ... **)**
Returns a message in the DHCP server packet tracing system. At level 0, no tracing occurs. At level 1, it traces only that the server received the packet and sent a reply. At level 4, it traces everything. The remaining arguments are concatenated and sent to the tracing system at the specified level. The default tracing is set using the DHCP server *extension-trace-level* attribute.	

Examples of Option and Object Method Calls

These sections include some examples of how to use methods to handle DHCP option and object data.

Handling Vendor Class Option Data

For DHCPv4, to include the Vendor-Identifying Vendor Class option (124) data for two enterprise-ids in the response to the client, here is some sample Tcl code that uses the **putOption** method:

```
$response putOption 65:66:67 option 124 enterprise 999998 #adds "abc" (65:66:67) under
enterprise-id 999998
$response putOption 68:69:6a:6b option v-i-vendor-class enterprise 999998 index append
#appends "defg" (68:69:6a:6b) under the same enterprise-id
$response putOption 01:02:03:04 option 124 enterprise 999999 #adds 01:02:03:04 under
enterprise-id 999999
```

To get the options, use the **getOption** method:

```
$response getOption option v-i-vendor-class instance-count #returns 2 because there were
two instances added (enterprise id 999998 and enterprise id 999999)
$response getOption option 124 #returns index 0 of instance 0, which is 65:66:67
$response getOption option 124 index-count #returns 2 because there were two vendor
classes added for the first enterprise id (9999998)
$response getOption option 124 index raw #returns
00:0f:42:3e:09:03:65:66:67:04:68:69:6a:6b for the complete encoding of the enterprise-id
999998 data (see RFC 3925)
$response getOption option 124 index 1 #returns 68:69:6a:6b
$response getOption option 124 instance 1 index-count #returns 1 because there is only one
vendor class
$response getOption option 124 instance 1 index raw #returns 00:0f:42:3f:05:04:01:02:03:04
for the complete encoding of the enterprise-id 999999 data (see RFC 3925)
$response getOption option 124 enterprise 999999 #returns 01:02:03:04
```

To remove the data, two **removeOption** calls are necessary because of the two separate enterprise-ids:

```
$response removeOption option 124
$response removeOption option 124
```

Handling Object Data

Suppose that at the **pre-packet-encode** extension point you want to extract data for all of the leases for the client. Here is sample Tcl code that uses the **setObject** method:

```
proc logleasesinit { request response environ } {
    if { [$environ get "extension-point"] == "initialize" } {
        # Set up for DHCPv6 only]
        $environ put dhcp-support "v6"
        $environ put extension-extensionapi-version 2
    }
}
proc logleases { request response environ } {
    for { set i 0 } { 1 } { incr i } {
        # Set context to next lease
        if { ![$response setObject lease $i] } {
        # Lease does not exist, so done
        break
    }
    # Log the lease address, prefix name, and prefix address
```

```
    $environ log LOG_INFO "Lease [$response get lease-ipaddress], Prefix\
        [$response get lease-prefix-name] - [$response get prefix-address]"
    }
    # Restore the lease context to where we started
        $response setObject lease initial
        # Do other things...
}
```

The C++ equivalent code for this might be:

```
// Print the current leases for the client
for( int i=0; ; i++ ) {
    if( !pRes->setObject( pRes, DEX_LEASE, DEX_BY_INSTANCE, i ) )
        break;
    const char *pszLeaseAddress =
        pRes->get( pRes, "lease-ipaddress", 0, 0 );
    if( pszLeaseAddress == 0 )
        pszLeaseAddress = "<error>";
    const char *pszPrefixName =
        pRes->get( pRes, "prefix-name", 0, 0 );
    if( pszPrefixName == 0 )
        pszPrefixName = "<error>";
    pEnv->log(pEnv, DEX_LOG_INFO,
        "Lease %s, Prefix %s",
        pszLeaseAddress, pszPrefixName );
}
```



GLOSSARY

A

A record — DNS Address resource record (RR). Maps a hostname to its address and specifies the Internet Protocol address (in dotted decimal form) of the host. There should be one A record for each host address.

access control list (ACL) — DHCP mechanism whereby the server can allow or disallow the request or action defined in a packet. *See also* transaction signature (TSIG).

address block — Block of IP addresses to use with DHCP subnet allocation that uses on-demand address pools.

admin — Default name of the superuser or global administrator.

administrator — User account to adopt certain functionality, be it defined by role, constrained role, or group.

alias — Pointer from one domain name to the official (canonical) domain name.

allocation priority — An alternate method of control over allocating addresses among scopes other than the default round-robin method.

ARIN — American Registry of Internet Numbers, one of several regional Internet Registries (IRs), manages IP resources in North America, parts of the Caribbean, and subequatorial Africa. Network Registrar provides an address space report for this registry.

Asynchronous Transfer Mode (ATM) — International standard for cell relay in which multiple service types (such as voice, video, or data) are conveyed in fixed-length (53-byte) cells.

authoritative name server — DNS name server that possesses complete information about a zone.

AXFR — Full DNS zone transfer. *See also* zone transfer and IXFR.

B

Berkeley Internet Name Domain (BIND) — Implementation of the Domain Name System (DNS) protocols. *See also* DNS.

binding — Collection of DHCP client options and lease information, managed by the main and backup DHCP servers. A binding database is a collection of configuration parameters associated with all DHCP clients. This database holds configuration information about all the datasets.

BOOTP — Bootstrap Protocol. Used by a network node to determine the IP address of its Ethernet interfaces, so that it can affect network booting.

C

cable modem termination system (CMTS) Cable modem termination system. Either a router or bridge, typically at the cable head end.

cache Data stored in indexed disk files to reduce the amount of physical memory.

caching name server Type of DNS server that caches information learned from other name servers so that it can answer requests quickly, without having to query other servers for each transaction.

canonical name Another name for an alias DNS host, inherent in a CNAME resource record (RR).

case sensitivity Values in Network Registrar are not case sensitive, with the exception of passwords.

Central Configuration Management (CCM) database Main database for the Network Registrar web-based user interface (web UI).

chaddr DHCP client hardware (MAC) address. Sent in an RFC 2131 packet between the client and server.

change logs, changesets A change log is a group of changesets made to the Network Registrar databases due to additions, modifications or deletions in the web UI. A changeset is a set of changes made to a single object in the database.

ciaddr DHCP client IP address. Sent in an RFC 2131 packet between the client and server.

class of address Category of an IP address that determines the location of the boundary between network prefix and host suffix. Internet addresses can be A, B, C, D, or E level addresses. Class D addresses are used for multicasting and are not used on hosts. Class E addresses are for experimental use only.

client-class Cisco Network Registrar feature that provides differentiated services to users that are connected to a common network. You can thereby group your user community based on administrative criteria, and then ensure that each user receives the appropriate class of service.

cluster In Network Registrar, a group of DNS, DHCP, and TFTP servers that share the same database.

CNAME record DNS Canonical Name resource record (RR). Used for nicknames or aliases. The name associated with the resource record is the nickname. The data portion is the official or canonical name.

CNRDB Name of one of the Network Registrar internal databases. The others are changeset database and MCD.

constraint Assigned limitation on the role or allowable functionality of an administrator.

D

Data Over Cable Service Interface Specification (DOCSIS) Data Over Cable Service Interface Specification. Standard created by cable companies in 1995 to work toward an open cable system standard and that resulted in specifications for connection points, called interfaces.

delegation Act of assigning responsibility for managing a DNS subzone to another server, or of assigning DHCP address blocks to local clusters.

DHCP Dynamic Host Configuration Protocol. Designed by the Internet Engineering Task Force (IETF) to reduce the amount of configuration that is required when using TCP/IP. DHCP allocates IP addresses to hosts. It also provides all the parameters that hosts require to operate and exchange information on the Internet network to which they are attached.

Digital Subscriber Line (DSL) Public network technology that delivers high bandwidth over conventional copper wiring at limited distances.

DNS Domain Name System. Handles the growing number of Internet users. DNS translates names, such as www.cisco.com, into Internet Protocol (IP) addresses, such as 192.168.40.0, so that computers can communicate with each other.

DNS update Protocol (RFC 2136) that integrates DNS with DHCP.

domain Portion of the DNS naming hierarchy tree that refers to general groupings of networks based on organization type or geography. The hierarchy is root, top- or first-level, and second-level domain.

domain name DNS name that can be either absolute or relative. An absolute name is the fully qualified domain name (FQDN) and is terminated with a period. A relative name is relative to the current domain and does not end with a period.

dotted decimal notation Syntactic representation of a 32-bit integer that consists of four eight-bit numbers written in base 10 with dots separating them for a representation of IP addresses. Many TCP/IP application programs accept dotted decimal notation in place of destination machine names.

E

expression Construct commonly used in the Network Registrar DHCP implementation to create client identities or look up clients. For example, an expression can be used to construct a scope from a template.

extension and extension point In Network Registrar, element of a script written in TCP, C, or C++ that customizes handling DHCP packets as the server processes them, and which supports additional levels of customizing DHCP clients.

F

failover Network Registrar feature (as described in RFC 2131) that provides for multiple, redundant DHCP servers, whereby one server can take over in case of a failure. DHCP clients can continue to keep and renew their leases without needing to know or care which server is responding to their requests.

forwarder DNS server designated to handle all offsite queries. Using forwarders relieves other DNS servers from having to send packets offsite.

forwarding, DHCP Mechanism of forwarding DHCP packets to another DHCP server on a per-client basis. You can achieve this in Network Registrar by using extension scripting.

FQDN Fully qualified domain name. Absolute domain name that unambiguously specifies a host location in the DNS hierarchy.

G

giaddr DHCP gateway (relay agent) IP address. Sent in an RFC 2131 packet between the client and server.

glue record DNS Address resource record that specifies the address of a subdomain authoritative name server. You only need glue records in the server delegating a domain, not in the domain itself.

group Associative entity that combines administrators so that they can be assigned roles and constrained roles.

H

High-Availability (HA) DNS DNS configuration in which a second primary server can be made available as a hot standby that shadows the main primary server, as described in RFC 2136.

HINFO record DNS Host Information resource record (RR). Provides information about the hardware and software of the host machine.

hint server *See* root hint server.

host Any network device with a TCP/IP network address.

I

IEEE Institute of Electrical and Electronics Engineers. Professional organization whose activities include developing communications and network standards.

in-addr.arpa DNS address mapping domain with which you can index host addresses and names. The Internet can thereby convert IP addresses back to hostnames. *See also* reverse zone.

IP address Internet Protocol address. For example, 192.168.40.123.

IP history Network Registrar tool that records the lease history of IP addresses in a database.

IPv6 New IP standard involving 128-bit addresses. Network Registrar provides a DHCPv6 implementation.

ISP Internet Service Provider. Company that provides leased line, dialup, and DSL (Point-to-Point over Ethernet and DHCP) access to customers.

iterative query Type of DNS query whereby the name server returns the closest answer to the querying server.

IXFR Incremental zone transfer. Standard that allows Network Registrar to update a slave (secondary) server by transferring only the changed data from the primary server.

L

lame delegation Condition when DNS servers listed in a zone are not configured to be authoritative for the zone.

LDAP Lightweight Directory Access Protocol. Method that provides directory services to integrate Network Registrar client and lease information.

lease IP address assignment to a DHCP client that also specifies how long the client can use the address. When the lease expires, the client must negotiate a new one with the DHCP server.

lease grace period Length of time the lease is retained in the DHCP server database after it expires. This protects a client lease in case the client and server are in different time zones, their clocks are not synchronized, or the client is not on the network when the lease expires.

lease history A report that can be generated to provide a historical view of when a client was issued a lease, for how long, when the client or server released the lease before it expired, and if and when the server renewed the lease and for how long.

lease query Process by which a relay agent can request lease (and reservation) data directly from a DHCP server in addition to gleaning it from client/server transactions.

local cluster Location of the local Network Registrar servers. *See also* regional cluster.

localhost Distinguished name referring to the name of the current machine. Localhost is useful for applications requiring a hostname.

loopback zone DNS zone that enables the server to direct traffic to itself. The host number is almost always 127.0.0.1.

