



Configuring Client-Classes and Clients

You can use Cisco CNS Network Registrar's client or client-class concept 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. If you do not enable client-class processing, the DHCP server provides client leases based solely on their network location.

Client-Class Process

You can enable or disable client-class processing for the DHCP server and apply a set of properties to groups of clients. With client-class processing enabled, the DHCP server assigns the client to an address from a matching scope. The server acts according to the data in each packet. To configure client-class:

1. Enable client-class processing for the DHCP server.
2. Define client-classes that include or exclude selection tags.
3. Apply the selection tags to specific scopes.
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 23-5.

Setting Up Client-Classes on Servers

Setting up client-classes involves:

1. Enabling client-class processing on the DHCP server.
2. Creating scope-selection tags (pre-6.0 only).
3. Creating the client-classes themselves.

Enabling Client-Class Processing

- Step 1** In the local cluster Web UI, click **DHCP**, then **DHCP Server** to open the Manage DHCP Server page.
- Step 2** Click the name of the server.
- Step 3** On the Edit DHCP Server page, under the Client Class attribute category, set *client-class* to enabled.

Step 4 Click **Modify Server**.

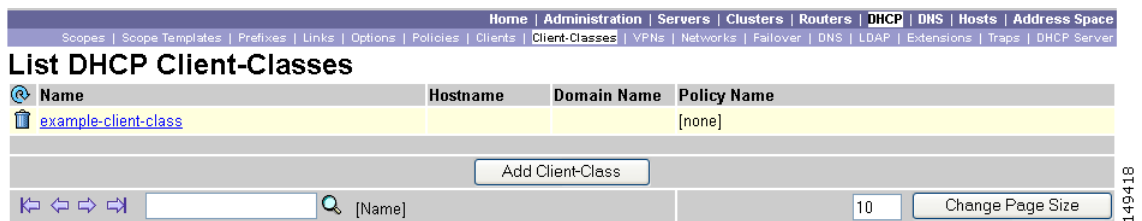
In the CLI, use `dhcp enable client-class` to enable client-class processing.

Defining Client-Classes and Their Properties

The next step is to define the client-classes themselves. Again, you do this on the server level.

Step 1 In the local cluster Web UI, click **DHCP**, then **Client-Classes** to open the List DHCP Client-Classes page (see [Figure 23-1](#)).

Figure 23-1 List DHCP Client-Classes Page (Local)



Step 2 Click **Add Client-Class** to open the Add DHCP Client-Class page (see [Figure 23-2](#) for a partial view).

Figure 23-2 Add DHCP Client-Class Page (Local)

Attribute	Value	Data Type	Default	Unset?
selection-criteria	<input type="text"/>			<input type="checkbox"/>
selection-criteria-excluded	<input type="text"/>			<input type="checkbox"/>
action	<input type="checkbox"/> exclude <input type="checkbox"/> one-shot <input type="checkbox"/> use-release-grace-period <input type="checkbox"/> none	flags		<input type="checkbox"/>
limitation-id	<input type="text"/>	expression		<input type="checkbox"/>
over-limit-client-class-name	<input type="text"/>	string		<input type="checkbox"/>

Step 3 Enter the client-class name, a client name (if you want to associate a client with the client-class), and the client-class domain name, and click a predefined policy from the Policy name drop-down list.

Step 4 Click **Add Client-Class**.

Step 5 On the List DHCP Client-Classes page, click the name of the newly created client-class.

Step 6 On the Edit DHCP Client-Class page, set the attributes.

Step 7 Click **Modify Client-Class**.

In the CLI:

- To create a 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.
 - To set the properties of the clients in the client-class, use **client-class name set attribute=value**.
-

Defining Host Name Properties

You can specify the host name that each client should adopt, using the *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 host name option the client sent.
- *@no-host-name-option*—The server ignores the host name sent by the client. If DNS name generation is in effect, a generated name is used, if set up as such for dynamic DNS updating.
- *@use-macaddress*—The server synthesizes a host name from the client's MAC address, hyphenates the octets, then adds an **x** at the front. For example, if a client's MAC address is 1,6:00:d0:ba:d3:bd:3b, the synthesized host name would be x1-6-00-d0-ba-d3-bd-3b.
- **<Not Specified>**—Leaves the host name unspecified.

Associating Policies

You can set the appropriate policy to associate with, and the action to perform for, the client-class by using the client-class *policy-name* attribute. In the Web UI, this is the Policy name field on the Add or Edit DHCP Client-Class page. In the CLI, use **client-class name set policy-name=value**.

Setting Other Properties

You can set all the other attributes for a client-class that you can for a client, such as the domain name, *action*, and the *user-defined* string. See the “[Setting Client Properties](#)” section on page 23-7 for details.

In the Web UI, these attributes are on the Add or Edit DHCP Client-Class page. In the CLI:

- To show the properties for a particular client-class, use **client-class name [show]**.
- You can also list the properties for all the client-classes created, or list just their names.
- To delete the client-class, use **client-class name delete**.
- To debug client-class problems, use **dhcp set log-settings=client-criteria-processing**.

Setting Client-Class Scope Selection Criteria

If you omit a general action to perform on a client-class, you can specify which scope-selection tags to include or exclude.

If a scope has a selection tag assigned to it and client-class assigns an:

- Inclusion tag, then the client can get an address from that scope.
- Exclusion tag, then the client will not get any address from that scope.

