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 19, "Configuring Scopes and Networks." The additional properties are:

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 CNS 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.

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's IP address (*siaddr*), and server's host name (*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's IP address
- *sname*—Optional server host name

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's boot process:

- *packet-siaddr*—IP address of the next server
- *packet-file-name*—Name of the boot file
- *packet-server-name*—Host name 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 re-use 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 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.

Setting Advanced DHCP Server Parameters

Table 22-1 describes the advanced DHCP server parameters that you can set in the local cluster Web UI and CLI.

Table 22-1   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 LDAP server's capacity to service client lookups.
		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.

Table 22-1   DHCP Advanced Parameters (continued)

Advanced Parameter	Action	Description
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 22-6.
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.

Step 1   In the local cluster Web UI, click **DHCP**, then **DHCP Server** to open the Manage DHCP Serve page.

In the CLI, 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 22-1).

Step 2   In the Web UI, click the name of the server.

Step 3   In the Web UI, add or modify attributes on the Edit DHCP Server page.

Step 4   In the Web UI, click **Modify Server** to make the changes.

Deferring Lease Extensions

The *defer-lease-extensions* attribute allows the DHCP server to optimize its response to a sudden flood of DHCP traffic. The parameter is enabled by default. 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 were to happen, the devices attached to the CMTSs produce a flood of DHCP traffic as they quickly come back online.

When you enable the *defer-lease-extensions* attribute, a DHCP client that renews a lease that is less than halfway to its expiration does not have its lease extended to an additional full lease time. Instead, the client receives a lease corresponding to the remaining time on its 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.

If a client is more than halfway to its expiration, this setting has no effect, and the lease is extended to the full configured lease interval as usual, including the database writes.

✎ **Note**   Deferring lease extensions significantly increases the server's 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's 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's 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 22-1 on page 22-4.) 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 machine's 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 of the local cluster Web UI) 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   Reload the DHCP server and check the name_dhcp_1_log file for a successful load.

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 28, "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.

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. Dictionaries: environment only.

2. **post-packet-decode**—Rewrites the input packet. Dictionaries: request and environment.

3. **post-class-lookup**—Evaluates the result of a *client-class-lookup-id* operation on the client-class. Dictionaries: request and environment.

4. **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.

5. **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.

6. **check-lease-acceptable**—Changes the results of the lease acceptability test. Do this only with extreme care. Dictionaries: request, response, and environment.

7. **lease-state-change**—Determines when the lease state changes this only with extreme care. Dictionaries: response and environment.

8. **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.

9. **pre-dns-add-forward**—Alters the name used for the DNS forward (A record) request. Dictionaries: environment only.

10. **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.

To extend the DHCP server, do the following:

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

Cisco CNS Network Registrar User's Guide

- Windows:

  – For 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 **extension** command to configure the DHCP server to recognize this extension.

Step 3 Attach the configured extension to one or more DHCP extension points using **dhcp attachExtension**.

Step 4 Reload the server.

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 22-1 on page 22-4 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 22-2.

Table 22-2   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 that client's current network location.
dns-update-detail (7)	Logs each sent and replied DNS update.

Table 22-2   DHCP Log Settings (continued)

Log Setting (Numeric Equivalent)	Description
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.

Cisco CNS Network Registrar User's Guide

- 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 23-12).

- Check the server statistics to aid in monitoring server performance (see the "Displaying Statistics" section on page 6-8).

- Consider setting the scope allocation priority (see the "Configuring Multiple Scopes Using Allocation Priority" section on page 19-7).

- If pinging hosts before offering addresses, consider adjusting the ping timeout period (see the "Pinging Hosts Before Offering Address" section on page 21-5).

- 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 23-17.

- If using DHCP failover, consider using the load balancing feature (see the "Setting Load Balancing" section on page 26-17).

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 CNS 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.

Configuring Virtual Private Networks Using DHCP

VPNs that you create provide a filtering mechanism for:

- Viewing the unified address space (see the "Viewing Unified Address Space" section on page 8-2).

- Listing address blocks (see the "Adding Address Blocks" section on page 8-5).

- Listing subnets (see the "Address Blocks and Subnets" section on page 8-4).

- Querying subnet utilization (see the "Generating Subnet Utilization History Reports" section on page 8-13).

- Querying lease history (see the "Running IP Lease Histories" section on page 21-16).

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.

Typical Virtual Private Networks

Figure 18-4 on page 18-5 shows a VPN scenario with DHCP client 1 as part of VPN *blue* and DHCP client 2 in VPN *red*. Both 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, host name, 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 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:

- In the local cluster Web UI—Click **DHCP**, then **VPNs**. This opens the List/Add VPNs page. Give the VPN a numerical key identifier and a unique name in the cluster.

- In the regional cluster Web UI—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 (see Figure 22-1). 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.

Figure 22-1   List/Add VPNs Page (Regional)

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:

    – In the local and regional cluster Web UI—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.

    – In the local and regional cluster Web UI—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   In the Web UI or CLI—Add a description for the VPN, if you wish.

Step 5   In the Web UI—Click **Add VPN**.

Step 6   Create a scope for the VPN. Keep the VPN name and scope name as similar as possible for identification purposes:

- In the local cluster Web UI—Click **DHCP**, then **Scopes**. This opens 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—You can 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   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.

  – In the local and regional cluster Web UIs—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.

  In the local cluster Web UI—Click **DHCP**, then **Client-Classes**. Create or edit a client-class and enter the *default-vpn* and *override-vpn* attribute values.

  In the regional cluster Web UI—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 21-3.)

```
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 Virtual Private Networks" section on page 22-14.
  - In the local cluster Web UI—Click **DHCP**, then **Scopes**. Create or edit a scope and set the Miscellaneous attribute *vpn-id*.
  - In the regional cluster Web UI—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 22-17.
- 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 18-5 on page 18-6 shows a sample subnet allocation configuration with subnets assigned to provisioning devices, along with the conventional DHCP client/server configuration. The subnet allocation feature is available only in the CLI and not in the Web UI.

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 22-12).

  ✎
  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's 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   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 Network Registrar's 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 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's 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.

Free Manuals Download Website

http://myh66.com

http://usermanuals.us

http://www.somanuals.com

http://www.4manuals.cc

http://www.manual-lib.com

http://www.404manual.com

http://www.luxmanual.com

http://aubethermostatmanual.com

Golf course search by state

http://golfingnear.com

Email search by domain

http://emailbydomain.com

Auto manuals search

http://auto.somanuals.com

TV manuals search

http://tv.somanuals.com