M

MAC address Standardized data link layer address. Required for every port or device that connects to a LAN. Other devices in the network use these addresses to locate specific ports on the network and to create and update routing tables and data structures. MAC addresses are six bytes long and are controlled by the IEEE. Also known as a hardware address, MAC layer address, and physical address. A typical MAC address is 1,6,00:d0:ba:d3:bd:3b.

mail exchanger Host that accepts electronic mail, some of which act as mail forwarders. *See also* MX record.

master name server Authoritative DNS name server that transfers zone data to secondary servers through zone transfers.

maximum client lead time (MCLT) In DHCP failover, a type of lease insurance that controls how much ahead of the backup server lease expiration the client lease expiration should be.

MCD Name of one of the Network Registrar internal databases. The other is CNRDB.

mcdshadow Network Registrar database backup utility. Other utilities include **cnr_exim**, **mcdadmin**, **cnrdb_recover**, **cnrdb_verify**, **cnrdb_checkpoint**, **keybuild**, **dbcheck**, and **cnr_zone_recovery**.

multinetting State of having multiple DHCP scopes on one subnet or several LAN segments.

Multiple Service Operator (MSO) Provides subscribers Internet access using cable or wireless technologies.

multithreading Process of performing multiple server tasks.

MX record DNS Mail Exchanger resource record (RR). Specifies where mail for a domain name should be delivered. You can have multiple MX records for a single domain name, ranked in preference order.

N

nameserver DNS host that stores data and RRs for a domain.

NAPTR DNS Naming Authority Pointer resource record (RR). Helps with name resolution in a particular namespace and is processed to get to a resolution service. Based on proposed standard RFC 2915.

negative cache time Memory cache the DNS server maintains for a quick response to repeated requests for negative information, such as “no such name” or “no such data.” Network Registrar discards this information at intervals.

network ID Portion of the 32-bit IP address that identifies which network a particular system is on, determined by performing an AND operation of the subnet mask and the IP address.

NOTIFY Standard (RFC 1996) whereby DNS master servers can inform their slaves that changes were made to their zones, and which initiates a zone transfer.

nrcmd Network Registrar command line interface (CLI).

O

on-demand address pool Wholesale IP address pool issued to a client (usually a VPN router or other provisioning device), from which it can draw for lease assignments. Also known as DHCP subnet allocation.

option, DHCP DHCP configuration parameter and other control information stored in the options field of a DHCP message. DHCP clients determine what options get requested and sent in a DHCP packet. Network Registrar allows for creating option definitions as well as the option sets to which they belong.

Organization report One of the reports to be submitted to ARIN, POC being the other report. *See also* ARIN and POC report.

Organizationally Unique Identifier (OUI) Assigned by the IEEE to identify the owner or ISP of a VPN. *See also* IEEE and virtual private network (VPN).

owner Owners can be created as distinguishing factors for address blocks, subnets, and zones. In the context or DNS RRs, an owner is the name of the RR.

P

ping Packet Internetwork Groper. A common method for troubleshooting device accessibility that uses a series of Internet Control Message Protocol (ICMP) Echo messages to determine if a remote host is active or inactive, and the round-trip delay in communicating with the host.

POC report Point of Contact report. One of the reports to be submitted to ARIN, Organization being the other report. *See also* ARIN and Organization report.

policy Group of DHCP attributes or options applied to a single scope or group of scopes. Embedded policies can be created for scopes and other DHCP objects.

polling Collection of subnet utilization or lease history data over a certain regular period.

primary master DNS server from which a secondary server receive data through a zone transfer request.

provisional address Address allocated by the DHCP server to an unknown clients for a short time, one-shot basis.

PTR record DNS Pointer resource record. Used to enable special names to point to some other location in the domain tree. Should refer to official (canonical) names and not aliases. *See also* in-addr.arpa.

pulling and pushing objects The Network Registrar regional cluster provides functions to pull network objects from the replica database of local cluster data, and push objects directly to the local clusters.

R

recursive query DNS query where the name server asks other DNS server for any nonauthoritative data not in its own cache. Recursive queries continue to query all name servers until receiving an answer or an error.

refresh interval Time interval in which a secondary DNS server checks the accuracy of its data by sending an AXFR packet to the primary server.

region Regions can be created as distinguishing factors for address blocks, subnets, and zones. A region is distinct from the regional cluster.

regional cluster Location of the regional Network Registrar CCM server. *See also* local cluster.

relay agent Device that connects two or more networks or network systems. In DHCP, a router on a virtual private network that is the IP helper for the DHCP server.

replica database CCM database that captures copies of local cluster configurations at the regional cluster. These configurations can be pulled to the regional cluster so that they can be pushed to other local clusters.

Request for Comments (RFC) TCP/IP set of standards.

reservation IP address or lease that is reserved for a specific DHCP client.

resolution exception Selectively forwarding DNS queries for specified domains to internal servers rather than recursively querying Internet root name and external servers.

resolver Client part of the DNS client/server mechanism. A resolver creates queries sent across a network to a name server, interprets responses, and returns information to the requesting programs.

resource record (RR) DNS configuration record, such as SOA, NS, A, CNAME, HINFO, WKS, MX, and PTR that comprises the data within a DNS zone. Mostly abbreviated as RR. *See* Appendix A, "Resource Records."

reverse zone DNS zone that uses names as addresses to support address queries. *See also* in-addr.arpa.

RIC server The Network Registrar Router Interface Configuration (RIC) server that manages router interfaces on Cisco Systems Universal Broadband Routers (uBRs) that manage cable modem termination systems (CMTSs). *See also* cable modem termination system (CMTS).

role, constrained role Administrators can be assigned one or more roles to determine what functionality they have in the application. A constrained role is a role constrained by further limitations. There are general roles for DNS, host, address block, DHCP, and CCM database administration. You can further constrain roles for specific hosts and zones. Some roles have distinguishing subroles, such as the database subrole.

root hint server DNS name server at the top of the hierarchy for all root name queries. A root name server knows the addresses of the authoritative name servers for all the top-level domains. Resolution of nonauthoritative or uncached data must start at the root servers. Sometimes called a hint server.

round-robin Action when a DNS server rearranges the order of its multiple same-type records each time it is queried.

routed bridge encapsulation (RBE) Process by which a stub-bridged segment is terminated on a point-to-point routed interface. Specifically, the router is routing on an IEEE 802.3 or Ethernet header carried over a point-to-point protocol, such as PPP, RFC 1483 ATM, or RFC 1490 Frame Relay.

S

scavenging Action of periodically scanning dynamic updates to the DNS server for stale resource records and purging these records.

scope Administrative grouping of TCP/IP addresses on a DHCP server. Required for lease assignments.

secondary master DNS name server that gets it zone data from another name server authoritative for the zone. When a secondary master server starts up, it contacts the primary master, from which it receives updates.

secondary subnet A single LAN might have more than one subnet number applicable to the same LAN or network segment in a router. Typically, one subnet is designated as primary, the others as secondary. A site might support addresses on more than one subnet number associated with a single interface. You must configure the DHCP server with the necessary information about your secondary subnets.

selection tags Mechanisms that help select DHCP scopes for clients and client-classes.

siaddr IP address of the server to use in the next step of the DHCP boot process. Sent in an RFC 2131 packet between the client and server.

slave forwarder DNS server that behaves like a stub resolver and passes most queries on to another name server for resolution. *See also* stub resolver.

slave servers DNS server that always forwards queries it cannot answer from its cache to a fixed list of forwarding servers instead of querying the root name servers for answers.

SNMP notification Simple Network Management Protocol messages that warn of server error conditions and problems. *See also* trap.

SOA record DNS Start of Authority resource record (RR). Designates the start of a zone.

SRV record Type of DNS resource record (RR) that allows administrators to use several servers for a single host domain, to move services from host to host with little difficulty, and to designate some hosts as primary servers for a service and others as backups.

staged edit mode Scope or zone edit mode in which the data is stored on the CCM server, but not live on the protocol server. *See also* synchronous edit mode.

stub resolver DNS server that hands off queries to another server instead of performing the full resolution itself.

subnet allocation, DHCP Network Registrar use of on-demand address pools for entire subnet allocation of IP addresses to provisioning devices.

subnet mask Separate IP address, or part of a host IP address, that determines the host address subnet. For example, 192.168.40.0 255.255.255.0 (or 192.168.40.0/24) indicates that the first 24 bits of the IP address are its subnet, 192.168.40. In this way, addresses do not need to be divided strictly along network class lines.

subnet pool Set of IP addresses associated with a network number and subnet mask, including secondary subnets.

subnet sorting Attribute of the Network Registrar DNS server. By enabling it, the server checks the network address of the client before responding to a query.

subnet utilization A report that can be generated to determine how many addresses in the subnet were allocated and what the free address space is.

subnetting Action of dividing any network class into multiple subnetworks.

subscriber limitation Limitation to the number of addresses service providers can determine for the DHCP server to give out to devices on customer premises, handled in Network Registrar by DHCP option 82 definitions.

subzone Partition of a delegated domain, represented as a child of the parent node. A subzone always ends with the name of its parent. For example, boston.example.com. can be a subzone of example.com.

subzone delegation Dividing a zone into subzones. You can delegate administrative authority for these subzones, and have them managed by people within those zones or served by separate servers.

supernet Aggregation of IP network addresses advertised as a single classless network address.

synchronization Synchronization can occur between the regional cluster and local clusters, the CCM and other protocol servers, failover servers, HA DNS servers, and routers.

synchronous edit mode Scope or zone edit mode in which the data is live on the protocol server. *See also* staged edit mode.

T

TAC Cisco Technical Assistance Center. Network Registrar provide a **cnr_tactool** utility to use in reporting issues to the TAC.

TCP/IP Suite of data communication protocols. Its name comes from two of the more important protocols in the suite: the Transmission Control Protocol (TCP) and the Internet Protocol (IP). It forms the basis of Internet traffic.

template DNS zones and DHCP scopes can have templates to create multiple objects with similar properties.

transaction signature (TSIG) DHCP mechanism that ensures that DNS messages come from a trusted source and are not tampered with. *See also* access control list (ACL).

trap Criteria set to detect certain SNMP events, such as to determine free addresses on the network. *See also* SNMP notification.

trimming and compacting Trimming is periodic elimination of old historical data to regulate the size of log and other files. Compacting is reducing data older than a certain age to subsets of the records.

Trivial File Transfer Protocol (TFTP) Protocol used to transfer files across the network using UDP. *See also* User Datagram Protocol (UDP).

U

Universal Time (UT) International standard time reference that was formerly called Greenwich Mean Time (GMT), also called Universal Coordinated Time (UCT).

update configuration, DNS Defines the relationship of a zone with its main and backup DNS servers for DNS update purposes.

update map, DNS Defines an update relationship between a DHCP policy and a list of DNS zones.

update policy, DNS Provide a mechanism in DHCP for managing update authorization at the DNS RR level.

User Datagram Protocol (UDP) Connectionless TCP/IP transport layer protocol.

V

virtual channel identifier (VCI) and virtual path identifier (VPI) 16-bit field in the header of an ATM cell. The VCI, together with the VPI, identifies the next destination of a cell as it passes through a series of ATM switches on its way to its destination. ATM switches use the VPI/VCI fields to identify the next network VCL that a cell needs to transit on its way to its final destination. The function of the VCI is similar to that of the DLCI in Frame Relay.

virtual private network (VPN) Protocol over which IP traffic of private address space can travel securely over a public TCP/IP network. A VPN uses tunneling to encrypt all information at the IP level. *See also* VRF.

VRF VPN Routing and Forwarding instance. Routing table and forwarding information base table, populated by routing protocol contexts. *See also* virtual private network (VPN).

W

well-known port Any set of IP protocol port numbers preassigned for specific uses by transport level protocols, for example, TCP and UDP. Each server listens at a well-known port so clients can locate it.

WKS record DNS Well Known Service resource record (RR). Used to list the services provided by the hosts in a zone. Common protocols are TCP and UDP.

Y

yiaddr “Your” client IP address, or address that the DHCP server offers (and ultimately assigns) the client. Sent in an RFC 2131 packet between the client and server.

Z

zone Delegation point in the DNS tree hierarchy that contains all the names from a certain point downward, except for those names that were delegated to other zones. A zone defines the contents of a contiguous section of the domain space, usually bounded by administrative boundaries. Each zone has configuration data composed of entries called resource records. A zone can map exactly to a single domain, but can also include only part of a domain, with the remainder delegated to another subzone.

zone distribution Configuration that simplifies creating multiple zones that share the same secondary zone attributes. The zone distribution requires adding one or more predefined secondary servers.

zone of authority Group of DNS domains for which a given name server is an authority.

zone transfer Action that occurs when a secondary DNS server starts up and updates itself from the primary server. A secondary DNS server queries a primary name server with a specific packet type called AXFR (transfer all) or IXFR (incrementally transfer) and initiates a transfer of a copy of the database.



INDEX

A

A records A-1, 1-1
  adding 16-3
  rogue 17-16
A6 records A-2
AAAA records A-2
  DNS update 28-2
absolute domain names
  See FQDN
access control lists (ACLs)
  creating 28-9
acl command (CLI)
  create 28-10
ACLs
  See access control lists (ACLs)
addrblock-admin role 5-3
  core functionality 5-3
  ipv6-management subrole 5-3
  ric-management subrole 5-3
address allocation
  attributes 20-14
  in scopes 20-16
  priority 20-12
  round-robin 20-12
address blocks 9-4
  adding 9-5
  administrator role 9-1
  children 9-10
  delegating 9-7
  embedded policies 23-20
  viewing 9-4
  when to add 9-5
address blocks, DHCP 19-5
  creating 23-20
  default subnet size 23-20
  orphaned leases 23-21
  policies, associating 23-20
address infrastructure, creating 5-18
address pools, on-demand
  See subnet allocation, DHCP
address ranges
  adding 5-18
  scopes 20-11
  subnets 9-11
Address records
  See A records
address restrictions, zones 5-22
address space
  delegated 9-4, 9-5
  local, pulling from 5-31
  managing 9-1
  unified 9-2
address usage reports
  displaying 7-15
  running 22-21
address-block-policy command (CLI)
  delete 23-20
  get 23-20
  getOption 23-20
  listOptions 23-20
  listVendorOptions 23-20
  setVendorOption 23-20
  show 23-20
  unset 23-20
  unsetOption 23-20

unsetVendorOption 23-20
addresses
dynamic 9-1
IP format 1-4
IPv6 26-2
provisional 24-12
static 9-1
usage, displaying 7-15
addr-trap command (CLI)
create 1-10, 20-22
set
high-threshold 20-22
low-threshold 20-22
admin command (CLI)
create 5-6
delete 5-6
enterPassword 5-7
list 5-6
listnames 5-6
set password 5-7
admin role 1-1
administrators 5-1, 1-1
adding 5-5
centrally managing 5-8
local cluster 5-16
passwords
adding 5-5
changing 5-7
managing 5-7
pulling replica 5-10
pushing to local 5-9
regional 5-26
relationship to groups 5-1
types 5-2
AFSDB records A-2
agent_server_log file 7-8
allocation priority, scopes 20-12
algorithm 20-13
limitation-id and 20-14
all-subnets-local, DHCP option B-4
and, DHCP expression 25-6
arithmetic functions, DHCP expressions 25-19
arp-cache-timeout, DHCP option B-5
as-blob, DHCP expression 25-6
ash, DHCP expression 25-7
as-sint, DHCP expression 25-6
associated-ip, DHCP option B-17
as-string, DHCP expression 25-7
as-uint, DHCP expression 25-7
Asynchronous Transfer Mode (ATM) 25-23, 1-1
AT_BLOB option validation B-24
AT_BOOL option validation B-24
AT_DATE option validation B-24
AT_DNSNAME option validation B-24
AT_INT option validation B-24
AT_INT8 option validation B-24
AT_INTI option validation B-24
AT_IP6ADDR option validation B-24
AT_IPADDR option validation B-24
AT_MACADDR option validation B-24
AT_MESSAGE option validation B-24
AT_NOLEN option validation B-24
AT_NSTRING option validation B-24
AT_OVERLOAD option validation B-24
AT_RANGEBYTE option validation B-24
AT_RANGESHORT option validation B-24
AT_RDNSNAME option validation B-24
AT_SHORT option validation B-24
AT_SHRTI option validation B-24
AT_SINT option validation B-24
AT_SINTI option validation B-24
AT_SSHORT option validation B-24
AT_SSHRTI option validation B-24
AT_STIME option validation B-24
AT_STRING option validation B-24
AT_TIME option validation B-24
AT_TYPECNT option validation B-25
AT_VENDOR_CLASS option validation B-25

AT_VENDOR_OPTS option validation B-25
AT_ZEROSIZE option validation B-25
attributes
displaying 2-5
Help window 2-6
modifying 2-5
visibility 2-5
auth, DHCPv6 option B-11
authentication subrole (ccm-admin) 5-3
authentication subrole (regional-admin) 5-4
authentication, DHCP option B-17
authoritative name servers 14-3, 1-1
adding 15-7
authorization subrole (ccm-admin) 5-3
authorization subrole (regional-admin) 5-4
auto-configure, DHCP option B-17

B

base roles 5-1
base32encode, DHCP expression 25-7
bcms-server-a, DHCPv6 option B-12
bcms-server-d, DHCPv6 option B-12
bcms-servers-a, DHCP option B-10
bcms-servers-d, DHCP option B-10
BIND files
format 15-9
importing 15-9
bit-and/or, DHCP expression 25-7
bit-not, DHCP expression 25-8
boot-file, DHCP option B-10
BOOTP
clients, moving/decommissioning 20-20
configuring 23-1
requirements 23-2
dynamic 27-22
enabling 23-3
PARTNER-DOWN state 27-14
scopes, enabling for 20-21
enabling, disabling 23-2
failover 27-22
file reply packet field 23-2
relay 23-3
failover 27-4
routers 23-3
scopes, enabling for 20-20
siaddr reply packet field 23-2
sname reply packet field 23-2
static 27-22
boot-size, DHCP option B-2
broadcast-address, DHCP option B-4
byte, DHCP expression 25-8

C

C/C++
API 29-9
extensions 29-4, 29-9
cable modem termination system (CMTS) 4-2
cablelabs-125, DHCP option C-2
cablelabs-17, DHCPv6 option C-7
cablelabs-client-configuration, DHCP option B-18
cache
attributes 17-16
flushing DNS 7-3, 17-15
caching-only servers 14-5
adding 17-6
case-sensitivity of values 1-2
catalina_log.*date*.txt file 7-8
ccm command (CLI)
polling attribute, setting 6-10
set 7-4
CCM database 1-2
backing up 8-2
files 7-8
logging 7-8
CCM server 1-1, 4-2
polling attributes 6-10

ccm-admin role 5-3
authentication subrole 5-3
authorization subrole 5-3
core functionality 5-3
database subrole 5-3
owner-region subrole 5-3
server-management subrole 5-3
ccm-management subrole (cfg-admin) 5-3
central configuration 6-1
Central Configuration Management (CCM) server
See CCM server
central-cfg-admin role 5-4
core functionality 5-4
dhcp-management subrole 5-4
ric-management subrole 5-4
central-dns-admin role 5-4
core functionality 5-4
security-management subrole 5-4
server-management subrole 5-4
central-host-admin role 5-4
core functionality 5-4
cfg-admin role 5-3
ccm-management subrole 5-3
core functionality 5-3
dhcp-management subrole 5-3
dns-management subrole 5-3
ric-management subrole 5-3
snmp-management subrole 5-3
tftp-management subrole 5-3
chaddr, DHCP field 1-2
change set database, DNS update 17-10
checkpoint interval 17-10
check-lease-acceptable extension point, DHCP 23-10, 29-36
checkpointing, zone
file
backing up 8-7
dumping 17-10
forcing 17-10
checkports_log file 7-8
child
address blocks 9-10
subnets 9-10
ciaddr, DHCP field 23-16, 1-2
Cisco routers 23-3
cisco-auto-configure, DHCP option B-18
cisco-client-last-transaction-time, DHCP option B-18
cisco-client-requested-host-name, DHCP option B-18
cisco-leased-ip, DHCP option B-18
cisco-vpn-id, DHCP option B-18
class of service 19-11
classless-static-route, DHCP option B-18
CLI 1-1, 2-12
command syntax 2-13
client command (CLI)
create 24-10
delete 24-10
list 24-10
listnames 24-10
set 24-10
action=one-shot 24-12
authenticate-until 24-13
client-class-name 24-10
client-lookup-id 25-1
default-vpn 23-18
limitation-id 24-16
limitation-key 25-1
over-limit-client-class-name 24-16
override-vpn 23-18
policy-name 24-12
selection-criteria 24-10
show 24-10
client ID
overriding 22-17
reservations, using for lease 22-17

client-class command (CLI)
create 24-3
delete 24-3
list 24-3
listnames 24-3
set 24-3
add-to-environment-dictionary 22-17
default-vpn 23-18
host-name 24-4
limitation-id 24-16
limitation-key 25-1
over-limit-client-class-name 24-16
override-client-id 22-17
override-vpn 23-18
selection-criteria 24-3
show 24-3
client-classes
client entries, skipping 24-13
configuring 24-1
defining 24-2
DHCPv6 26-26
editing 24-5
embedded policies 24-5
enabling 19-13
failover synchronization effect 27-7
host-name setting 24-4
local, pulling 6-16
local, pushing 6-15
lookup ID 22-17
process 24-2
processing order to determine 24-7
quality of service 19-11
RADIUS pool name, mapping 24-7
regional 6-15
troubleshooting 24-8
user class identifier, mapping 24-7
client-class-policy command (CLI)
set 24-6
setLeaseTime 24-6
setOption 24-6
setV6Option 24-6
setV6VendorOption 24-6
setVendorOption 24-6
show 24-6
client-data, DHCPv6 option B-13
client-fqdn, DHCP option B-17
client-fqdn, DHCPv6 option B-13
client-identifier, DHCP option B-10
client-identifier, DHCPv6 option B-11
client-last-transaction-time, DHCP option B-17
client-policy command (CLI)
set 24-11
setLeaseTime 24-11
setOption 24-11
setV6Option 24-11
setV6VendorOption 24-11
setVendorOption 24-11
show 24-11
clients
authentication, limiting 24-13
caching parameters 24-14
configuring 24-9
default 24-9
DHCPv6 26-4, 26-26
editing 24-11
embedded policies 24-11
failover synchronization effect 27-7
hardware address 1-2
IP address 23-16, 1-2
lease request name 19-13
listing 24-10
properties 24-10
provisioning 24-14
subscribers, set by limitation-id 24-16
unknown 24-12

Windows 2000 24-11
your IP address 23-15, 1-10
clt-time, DHCPv6 option B-13
cluster command (CLI)
create 6-3
polling attributes, setting 6-10
set 6-4
clusters
activating 6-8
data, recovering 6-8
deactivating 6-8
local 4-2
lock, overriding 2-13
polling attributes 6-10
poll-replica-interval 6-7
poll-replica-offset 6-7
poll-subnet-util-interval 9-15
poll-subnet-util-offset 9-15
poll-subnet-util-retry 9-15
regional 4-2
secure connections 6-2
time skew effects 6-10
CMTS
See cable modem termination system
CNAME records A-3, 1-2
cnr_exim utility 8-11
cnr_keygen utility 28-12
cnr_tactool utility 7-25
CNRDB database 1-2
backing up 8-7
damaged 8-7
files 8-7
log files 8-7
recovering 8-9
cnrdb_checkpoint utility 8-15
cnrdb_recover utility 8-14
cnrdb_verify utility 8-15
cnrsnmp_log file 7-8
cnrwebui_log file 7-8
.com domain 14-2
comment, DHCP expression 25-8
COMMUNICATIONS-INTERRUPTED, failover 19-10
concat, DHCP expression 25-8
concepts and protocols 1-1
config_mcd_log file 7-8
configuration
file, lease-notification 22-29
guidelines 4-5
special cases 4-6
configuring client-classes
See client-classes
configuring DHCP servers
scopes, *See* scopes
configuring DHCP servers, *See* DHCP
configuring DNS servers
loopback zones, *See* loopback, zones
NOTIFY, *See* NOTIFY
secondary, *See* secondary, name servers
configuring DNS update
See DNS update
configuring DNS update maps
See DNS update, maps
configuring failover
See failover, DHCP
configuring policies
See policies
configuring virtual private networks
See VPNs
consistency rules
adding 7-19
listing 7-20
viewing 7-19
constrained roles 5-2, 1-7
cookie-servers, DHCP option B-2
create-prefix, DHCP expression 25-8
create-prefix-addr, DHCP expression 25-8
create-prefix-range, DHCP expression 25-9
create-v6-option, DHCP expression 25-9

cron task (UNIX) 22-27

D

dashboard 3-1
DHCP address utilization chart 3-13
DHCP buffer capacity chart 3-12
DHCP DNS update activity chart 3-13
DHCP failover status chart 3-14
DHCP general indicators chart 3-15
DHCP response latency chart 3-12
DHCP server request activity 3-10
DHCP server response activity 3-11
DNS caching queries chart 3-16
DNS forwarded queries chart 3-16
DNS general indicators chart 3-19
DNS inbound zone transfers chart 3-17
DNS network errors chart 3-18
DNS outbound zone transfers chart 3-17
DNS related servers errors chart 3-18
system metrics chart 3-9
data directory, changing 8-2
Data over Cable Service Interface Specification
See DOCSIS
database subrole (ccm-admin) 5-3
database subrole (regional-admin) 5-4
databases
backup 8-1
strategies 8-2
third party programs 8-3
binding 1-1
CCM 8-2
CNRDB 8-2, 1-2
DNS update change set 17-10
exporting 8-11
importing 8-11
log files 7-8
MCD 8-2, 1-5
recovering, restoring 8-5
replica 6-7
startup, loading on 7-1
dbcheck utility (backup) 8-4, 8-16
decodeRFC1035, DHCP expression 25-9
default-ip-ttl, DHCP option B-3
default-tcp-ttl, DHCP option B-5
delegated address space 9-4, 9-5
delegation-only zones 17-7
deployment cases
large enterprise network 4-3, 28-2
small to medium size LANs 4-3
dex extensions 29-9
dex.h file 29-10
DHCID records A-3
DNS update 28-3
DHCP
administration 19-2
buffers, allocating 23-4
clients
client lease request name 19-13
IP addresses 23-16, 1-2
MAC addresses 19-13, 1-2
your IP address 23-15, 1-10
client-server model 19-1
configuration guidelines 4-5
custom options 21-12
custom options, adding 21-12
dashboard
address utilization chart 3-13
buffer capacity chart 3-12
DHCP failover status chart 3-14
DNS update activity chart 3-13
general indicators chart 3-15
response latency chart 3-12
server request activity chart 3-10
server response activity chart 3-11
defer lease extensions option 23-5
equal-priority-most-available 20-13, 20-15
ethernet addresses 20-2

forwarding 23-21
last transaction time granularity 23-6
lease history collection 6-12
lease times 20-2
leasequery, *See* leasequery
lease-state updates to LDAP 24-26
library path 23-8
log settings 23-12
option 82 24-15
options 21-6
  RFC 2132 B-1
  syntax for more complex 21-7
  validation B-14
ping packets 23-5
policies, *See* policies
priority-address-allocation 20-15
related servers, displaying 7-16
related servers, getting 6-6
remote WAN configuration 20-2
request
  buffers 27-18
  processing 19-12
requests for other servers, ignoring 22-20
requests, number of 23-4
response buffers 27-18
responses, number of 23-5
reverse zones, synthesizing 28-4
sample users 19-2
scopes disabling for 20-21
servers
  configuring 23-1
  forwarding, switching 23-21
  interface, removing address 20-2
  interface, setting 20-2
  IP address of next DHCP 1-8
  logging 7-8, 28-29
  troubleshooting 23-12
SMS network discovery records 23-8
subnet utilization collection 6-11
  switching 23-21
  unicast, enabling 23-5
  v6, *See* DHCPv6
dhcp command (CLI)
  attachExtension 23-21, 29-3
  detachExtension 23-21, 29-3
  disable
    vpn-communication 23-21
  enable
    client-class 24-3
    collect-sample-counters 7-12
    defer-lease-extensions 23-5
    delete-orphaned-leases 20-18, 23-21
    delete-orphaned-subnets 23-21
    get-subnet-mask-from-policy 21-5
    hardware-unicast 23-5
    ignore-requests-for-other-servers 22-20
    ip-history 22-22
    return-client-fqdn-if-asked 28-21
    save-lease-renewal-time 24-25
    skip-client-lookup 24-13
    update-dns-for-bootp 23-3
    use-client-fqdn 28-21
    use-client-fqdn-first 28-21
    use-ldap-client-data 24-22
  get 23-6
    dbsn 8-10
  getPrefixCount 26-22
  getRelatedServers 6-6, 7-18
  getScopeCount 20-11
  getStats 7-13
  lease history collection attributes 6-12
  limitationList 7-4, 24-18
  resetStats 7-12

set 23-6
activity-summary-interval 7-12, 23-13
client-cache-count 24-14
client-cache-ttl 24-14
client-class-lookup-id 22-17, 24-16, 25-1
failover-poll-interval 27-18
failover-poll-timeout 27-18
failover-recover 27-12
last-transaction-time-granularity 23-6
ldap-mode 24-22
log-settings 7-7, 23-12, 28-19
map-radius-class 24-7
max-dhcp-requests 23-4, 27-18
max-dhcp-responses 23-5
max-dhcp-responses) 27-18
max-ping-packets 23-5
max-waiting-packets 23-13
mcd-blobs-per-bulk-read 23-14
sms-lease-interval 23-8
sms-library-path 23-8
sms-network-discovery 23-8
sms-site-code 23-8
synthesize-reverse-zone 28-6
traps-enabled 1-10
username 24-21
vpn-communication 23-19
setPartnerDown 27-17
show 23-6
start 7-4
stop 7-4
subnet utilization collection attributes 6-11
unset 23-6
updateSms 23-7, 23-9, 27-19
DHCP forwarding, enabling 23-21
dhcp-address-block command (CLI)
set
default-subnet-size 23-20
vpn 23-18
vpn-id 23-18
unset 23-20
dhcp-admin role 5-3
core functionality 5-3
ipv6-management subrole 5-3
dhcp-client-identifier, DHCP option B-10
DHCPDISCOVER packets 24-15
dhcp-dns-update command (CLI)
create 28-7
enable
update-dns-first 28-7
update-dns-for-bootp 28-7
set
backup-server-addr 28-7
backup-server-key 28-13
dynamic-dns 28-7
forward-zone-name 28-6
reverse-zone-name 28-6
reverse-zone-prefix-length 28-4
server-addr 28-7
server-key 28-13
synthesize-name 28-6
synthetic-name-stem 28-6
v6-synthetic-name-generator 28-4
dhcp-interface command (CLI) 20-2
DHCPLEASEQUERY packets
See leasequery
dhcp-lease-time, DHCP option B-8
dhcp-management subrole (central-cfg-admin) 5-4
dhcp-management subrole (cfg-admin) 5-3
dhcp-management subrole (regional-addr-admin) 5-4
dhcp-max-message-size, DHCP option B-9
dhcp-message, DHCP option B-9
dhcp-message-type, DHCP option B-8
dhcp-option-overload, DHCP option B-8

dhcp-parameter-request-list, DHCP option B-9
dhcp-rebinding-time, DHCP option B-9
DHCPRENEW packets 24-18
dhcp-renewal-time, DHCP option B-9
DHCPREQUEST packets 20-2, B-8
dhcp-requested-address, DHCP option B-8
dhcp-server-identifier, DHCP option B-9
dhcp-user-class-id, DHCP option B-17
DHCPv6
  AAAA records. 28-2
  address generation 26-3
  bindings 26-5
  client FQDN, using 28-5
  client-classes 26-26
  clients 26-26
  concepts 26-2
  DHCID records. 28-3
  DNS update
  DNS update, upgrading 28-3
  lease affinity 26-5
  leases 26-4
  links 26-2
  managing 26-1
  options 26-27
  policies 26-25
  policy hierarchy 26-7
  prefixes 26-2
  PTR records. 28-2
  reconfigure support 26-27
  server attributes
    max-client-leases 26-25
    v6-client-class-lookup-id 26-25
  synthetic names, generating 28-3
dig tool, troubleshooting DNS update 28-19
Digital Subscriber Line (DSL) 25-23, 1-3
distributions, zone 15-19
DNS
  address format 14-2
  authoritative server, *See* authoritative name servers
  cache, flushing 17-15
  caching-only servers, *See* caching-only servers
  client/server model 14-5
  configuration guidelines 4-5
  dashboard
    caching queries chart 3-16
    forwarded queries chart 3-16
    general indicators chart 3-19
    inbound zone transfers chart 3-17
    network errors chart 3-18
    outbound zone transfers chart 3-17
    related servers errors chart 3-18
  distributed database 15-1
  domains, *See* domains
  dynamic DNS update, *See* DNS update
  exception handling 17-4
  flushing cache 7-3
  forward-retry-time 17-8
  glue records 15-17, 1-4
    invalid 15-17
    prefetching 17-13
    removing 15-18
  lame delegation 15-17
  lame delegation, reporting 17-14
  log settings 17-19
  maximum
    cache TTL property 17-15
    memory cache size 17-15
  name servers 14-5
  name-to-address resolution 14-5, A-1
  negative cache time 17-14, 1-6
  options
    flushing DNS cache 7-3, 17-15
    local and external ports 17-17
    maximum memory cache size 17-15
  ports 17-17

primary servers, *See* primary name servers
recursive queries 17-8
request-expiration-time 17-8
request-retry-time 17-8
reverse name servers 14-6
root name servers, defining 17-7
secondary servers, *See* secondary, name servers
servers
commands 17-1
logging 7-8, 17-19, 28-28
managing 17-1
name 17-2
network interfaces, configuring 17-2
secondary, *See* secondary, name servers
simulating top-of-zone A records 28-28
slave servers, creating 17-3
tcp-query-retry-time 17-8
troubleshooting 17-18
dns command (CLI)
addException 17-6
addForwarder 17-4
addRootHint 17-8
disable
no-fetch-glue 17-14
subnet-sorting 17-9
enable
fake-ip-name-response 17-16
ixfr-enable 17-10
lame-deleg-notify 17-14, 17-18
notify 17-11
round-robin 17-9
save-negative-cache-entries 17-16
simulate-zone-top-dynupdate 28-28
slave-mode 17-4
subnet-sorting 17-9
findRR 16-8
flushCache 7-3, 17-16
forceXfer 7-3
get
round-robin 17-9
getRRCount 16-6
getStats 7-12
ha 18-4
getZoneCount 15-13
listExceptions 17-6
listForwarders 17-4
removecachedRR 16-5
removeException 17-6
removeForwarder 17-4
resetStats 7-12
scavenge 7-3, 28-19
set
activity-summary-interval 17-20
checkpoint-interval 17-10
exception-forwarding 17-6
local-port-num 17-17
log-settings 7-7, 17-19
max-cache-ttl 17-15
max-negcache-ttl 17-15
mem-cache-size 17-15, 17-18
notify-min-interval 17-17
notify-send-stagger 17-18
notify-wait 17-18
query-source-address 17-17
query-source-port 17-17
remote-port-num 17-17
setPartnerDown 18-4
show 17-3
DNS update
backup-server-key 28-13
benefits 19-6
change set database 17-10
checkpoint, *See* checkpointing, zone

configurations
DHCPv6 synthetic name generator 28-3
failover synchronization effect 27-7
policies, setting for 21-5
synthetic name stem 28-3
confirming records 28-17
creating 28-7
DCHPv6 28-2
definition 28-1
DHCPv6
AAAA records 28-2
DHCPID records 28-3
PTR records 28-2
synthetic names 28-3
dynamic-dns 28-7
enabling 15-19, 28-5
failover synchronization effect 27-7
forcing 28-30
large deployments 4-6, 28-2
logging 28-19
maps
creating 28-7, 28-9
dhcp-named-policy 28-9
dhcp-policy-selector 28-9
operation 19-7
policies
configuring 28-13
creating 28-14
deleting 28-16
editing 28-15
interaction with previous releases 28-13
rules 28-14
zones, applying to 28-17
policies, *See* DNS update, policies
server-key 28-13
simulating top-of-zone A records 28-28
troubleshooting 28-19
TSIG security 28-11
Windows 2000 clients 28-20
dns-admin role 5-3
core functionality 5-3
ipv6-management subrole 5-3
security-management subrole 5-3
server-management subrole 5-3
dns-management subrole (cfg-admin) 5-3
dns-servers, DHCPv6 option B-12
dns-update-map command (CLI)
create 28-9
set
dhcp-named-policy 28-9
dhcp-policy-selector 28-9
DOCSIS 4-1, 1-2
TFTP and 1-2
documentation
conventions xxx
roadmap xxxi
domain names
dot-separated 14-3
space 14-2
tree structure 14-2
domain-list, DHCPv6 option B-12
domain-name, DHCP option B-2
domain-name-servers, DHCP option B-2
domains
difference from zones 14-3
names, *See* domain names
registering 14-3
domain-search, DHCP option B-18
do-times, DHCP expression 25-9
dual zone updates 28-22
dynamic
addresses 9-1
BOOTP, *See* BOOTP, dynamic
DNS, *See* DNS update

E

edit mode
- staged 1-8
- synchronous 1-9

.edu domain 14-2
elapsed-time, DHCPv6 option B-11
encodeRFC1035, DHCP expression 25-9
end, DHCP option B-2
enterprise users 4-1
environment dictionary, extension point 29-22
- data items 29-23
- initial-environment-dictionary property 29-25

environment-destructor extension point, DHCP 23-10, 29-42
environmentdictionary, DHCP expression 25-10
equal(i), DHCP expression 25-10
error, DHCP expression 25-10
event logging 7-7
Event Viewer, Windows 7-7
exception
- *See* resolution exception

export command (CLI)
- addresses, VPN 23-19
- leases 22-6
  - -vpn 23-19
- option-set 21-15

expressions 25-1
- address range for scope templates 20-7
- creating 25-3
- debugging 25-24
- embedded policy for scope templates 20-7
- link templates 26-11
- prefix templates 26-17
- scope templates 20-4
  - scope name 20-6

extension command (CLI)
- create 29-3
- list 29-3
- set
  - init-entry 29-3

extension points, DHCP 23-9
- listing 29-2

extensions 23-9
- API C-22
- C/C++ 29-4, 29-9
- check-lease-acceptable 29-36
- client-classes
  - modifying 29-15
  - processing 29-15
- configuration errors 29-6
- creating 29-2
- deciding approaches 29-4
- decoded packet data items C-1
- definition 29-1
- determining tasks 29-3
- DEX 29-9
  - environment methods C-35
- DEX attributes
  - dictionary C-27
  - methods C-27
- dex.h file 29-10
- dhcp-parameter-request-list option 29-27
- DHCPv6, enabling 29-14
- dictionaries 29-5, 29-22
  - entries C-1
- DNS requests, processing 29-21
- environment dictionary 29-5, 29-22
- environment-destructor 29-42
- failover synchronization effect 27-7
- init-entry 29-28
  - C/C++ 29-12
  - Tcl 29-9
- initial-environment-dictionary property 29-25

leases
acceptability, determining 29-17
finding 29-16
requests, serializing 29-17
state changes, tracing 29-21
lease-state-change 29-37
networks and links, determining 29-16
packet data items, decoded 29-26
packets
decoding 29-14
receiving 29-14
post-class-lookup 29-30
post-client-lookup 29-34
post-packet-decode 29-29
post-packet-encode 29-41
post-send-packet 29-42
pre-client-lookup 29-30
pre-packet-decode 29-28
pre-packet-encode 29-39
recognizing 29-7
request dictionary 29-5, 29-25, C-9
request processing 29-12
response containers, building 29-16
response dictionary 29-5, 29-25, C-15
response packets
encoding 29-20
gathering data 29-20
sending 29-21
routine signature 29-5
stable storage, updating 29-21
Tcl 29-4, 29-7
environment methods C-25
Tcl attributes
dictionary C-23
methods C-23
thread-safe 29-10
extensions-path, DHCP option B-3
external port 17-17

F

failover, DHCP
address ranges, ensuring 27-3
attributes
advanced 27-12
affected 27-7
related failover 6-4
related servers 6-5
back office scenarios 27-2
backup
allocation boundary 20-15
percentage 19-11
server, removing 27-21
backup percentage 27-13
benefits 19-8
BOOTP
clients 27-22
relay 27-4
changing roles 27-20
checklist 27-3
configuring 27-1
confirming 27-9
creating server pairs 5-33
dynamic BOOTP backup percentage 27-14
lazy updates 27-15
LDAP 27-4
lease period factor 27-15
lease query 27-23
load balancing 27-19
compatibility with earlier release 27-19
configuring 27-19
local server synchronization 27-4
logging 27-8, 27-23
main server, adding new 27-21
maximum client lead time 27-15
monitoring 27-23
multiple interfaces 27-22
network discovery 27-19

network failures 27-24
operation 19-8
halting 27-21
pairs
address matches 27-5
backup percentage 27-13
creating 27-5
synchronizing 27-8
partner down state 27-16
polling
attributes 27-18
priority 6-11
polling attributes, setting 6-10
regional cluster synchronization 27-6, 27-9
regional pairs 6-17
related servers, getting 6-6
removing backup server 27-21
replacing servers with defective storage 27-21
request/response buffer settings 27-18
restrictions, communications interrupted 19-10
safe period 27-16
scenarios 27-1
server pairs, creating 27-4
simple scenarios 27-1
state transitions 27-9
states 19-9
symmetrical scenarios 27-3
synchronize function 27-6
synchronizing 27-7
synchronizing pairs 5-33
troubleshooting 27-23
types 19-8, 27-1
failover-pair command (CLI)
addMatch 27-5
create 27-5
enable
load-balancing 27-13
use-safe-period 27-17
listMatches 27-5
polling attributes, setting 6-10
removeMatch 27-5
set
backup-pct 27-13
dynamic-bootp-backup-pct 27-23
load-balancing 27-19
safe-period 27-17
sync 27-8
file_tftp_1_log file 7-8, 7-24
file_tftp_1_trace file 7-8
filter, lease 22-9
finger-servers, DHCP option B-8
font-servers, DHCP option B-7
force-lock command (CLI) 2-13
forwarding DNS servers 17-3
listing 17-4
forwarding servers
DHCP 23-21
FQDN 14-2, 16-3, 1-3
DHCP processing 19-12
option, DHCP 28-20
free-address-low-threshold event, SNMP 1-6

G

gateway address 19-14, 23-15, 1-4
generate-lease extension point, DHCP 23-10
geo-conf, DHCP option B-18
geoconf-civic, DHCP option B-17
geoconf-civic, DHCPv6 option B-13
giaddr, DHCP field 19-14, 23-15, 1-4
glue records 15-17, 1-4
invalid 15-17
prefetching 17-13
removing 15-18
.gov domain 14-2
grace period, leases 21-4
grep tool (UNIX) 7-25

group command (CLI)
create 5-6
groups 5-1, 5-4
adding 5-6
interaction with roles 5-1
pulling 5-12
pushing 5-11

H

HA DNS
backup server, setting 18-2
configuring 18-1
enabling 18-2
main server, setting 18-2
partner down, setting 18-4
server pairs, creating 18-4
synchronizing server pairs 18-4
ha-dns-pair command (CLI)
create 18-4
enable
ha-dns 18-4
set
ha-dns-backup-server 18-2
ha-dns-main-server 18-2
sync 18-4
Help
attributes 2-6
pages 2-7
HINFO records A-3, 1-4
Host Info records
See HINFO records
host-admin role 5-4
core functionality 5-4
hostmaster, setting for zone 15-7
host-name, DHCP option B-2
host-regexpr syntax 5-22
hosts
adding to zones 10-1, 16-11
BOOTP 23-1
creating 5-21
dynamic 15-19, 28-1
editing 10-3
multiple interface 27-22
pinging 22-7
removing 10-3
testing address ranges 5-24
zone restrictions 5-22
HTTPS login 2-2

I

iaaddr, DHCPv6 option B-11
ia-na, DHCPv6 option B-11
ia-pd, DHCPv6 option B-12
iaprefix, DHCPv6 option B-12
ia-ta, DHCPv6 option B-11
ICMP
echo, *See* PING
errors, ignoring 22-7
ieee802.3-encapsulation, DHCP option B-5
IETF 19-1, 20-23, 1-3
if, DHCP expression 25-11
ifconfig tool (UNIX) 7-25
import command (CLI) 15-9
keys 28-13
leases 22-6, 23-19
option-set 21-15
impress-servers, DHCP option B-2
in-addr.arpa domain 14-6, 1-4
incremental zone transfers 1-4
enabling 4-5, 17-9
info-refresh-time, DHCPv6 option B-12
init-entry extension point
C/C++ 29-12
Tcl 29-9

init-entry extension point, DHCP 23-10, 29-28
initial-url, DHCP option B-17
INIT-REBOOT packets 19-10
install_cnr_log file 7-8
interface cards 20-2, 28-28
interface-id, DHCPv6 option B-12
interface-mtu, DHCP option B-4
Internet Control Message Protocol
See ICMP
Internet Engineering Task Force 19-1, 1-3
Internet Operating System
See IOS
Internet Service Providers 4-1
interoperability of releases 4-6
Intranet Builder 17-4
IOS, VPN support 23-14
IP addresses
See addresses, IP
IP helper address 23-3
IP history
See lease history
ip6-string, DHCP expression 25-11
ipconfig utility 20-24, 24-12
ip-forwarding, DHCP option B-3
ip-helper 27-23
adding to router 5-29
iphist utility 22-24
ip-node license 5-8
ip-string, DHCP expression 25-11
IPv6 addresses 26-1
ipv6-management subrole (addrblock-admin) 5-3
ipv6-management subrole (dhcp-admin) 5-3
ipv6-management subrole (dns-admin) 5-3
irc-servers, DHCP option B-8
ISDN records A-3
iSNS, DHCP option B-10
ISPs
See Internet Service Providers
is-string, DHCP expression 25-11

J

jsui_log.*date*.txt file 7-8

K

key command (CLI)
create 28-12
keybuild utility (backup) 8-4, 8-15
keys, generating secrets 28-12

L

lame delegation 15-17, 1-4
reporting 17-14
LAN segments 19-12, 1-8
large enterprise deployments 4-3
lazy updates, failover 27-15
LDAP 16-10
client
configuration 24-20
data use, enabling 24-22
configuring 24-19
connections 24-31
DHCP
client queries 24-21
mapping 24-21
directory support 24-21
distinguished name (dn) 24-21
embedded policies 24-23
entry creation 24-29
enabling 24-30
event service, failover synchronization effect 27-7
failover configuration 27-4
filtering searches 24-28
lease-state attributes 24-24
passwords 24-21
protocol definition 24-19
queries, enabling 24-21

query-timeout 24-31
schema checking, disabling 24-20
state updates 24-27
storing lease data 24-26
threadwaittime 24-31
timeout 24-31
troubleshooting 24-30
typical attribute settings 24-31
unprovisioning client entries 24-23
updates, enabling 24-28
ldap command (CLI)
create 24-21
delete 24-21
enable
can-query 24-8, 24-22
can-update 24-28
getEntry 24-22
list 24-23
listnames 24-23
set
create-object-classes 24-30
dn-attribute 24-29
dn-create-format 24-29
dn-format 24-29
password 24-21
preference 24-22
search-filter 24-21
search-path 24-21
search-scope 24-21
update-search-attribute 24-28
update-search-filter 24-28
update-search-path 24-28
update-search-scope 24-28
username 24-28
setEntry 24-22
create-dictionary givenname 24-30
create-dictionary localityname 24-30
create-dictionary sn 24-30
update-dictionary carlicense 24-28
update-dictionary uid 24-28
show 24-23
unsetEntry 24-22
ldap-url, DHCP option B-17
lease command (CLI)
activate 22-8
deactivate 22-8
delete-reservation 22-16
force-available 22-18, 23-3
list
-macaddr 22-10
send-reservation 22-13
set
address 24-25
client-dns-name 24-25
client-domain-name 24-25
client-flags 24-25
client-host-name 24-25
client-id 24-25
client-mac-addr 24-25
expiration 24-25
flags 24-25
lease-renewal-time 24-25
start-time-of-state 24-25
state 24-25
vendor-class-identifier 24-25
show 22-3
lease extensions, deferring 23-6
lease history 22-21
automatic trimming
age 22-26
interval 22-26
collecting 22-22
collection maximum age 6-12
database directory 22-22
detail 22-23
enabling 6-12, 22-22
failover server priority 6-11
iphist utility 22-24

maximum age for trimming 22-27
polling
data 6-9
interval 6-10
offset 6-11
retry interval 6-11
querying 22-23
recording 9-14, 22-22
reports 22-21
trimming 22-26
lease period factor, failover 27-15
lease6 command (CLI)
activate 26-22
deactivate 26-22
force-available 26-22
list 22-11
lease-history subrole (regional-addr-admin) 5-4
lease-notification command (CLI) 22-27
available 22-21
mail-host 22-27
recipients 22-27
scopes 22-27
specifying config file 22-29
leasequery 22-29
DHCPv4 pre-RFC implementation 22-30
DHCPv4 RFC 4388 implementation 22-31
DHCPv6 implementation 22-31
implementations 22-29
logging 23-13
reservations and 22-30
statistics 22-32
leases 19-3
activity 7-18
address usage reports 22-21
affinity 26-5
benefits 19-3
database 7-8
deactivating 23-3
in scopes 22-8
deferring extensions 23-5
defined 19-2
DHCPv6 26-4
DHCPv6 life cycle 26-6
displaying 7-18, 22-27
excluding addresses from ranges 22-8
expired state 19-2
exporting 22-5
file 22-6
time format 22-6
forcing available 22-18
grace period 21-4
history reports 22-21
importing 22-5
inhibiting renewals 22-18
LDAP attributes 24-27
notification, receiving 22-27
obtaining 19-7
one-shot 24-12
orphaned 20-18
by VPN 23-21
permanent 21-3, 22-5
pinging before allocating 22-7
provisional 24-12
querying, *See* leasequery
reacquiring 19-8
reactivating 22-8
recommended renewal times 4-6
releasing 19-7
renewal time, saving as state 24-25
renewing 19-2
reservations, *See* reservations, lease
reusing 23-3
scopes 19-2, 22-1
listing 22-3
viewing 22-1
searching, filtering 22-9
state updates in LDAP 24-26
states 22-4, 24-25

timeouts for unavailable 22-20
times
guidelines 22-4
in import files 22-7
overrides, allowing 22-5
types 19-3
unavailable
clearing 22-18
handling 22-19
utilization 22-27
lease-state-change extension point, DHCP 23-10, 29-37
length, DHCP expression 25-11
let, DHCP expression 25-11
license 5-8
license command (CLI)
create 5-8
list 5-8
listnames 5-8
show 5-8
licenses 5-8
adding 2-3
link command (CLI)
applyTemplate 26-14
create 26-14
template, with 26-14
template-root-prefix, with 26-14
listPrefixes 26-14
listPrefixNames 26-14
link templates
expressions 26-11
links
configuring 26-9
DHCPv6 26-2
link-template command (CLI)
apply-to (link) 26-11
create 26-10
clone 26-11
Linux
CLI location 2-12
list, DHCP expression 25-12
local clusters 1-1, 4-2
administration 5-1
connecting to 6-6
editing 6-3
replicating data 6-7
synchronizing with 6-7
tutorial 5-15
view of tree 6-2
local port 17-17
localhost 15-5, 17-18
localhost_access_log.*date*.txt file 7-8
lock files 8-3
log, DHCP expression 25-12
log.xxx files, CNRDB 8-7
logging
NOTIFY, *See* NOTIFY
logging out 2-7
login, Web UI 2-2
log-servers, DHCP option B-2
loopback
addresses 1-5
zones 15-5, 1-5
lpr-servers, DHCP option B-2
lq-client-links, DHCPv6 option B-13
lq-query, DHCPv6 option B-13
lq-relay-data, DHCPv6 option B-13

M

MAC addresses, clients 19-13
Main Menu page
local
advanced mode 2-9
basic mode 2-8
regional 2-12
management components 1-1
mapcharacters, DHCP expression 25-12
mask-blob, DHCP expression 25-12

mask-int, DHCP expression 25-12
mask-supplier, DHCP option B-4
max-dgram-reassembly, DHCP option B-3
maximum client lead time, failover 19-10, 27-15
MB records A-3
MCD database 1-5
  backing up 8-4
  files 8-6
  integrity 8-5
  recovering
    backup 8-5
    damaged 8-5
mcdadmin utility 8-12
mcdshadow backup utility 8-3, 1-5
  third party programs and 8-3
mcdupgrade_log file 7-8
MCLT
  See maximum client lead time
mcns-security-server, DHCP option B-18
merit-dump, DHCP option B-2
MG records A-4
Microsoft DHCP stack 20-2
.mil domain 14-2
MINFO records A-4
mobile-ip-home-agents, DHCP option B-16
MR records A-4
MSOs 4-1
multinetting 19-12, 1-5
multiple interfaces, failover 27-22
Multiple Service Operators
  See MSOs
multithreaded server 1-2
MX records A-4, 1-5

N

Name Server records
  See NS records
name servers
  authoritative, *See* authoritative name servers
  domain 14-5
  primary, *See* primary name servers
  reverse 14-6
  secondary, *See* secondary, name servers
  types 14-5
name_dhcp_1_log file 7-8
name_dns_1_log file 7-8
name-servers, DHCP option B-2
name-service-search, DHCP option B-18
name-to-address resolution 14-5, A-1
Naming Authority Pointer records
  See NAPTR records
NAPTR records 16-9, A-5
navigation bar labels xxx
nds-context, DHCP option B-17
nds-servers, DHCP option B-17
nds-tree, DHCP option B-17
negative cache time 17-14, 1-6
netbios-dd-servers, DHCP option B-6
netbios-name-servers, DHCP option B-6
netbios-node-type, DHCP option B-7
netbios-scope, DHCP option B-7
netinfo-parent-server-addr, DHCP option B-17
netinfo-parent-server-tag, DHCP option B-17
netwareip-domain, DHCP option B-16
netwareip-information, DHCP option B-16
Network Information Service
  See nis entries
network interfaces
  DNS server 17-2
Network News Transport Protocol B-7
network number 14-3
Network Time Protocol B-6
network time service, use of 6-10

networks
editing names 20-24
listing 20-24
managing 20-24
nis+-domain, DHCP option B-7
nis+-servers, DHCP option B-7
nis-domain, DHCP option B-5
nis-domain-name, DHCPv6 option B-12
nisp-domain-name, DHCPv6 option B-12
nisp-servers, DHCPv6 option B-12
nis-servers, DHCP option B-5
nis-servers, DHCPv6 option B-12
nntp-servers, DHCP option B-7
non-local-source-routing, DHCP option B-3
nonsecure login 2-2
NORMAL state, failover 19-9
not, DHCP expression 25-13
notification, lease 22-27
NOTIFY 1-6
enabling 17-11
logging transactions 17-19
NS records A-5
NSAP records A-6
nslookup utility 17-20
troubleshooting DNS update 28-19
ntp-servers, DHCP option B-6
null, DHCP expression 25-13

O

on-demand address pools
See subnet allocation, DHCP
option command (CLI)
create 21-14
get 21-11
listtypes 21-14
show 21-11
unset 21-11
option sets
exporting 21-15
importing 21-15
local
pulling 21-16
pushing 21-16
options
custom DHCP 21-12
data types, listing 21-14
DHCPv6 26-8
setting 26-27
failover synchronization effect 27-7
policy hierarchy 21-3
option-set command (CLI)
create 21-14
list/listnames 21-14
show 21-11
OptionSetJumpStart.txt file 21-16
OptionSetPXE.txt file 21-16
or, DHCP expression 25-13
organization, registering 13-6
Organizationally Unique Identifier
See OUI
oro, DHCPv6 option B-11
OUI, for VPNs 23-17, 1-6
owner command (CLI)
create 12-2
owner-region subrole (ccm-admin) 5-3
owner-region subrole (regional-admin) 5-4
owners
configuring 12-1
managing 12-1
pulling 5-15
pushing 5-14

P

pad, DHCP option B-2
PARTNER-DOWN state, failover 19-10
  enabling 27-16
passwords 5-7
  administrator 5-5
    changing 5-7
  changing 2-3
  nondisplaying 5-7
path-mtu-aging-timeout, DHCP option B-3
path-mtu-plateau-tables, DHCP option B-3
PAUSED state, failover 19-11
PDU, SNMP 1-7
performance guidelines 4-5
perform-mask-discovery, DHCP option B-4
persistent cache 17-8
pick-first-value, DHCP expression 25-13
ping utility 22-7
pinging hosts before offering 22-7
point of contact, registering 13-4
Pointer (Reverse Mapping) record
  See PTR record
policies
  cloning 21-6
  compared with scopes 19-3
  configuring 21-1
  creating regional 6-13
  defined 19-3, 21-1, 1-6
  DHCPv6 26-25
  dual zone updates, allowing 28-22
  embedded 21-2
    editing 21-8
    scopes 20-19
    vendor options 20-19
  failover synchronization effect 27-7
  guidelines 20-2
  hierarchy 21-3
  lease time overrides, allowing 22-5
  local
    pulling 6-14
    pushing 6-14
  named 21-2
  options 19-13
    adding 21-6
  scopes 19-12
  scopes, *See* scopes
  subnet-masks, setting 21-5
  suboptions
    adding 21-7
  system-default 21-1
policies, DHCP
  address block 23-20
  pushing to local clusters 5-31
policy command (CLI)
  create 21-5, 23-2
    clone 21-6
  disable
    allow-client-a-record-update 28-20
  enable
    allow-client-a-record-update 28-20, 28-21
    allow-dual-zone-dns-update 28-22
    allow-lease-time-override 22-5
    permanent-leases 21-5
    use-client-id-for-reservations 22-17
  getOption 21-7
    dhcp-lease-time 21-5
  listOptions 21-5
  set 21-5, 23-2
    grace-period 24-12
    limitation-count 25-2
  setLeaseTime 21-5
  setOption 21-7, 23-3
    subnet-mask 21-5
  unsetOption 21-7
policy-filters, DHCP option B-3

polling 1-6
failover server priority 6-11
interval 6-10
lease history data 6-9
offset 6-11
retry interval 6-11
subnet utilization data 6-9
time skew effects 6-10
pop3-servers, DHCP option B-7
Post Office Protocol B-7
post-class-lookup extension point, DHCP 23-10, 29-30
post-client-lookup extension point, DHCP 23-10, 29-34
post-packet-decode extension point, DHCP 23-10, 29-29
post-packet-encode extension point, DHCP 23-10, 29-41
post-send-packet extension point, DHCP 23-10, 29-42
POTENTIAL-CONFLICT state, failover 19-10
pre-client-lookup extension point, DHCP 23-10, 29-30
pre-dns-add-forward extension point, DHCP 23-10
preference, DHCPv6 option B-11
preferences, user 2-10
prefix command (CLI)
addReservation 26-22
applyTemplate 26-21
create 26-21
template, with 26-21
createReverseZone 28-4
deleteReverseZone 28-4
listLeases 26-22
set
selection-tags 24-4
prefix templates
expressions 26-17
prefixes
configuring 26-14
count on server, getting 26-22
DHCPv6 26-2
reverse zones, creating 28-4
interface-identifier, assigning 26-4
prefix-template command (CLI)
apply-to (prefix) 26-17
create 26-17
clone 26-17
set
selection-tags 24-4
pre-packet-decode extension point, DHCP 23-10, 29-28
pre-packet-encode extension point, DHCP 23-10, 29-39
primary name servers 14-4
configuring 15-5
defined 14-5
SOA property A-7
zone, setting for 15-7
progrn, DHCP expression 25-13
Protocol Data Unit, SNMP
See PDU
PTR records A-6, 1-7
DNS update 28-2
PXE clients 21-16
PXE clients, importing option sets for 21-15
pxe-client-arch, DHCP option B-17
pxe-client-machine-id, DHCP option B-17
pxe-client-network-id, DHCP option B-17

R

rapid-commit, DHCPv6 option B-11
reconfigure support, DHCPv6 26-27
reconfigure-accept, DHCPv6 option B-12
reconfigure-message, DHCPv6 option B-12
RECOVER state, failover 19-10
RECOVER-DONE state, failover 19-10
recursive queries 1-7
caching-only server 17-6
enabling 17-8
region command (CLI)
create 12-2

regional clusters 1-1, 4-2
adding
local clusters 5-27, 6-2
server clusters 6-2
administration 5-1
client-classes
pulling 6-16
pushing 6-15
failover pairs 6-17
policies 6-13
pulling 6-14
pushing 6-14
pushing
option definition sets 21-16
reservations
pushing 6-18
scope templates 6-12
pushing 6-12
subnets to local clusters
pushing 9-8
subnets to routers
pushing 9-8
tutorial 5-25
VPNs 6-16
regional-addr-admin role 5-4
core functionality 5-4
dhcp-management subrole 5-4
lease-history subrole 5-4
subnet-utilization subrole 5-4
regional-admin role 5-4
authentication subrole 5-4
authorization subrole 5-4
core functionality 5-4
database subrole 5-4
owner-region subrole 5-4
regions
configuring 12-1
managing 12-2
pulling 5-15
pushing 5-14
regular expressions (regex) 5-22
related servers
DHCP attributes 6-4
states 6-4
relative domain names 16-3
relay-agent-info, DHCP option B-17
relay-agent-subscriber-id, DHCPv6 option B-13
relay-message, DHCPv6 option B-11
remote-dns command (CLI)
create 17-18
remote-id, DHCPv6 option B-13
removing
cached RRs 7-3
replica data 6-7
replica data, viewing 6-7
report command (CLI) 7-16, 22-21
reports
address usage 7-15, 22-21
allocation 13-1
ARIN 13-1
IPv4 utilization 13-7
lease history 22-21
managing 13-1
organization
creating 13-5
editing 13-6
point of contact 13-2
editing 13-4
subnet utilization 9-14
WHOIS/SWIP 13-8
request dictionary, extension points 29-25
request dump, DHCP expression 25-16
request option, DHCP expression 25-13
request packetfield, DHCP expression 25-15

reservation command (CLI)
create 22-13
delete 22-16
list 22-13
set 22-13
reservations, lease
creating 22-11
LEASEQUERY and 22-30
listing 22-13
properties, setting 22-13
pushing to local clusters 6-18
removing 22-16
using client ID 22-17
resolution exception 17-4
listing 17-6
resource records
A A-1, 1-1
A6 A-2
AAAA 28-2, A-2
adding 16-2
adding in CLI 16-3
AFSDB A-2
cached, removing 7-3
CNAME A-3, 1-2
configuring 16-1
count on server, getting 16-6
DHCID 28-3, A-3
editing 16-1
subzone information 15-18
filtering 16-8
HINFO A-3, 1-4
ISDN A-3
listing 16-5
MB A-3
MG A-4
MINFO A-4
MR A-4
MX A-4, 1-5
NAPTR A-5
NS A-5
NSAP A-6
protecting 16-3
PTR A-6, 1-7
removing 16-4
cached 16-5
RP A-6
RT A-6
SOA A-7, 1-8
SRV A-7
TXT A-7
types A-1
WKS A-8, 1-10
resource-location-servers, DHCP option B-2
response dictionary, extension points 29-25
reverse
domains 14-6
mapping records A-6
name servers 14-6
zones 14-6, 1-7
configuring 15-12
prefixes, creating from 28-4
secondary 15-15
RFCs
1001, 1002 B-6
1035 15-9, 28-26
1042 B-5
1123 1-2
1179 B-2
1191 B-3
1256 B-4
1350 1-2
1497 B-1
1782 1-2
1783 1-2
1995 4-5, 17-9
1996 4-5, 17-11
2131 19-8, 23-2
2132 B-1

2136 15-19
2308 17-14
2316 1-3
2685 23-17
2782 16-9, 28-24, A-7
2915 A-5
2916 16-9
3046 24-15, 25-1
3074 27-19
3263 16-9
3315 B-10
3319 B-12
3403 16-9
3898 B-12
4075 B-12
4174 B-10
4242 B-12
4280 B-10, B-12
4291 26-3
4388 22-31, B-10
4580 B-13
4649 B-13
4701 28-2, A-3
4702 28-2
4704 28-2
4833 B-14
5007 22-31
865 B-2
868 B-2
887 B-2
893 B-4
894 B-5
950 B-2

RIC server
See Router Interface Configuration server
ric_server_log file 7-8
ric-management subrole (addrblock-admin) 5-3
ric-management subrole (central-cfg-admin) 5-4
ric-management subrole (cfg-admin) 5-3
role command (CLI)
create 5-7
roles 5-1
adding 5-6
address block administrator 9-1
base 5-1
constrained 5-2, 1-7
creating 5-21
groups 5-4
interaction with groups 5-1
pulling 5-13
pushing 5-13
subroles 5-2
root name servers 14-3, 17-3, 1-8
adding 17-7
slave servers 1-8
root-path, DHCP option B-3
round-robin 1-8
enabling 17-8
scope selection 19-12, 20-11
routed bridge encapsulation (RBE) 1-8
router command (CLI)
create 11-2
set 11-4
Router Interface Configuration (RIC) server 4-2
logging and tracing 7-8
router interfaces
adding 5-28
editing 11-4
editing attributes 11-4
viewing 11-4
router-discovery, DHCP option B-4
router-interface command (CLI)
set 11-4

routers
adding 5-28, 11-2
alternative login methods 11-3
BOOTP relay 23-3
bundling 11-5
Cisco 23-3
creating 11-2
editing 11-3
editing attributes 11-4
gateway addresses 19-14, 23-15, 1-4
ip-helper 5-29
listing 11-2
managed 11-2
policies for 19-3
resynchronizing 11-4
secure communication mode 11-3
subnet 21-6
subnets, pushing 9-8
uBR7200 5-28
virtual 11-2
routers, DHCP option B-2
router-solicitation-address, DHCP option B-4
RP records A-6
RT records A-6

S

safe period, failover 19-10, 27-16
scope command (CLI)
addRange 20-11
addReservation 22-13
changeMask 20-17
change-subnet 20-17
clearUnavailable 22-18
create 20-11
template= 20-8
delete 20-23
disable 20-17
dhcp 20-21, 23-3, 27-23
ping-clients 22-19
enable 20-17
bootp 20-21
deactivated 20-21
dhcp 20-21
dynamic-bootp 20-21, 23-3, 27-22
ignore-declines 22-20
renew-only 20-22
update-dns-for-bootp 23-3
get 20-17
list 20-17
listLeases 22-3
listnames 20-17
listRanges 22-9
listReservations 22-13
removeRange 22-9
removeReservation 22-16, 23-3
report-staged-edits 20-18
set 20-17
backup-pct 27-13
dynamic-dns 28-23
free-address-config 20-23
policy 20-11
primary-subnet 20-20
selection-tags 24-4
vpn 23-19
vpn-id 20-11, 23-19
show 20-17
unset
primary-subnet 20-20
scope templates
address range expressions 20-7
cloning 20-9
creating 20-4
creating on regional cluster 5-33
editing 20-8
attributes 20-8

embedded policy expressions 5-33, 20-7
expressions 20-4
managing 20-3
name expression 5-33
pulling from local clusters 6-13
pushing to local clusters 6-12
range expressions 5-33
regional 6-12
scope name expressions 20-6
scope-policy command (CLI)
disable 20-19
enable 20-19
listVendorOptions 20-19
set 20-19
setVendorOptions 20-19
show 20-19
unset 20-19
unsetVendorOptions 20-19
scopes
address
allocation 20-11
ranges 20-9
allocate-first-available 20-12, 20-15
allocation-priority 20-13, 20-14
attributes
bootp 20-20
disabling 20-17
dynamic bootp 20-20
enabling 20-17
getting 20-17
listing 20-17
primary-subnet 20-20
setting 20-17
showing 20-17
BOOTP, enabling 20-20
compared to policies 19-3
count on server, getting 20-11
creating 20-9
deactivating 20-21
defined 19-3
defining 20-3
DHCP, disabling 20-21
edit modes 20-18
editing 20-17
failover
backup percentage 27-13
synchronization effect 27-7
failover-backup-allocation-boundary 20-15, 20-16
forcing lease availability 22-18
free address SNMP traps 20-22
inhibiting lease renewals 22-18
internal address allocation 20-16
leases, *See* leases
moving/decommissioning BOOTP clients 20-20
multiple 20-11
named policies 21-2
names 20-9
netmask, changing 20-17
network addresses 20-9
pinging clients 22-7
policies, *See* policies
primary subnets 20-20
ranges
adding 20-11
listing 22-9
removing 22-9
removing 20-23
if not reusing addresses 20-23
if reusing addresses 20-24
ranges 22-9
reusing addresses 20-24
renew-only 20-21
secondary 20-19
staged edit mode 1-8
staged edits, reporting 20-18
subnet masks, removing 21-5
subnet, changing 20-17
synchronous edit mode 1-9

traps, SNMP, free address 20-22
unreserving leases 22-16
VPNs 23-17
scope-template command (CLI)
apply-to scope 20-8
create 20-4
clone 20-9
set 20-8
options-exp 20-6
ranges-exp 20-6
scope-name 20-6
selection-tag-list 24-4
SCP
See System Configuration Protocol
screen displays xxx
scripts
See extensions
search, DHCP expression 25-16
search, lease 22-9
secondary
expiration time 17-13
master 1-8
name servers, DNS 14-4, 1-8
adding 15-14
defined 14-5
separating DHCP servers from 20-2
refresh time 15-15, 17-12
retry time 17-13
reverse zones 15-15
adding 15-15
scopes 20-19
subnets 20-19, 1-8
zones 15-14
adding 15-14
secondary name servers 15-14
secure
cluster connections 6-2
login 2-2
security-management subrole (central-dns-admin) 5-4
security-management subrole (dns-admin) 5-3
selection tags
appending dhcp-user-class-id 24-8
appending RADIUS class 24-8
appending RADIUS pool 24-8
mapping RADIUS class 24-7
mapping RADIUS pool name 24-7
mapping user class identifier 24-8
server agent
stopping, restarting 8-6
server clusters, adding 6-2
server command (CLI)
enable/disable start-on-reboot 7-1
getHealth 7-11
getStats 7-12
reload 7-4
serverLogs
set logsize 7-8
show 7-8
set 7-4
start 7-4
stop 7-4
server-identifier, DHCPv6 option B-11
server-management subrole (ccm-admin) 5-3
server-management subrole (central-dns-admin) 5-4
server-management subrole (dns-admin) 5-3
servers
controlling 7-1
events, logging 7-6
failures, troubleshooting 7-22
health, displaying 7-10
IP address 1-8
related, states 6-4
state, displaying 7-10
statistics, showing 7-11
server-unicast, DHCPv6 option B-11

session command (CLI)
assert 8-10
dhcp.dbsn 8-10
locked 8-10
get
scope-edit-mode 20-18
set
current-vpn 20-11, 23-17
default-format 8-10
scope-edit-mode 20-18
zone-edit-mode 15-2
Session Initiation Protocol (SIP) proxies 16-9
setq, DHCP expression 25-16
setting
custom DHCP options 21-12
negative cache time 1-6
sha256, DHCP expression 25-16
shadow backups 8-1
for MCDDB recovery 8-5
manual 8-3
third party backup programs and 8-3
time, setting 8-3
SHUTDOWN state, failover 19-11
siaddr, DHCP field 23-2, 1-8
Simple Mail Transport Protocol
See SNMP
simulating top-of-zone A records 28-28
single sign-on 2-4, 6-2
sip-servers, DHCP option B-18
sip-servers-address, DHCPv6 option B-12
slave servers 17-3, 1-8
slp-directory-agent, DHCP option B-17
slp-service-scope, DHCP option B-17
SMS
failover on UNIX 27-19
integrating with DHCP server 23-7
network discovery (dhcp command) 23-8
smtp-servers, DHCP option B-7
SNMP
free-address-low-threshold 1-6
logging and tracing 7-8
notification 1-3
notification events 1-7
PDU 1-7
traps 1-6
failover mismatch 27-24
free address on scopes 20-22
PDUs 1-3
server not responding 27-24
v2c standard 1-3
snmp command (CLI)
disable server-active 1-6
enable server-active 1-5
set
cache-ttl 1-5
community 1-5
trap-source-addr 1-5
snmp-interface command (CLI) 1-6
snmp-management subrole (cfg-admin) 5-3
sntp-servers, DHCPv6 option B-12
SOA records A-7, 1-8
defined 15-7
secondary
expiration time 17-13
refresh time 17-12
retry time 17-13
TTL property 17-12
Windows 2000 28-24
Solaris
CLI location 2-12
guidelines 4-5
mouse buttons xxx
SRV records A-7
enabling viewing 28-26
Windows 2000 28-23
SSH connections, routers 11-3

SSL
cluster connections 6-2
secure login 2-2
SSO login 2-4
staged
edit mode 1-8
scope edit mode 20-18
zone edit mode 15-1
starts-with, DHCP expression 25-17
STARTUP state, failover 19-9
static
addresses 20-2
BOOTP 27-22
configuration for W2K 28-25
routes B-4
static-routes, DHCP option B-15
statistics
leasequery 22-32
server 7-11
status-code, DHCPv6 option B-11
streettalk-directory-assistance-servers, DHCP option B-8
streettalk-servers, DHCP option B-8
subnet allocation, DHCP 23-14, 1-9
configuring 23-19
subnet utilization
collect-addr-util-duration 9-14
collect-addr-util-interval 9-14
collection duration 6-11
data, collecting 9-15
enabling collection 6-11
failover server priority 6-11
polling
data 6-9
interval 6-11
offset 6-11
retry interval 6-11
poll-subnet-util-interval 9-15
poll-subnet-util-offset 9-15
poll-subnet-util-retry 9-15
querying 9-15
reports 9-14
server polling interval 6-11
trimming data 9-16
trim-subnet-util-age 9-17
trim-subnet-util-interval 9-17
viewing data 9-17
subnet-alloc, DHCP option B-18
subnet-mask, DHCP option B-2
subnets 9-4
adding 5-18
address ranges 9-11
child 9-10
client-accessible 19-13
creating 5-31
defined 19-6
editing 9-8
joining 20-19
local B-4
multiple logical 20-19
pushing to local clusters 9-8
reclaiming 9-10
sorting, enabling 17-9
viewing 9-4
subnets, DHCP
address block 23-20
allocation request 23-20
increment 23-20
initial 23-20
subnet-selection, DHCP option B-18
subnet-utilization subrole (regional-addr-admin) 5-4
subroles 5-2
central administration management 5-8
subscriber limitation, using option 82 24-14
troubleshooting 24-19
subscribers, set by limitation-id 24-16
substring, DHCP expression 25-17
subzone forwarding 17-4

subzones 1-8
adding 15-16
delegating 15-17, 1-3, 1-9
editing 15-18
lame delegation 15-17
name server 15-17
naming 15-16
removing 15-18
swap-server, DHCP option B-2
synchronization, failover
complete 27-6
exact 27-6
operations 27-7
update 27-6
synchronous
edit mode 1-9
scope edit mode 20-18
zone edit mode 15-1
System Configuration Protocol (SCP) 4-2
System Management Server
See SMS
system metrics chart, dashboard 3-9

T

TAC tool 7-25
tail command (Solaris) 7-6
tasks, scheduling 7-4
Tcl
API 29-7
extensions 23-9, 29-4, 29-7
TCP Default TTL, DHCP option B-5
tcp-keepalive-garbage, DHCP option B-5
tcp-keepalive-interval, DHCP option B-5
/temp directory 8-3
templates
scope 20-3
zone 15-2
Text records
See TXT records
TFTP 1-2, B-10
file caching 7-24
logging and tracing 7-24
packets, tracing 7-23
troubleshooting 7-23
tftp command (CLI)
enable 1-2
enable file-cache 7-24
getStats 7-14
getTraceLevel 1-2
serverLogs 1-2
set 1-2
file-cache-directory 7-24
file-cache-max-memory-size 7-24
home-directory 7-24
log-file-count 7-24
log-level 7-24
log-settings 7-7, 7-24
show 1-2
tftp-interface command (CLI) 1-3
tftp-management subrole (cfg-admin) 5-3
tftp-server, DHCP option B-10
time service, use of 6-10
Time to Live property
See TTL property
time-offset, DHCP option B-2
timeouts for unavailable leases 22-20
time-servers, DHCP option B-2
to-blob, DHCP expression 25-17
to-lower, DHCP expression 25-17
Tomcat
database log files 7-8
server 4-2
top tool (UNIX) 7-25
to-sint, DHCP expression 25-18
to-string, DHCP expression 25-18
to-uint, DHCP expression 25-18

trailer-encapsulation, DHCP option B-4
trap command (CLI)
set
free-address-low-threshold 1-6
trap-recipient command (CLI)
create 1-6
traps, SNMP 1-3, 27-24
creating 20-22
failover synchronization effect 27-7
free-address-high 1-6
free-address-low 1-6
low and high address thresholds 20-22
recipients, creating 1-6
trimming
immediate subnet utilization 9-17
subnet utilization records 9-16
Trivial File Transfer Protocol
See TFTP
try, DHCP expression 25-18
TSIG keys 28-11
creating 28-11
DNS update configuration attributes 28-13
failover synchronization effect 27-7
importing 28-13
pulling 28-11
pushing 28-11
rules for secrets 28-13
secrets, generating 28-12
TTL property A-1
default 17-12, 21-2
responses 17-12
maximum
cache 17-15
memory cache size 17-15
negative cache 17-14, 1-6
zone 17-12
tutorial
local cluster 5-15
regional cluster 5-25
TXT records A-7
tz-database, DHCP option B-17
tz-posix, DHCP option B-17

U

uBR 10000 routers 11-2
uBR 7200 routers 5-28, 11-2
Ultra-SCSI disk 4-3
unicast, enabling 23-5
unified address space 9-2
Universal Resource Identifiers (URIs) 16-9
UNIX, troubleshooting tools 7-25
update-policy (CLI command)
create 28-14
delete 28-16
rules
add 28-16
remove 28-16
user interfaces 2-1
user preferences, setting 2-10
user-auth, DHCP option B-17
user-class, DHCPv6 option B-11
users
differentiated services 19-11
event warnings 7-7
lease availability 19-7
types of 4-1
utility programs
ipconfig 20-24
ping 22-7
third party backup 8-3

V

vendor-class, DHCPv6 option B-11
vendor-encapsulated-options, DHCP option B-15
vendor-opts, DHCPv6 option B-12

virtual channel identifier 25-23
virtual path identifier 25-23, 1-10
virtual private networks
See VPNs
virtual routing and forwarding table ID
See VRFs
virus scanning, excluding directories 8-10
visibility of attributes 2-5
v-i-vendor-class, DHCP option B-18
v-i-vendor-info, DHCP option B-18
vmstat tool (UNIX) 7-25
vpn command (CLI)
create 23-17
set
vrf-name 23-17
vpn-id, DHCP option B-18
VPNs 19-5, 23-14
configuring 23-14
creating 23-16
current 23-18
failover synchronization effect 27-7
identifier 23-17
IOS support 23-14
leases in, importing 23-19
local
pulling 6-17
pushing 6-16
orphaned leases 23-21
regional 6-16
scopes 20-11, 23-17
setting current 20-11
VRFs 23-17
VRFs 23-15, 23-17

W

Web UI 1-1
attributes
displaying 2-5
modifying 2-5
visibility 2-5
changes, committing 2-4
deployment scenarios 4-2
help
attributes 2-6
topics 2-7
logging 7-8
logging in 2-2
navigation 2-4
Well Known Services record
See WKS records
Windows
CLI location 2-12
Event Viewer 7-7
ipconfig utility 24-12
logging 7-6
mouse buttons xxx
Windows 2000
client properties 24-11
design practices 28-25
DNS update 28-20, 28-27
domain controllers 28-27
environments 28-26
SRV records 28-23, A-7
Windows SMS
See SMS
winipcfg utility 24-12
WKS records A-8, 1-10
www-servers, DHCP option B-7

X

x-display-managers, DHCP option B-7

Y

yiaddr, DHCP field 23-15, 1-10

Z

zone
- edit mode, setting 15-2
- templates
  - cloning 15-5
  - creating using CLI 15-5

zone (CLI command)
- listRR
  - dns 28-17
- set
  - update-policy-list 28-17

zone command (CLI)
- addDNSRR 10-2, 16-3
- addHost 10-2
- addRR 10-2, 16-3
  - A 15-8
  - PTR 15-13
  - -staged or -sync 16-3
- applyTemplate 15-8
- chkpt 17-10
- cleanRR 16-9
- create
  - primary 15-8
  - secondary 15-15
  - template, using 15-8
- dumpchkpt 17-10
- enable
  - notify 17-11
- findRR 16-8
- forceXfer 15-16
- get
  - serial 15-7, 15-8
- getScavengeStartTime 28-19
- listHosts 10-2, 15-8
- listRR 15-9, 16-6
- removeCacheRR 7-3
- removeDNSRR 10-3, 16-5
- removeHost 10-3
- removeRR 10-3, 16-5
- set
  - checkpoint-interval 17-10
  - defttl 17-12
  - dist-map 15-23
  - expire 17-13
  - log-settings 28-19
  - nameservers 15-8
  - notify-set 17-11
  - refresh 17-13
  - retry 17-13
  - scvg-ignore-restart-interval 28-18
  - scvg-interval 28-18
  - scvg-no-refresh-interval 28-18
  - scvg-refresh-interval 28-18
- show 15-8
- subzone-forward=no-forward 17-5
- syncToDns 15-9

zone distributions
- creating 7-18, 15-23
- listing 7-18
- managing 15-19
- master servers, adding 15-23
- secondaries, adding 15-23
- synchronizing 15-23

zone templates
- applying to zones 15-5
- creating 15-2
- zone distributions, associating 15-23

zone transfers
- defined 14-5
- enabling 15-15
- forcing 7-3, 15-16
- forcing all 15-16

zone tree, viewing 5-20

zone-dist command (CLI)
- addSecondary 15-23
- create 7-18, 15-23
- list 7-18
- set
  - master-servers 15-23
- sync 15-23

zone-regexpr syntax 5-22

zones
- address restrictions 5-22
- applying update policies 28-17
- authoritative name servers 15-7
- cached resource records, removing 16-5
- caching-only servers 17-6
- configuring 10-1, 11-1, 16-1, 17-1
- count on server, getting 15-13
- defined 14-3
- delegated subzone records, editing 15-18
- delegated subzones, removing 15-18
- delegation-only 17-7
- difference from domains 14-3
- displaying resource records 16-5
- DNS update, enabling 15-19
- domains, adding 16-2
- dual zone updates 28-22
- edits
  - staged and synchronous modes 15-1
- host tables, editing 10-3
- hostmaster 15-7
- hosts
  - adding 10-1, 16-11
  - removing 10-3
- importing 15-9
- infrastructure 5-19
- listing 5-20
- loopback, *See* loopback, zones
- names, creating 15-6
- point of delegation 14-3
- primary server, setting 15-7
- resource records
  - adding 16-2
  - filtering 16-8
  - protecting 16-3
  - removing 16-4
- restricting hosts 5-22
- reverse, *See* reverse zones
- scavenging 7-3, 28-18
  - start time, getting 28-19
- secondary
  - expiration time 17-13
  - refresh time 17-12
  - retry time 17-13
- secondary name servers, adding 15-14
- secondary reverse, adding 15-15
- serial numbers 15-7
- staged edit mode 1-8
- subzones, delegating 1-9
- synchronous edit mode 1-9
- template, adding from 15-8
- transfers, *See* zone transfers
- TTL property, setting 17-12
- zone distributions, associating 15-23
- zone transfers, enabling 15-15

zone-template command (CLI)
- apply-to 15-5
- create 15-5
  - clone 15-5
- set
  - dist-map 15-23