For example, assume three scopes, A, B, and C, with these attributes—A(red), B(blue), C(blue,green). If a client-class specifies inclusion of red, then the client gets an address from scope A. Inclusion of blue gives the client an address from either scope B or C. Inclusion of blue and exclusion of green gives the client an address from scope B only.

**Tip**

Avoid setting conflicting inclusion and exclusion criteria. Ensure that they are mutually exclusive.

-
- Step 1** In the local cluster Web UI, create the client-class.
- Step 2** On the List DHCP Client-Classes page, click the name of the client-class.
- Step 3** On the Edit DHCP Client-Class page, set these two attributes:
- a. *selection-criteria*—Enter a selection tag to use to include this client-class in the scope.
 - b. *selection-criteria-excluded*—Enter a tag to use to exclude this client-class from the scope.
- Step 4** Click **Modify Client-Class**.
- In the CLI, use **client-class *name* set selection-criteria** to set the inclusion criteria and **client-class *name* set selection-criteria-excluded** to set the exclusion criteria.
-

Associating Selection Tags with Scopes

The next step is to associate the appropriate selection tags with the scope, which must be under the server you configured.

-
- Step 1** In the local cluster Web UI, create the scope.
- Step 2** On the List/Add DHCP Scopes page, click the name of the scope.
- Step 3** On the Edit DHCP Scope page, in the Selection Tags area, include in the Selection Tag field a comma-separated list of scope-selection tags created in the *selection-criteria* attribute for the client-class.
- Step 4** Click **Modify Scope**.
- In the CLI, use **scope *name* set selection-tags** to associate existing selection tags with a scope.
- Step 5** Reload the DHCP server.
-

Configuring Embedded Policies for Client-Classes

Network Registrar automatically creates an embedded policy for each client-class. The embedded policy has no properties or DHCP options associated with it until you add them. This is like an embedded policy for a scope.

-
- Step 1** In the local cluster Web UI, create the client-class.
- Step 2** Click the name of the client-class on the List DHCP Client-Classes page.
- Step 3** Click **Edit Embedded Policy** to open the Edit DHCP Embedded Policy for Client-Class page.

Step 4 Modify the fields, options, and attributes on this page. If necessary, unset attributes.

Step 5 Click **Modify Embedded Policy**.

In the CLI:

- To see if there are any embedded property values already set for a client-class, use **client-class-policy**, using the client-class name as the policy name.
- You can enable, disable, get, set, and unset client-class embedded policy attributes. Note that deleting a client-class policy unsets all of its embedded policy properties.
- You can also list, get, set, and unset DHCP options and vendor options.
- If necessary, set the lease time for the embedded client-class. Verify by listing the options.

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 of these external options use DHCP option 82, as described in the [“Subscriber Limitation Using Option 82”](#) section on page 23-12. 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)

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 24, “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 scope-selection tag or client-class name, or append the value to the scope-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 23-1. See the following subsections. Note that to use the client-class feature, the DHCP server *client-class* attribute must be enabled.

Processing Order to Determine Client-Classes

This is the order in which the possible sources are used to determine client-class names:

1. The client-class name in the extension environment dictionary is used.
2. If a real client-entry is found in the database, the client-entry's *client-class-name* is used. Otherwise, enable the *skip-client-lookup* DHCP server attribute to prevent unnecessary database reads.
3. If the RADIUS framed-pool value is mapped to a client-class (**dhcp set map-radius-pool-name=map-as-class**), then the framed-pool value is used.
4. If the RADIUS class value is mapped to a client-class (**dhcp set map-radius-class=map-as-class**), then the class value is used.
5. If the *dhcp-user-class-id* DHCP option (77) is mapped to a client-class (**dhcp set map-user-class-id=map-as-class**), then the option value is used. (Note that you can alternatively use a lookup ID expression instead of this mapping; see the [“Client-Class Lookup Expression Processing”](#) section on page 23-14.)
6. If no mapping or user-class ID is found, the default-client-class-name from the environment dictionary is used.
7. If no default-client-class-name or client-entry is found, the client-class-name from the client named **default** (if found) is used.

Processing Order to Determine Scope-Selection Tags

This is the order in which the possible sources are used to determine scope-selection tags (the server uses the first non-null source):

1. The scope-selection tags in the extension environment dictionary.
2. If a real client-entry is found in the database, the client-entry's *scope-selection-tags*. Otherwise, enable the *skip-client-lookup* DHCP server attribute to prevent unnecessary database reads.
3. Scope-selection tags in the client's client-class.
4. If the RADIUS framed-pool value is available and mapped to a tag (**dhcp set map-radius-pool-name=map-as-tag**), then it is used.
5. If the RADIUS class value is available and mapped to a tag (**dhcp set map-radius-class=map-as-tag**), then it is used.
6. If the *dhcp-user-class-id* DHCP option (77) is available and mapped to a tag (**dhcp set map-user-class-id=map-as-tag**), then it is used.

Next, one of the following could be appended to the list of selection tags (if any) found:

1. If a RADIUS framed-pool value is available and the *map-radius-pool* DHCP attribute is set to append to the selection tags (**dhcp set map-radius-pool=append-to-tags**), then it is appended.
2. If a RADIUS class value is available and the *map-radius-class* DHCP attribute is set to append to the selection tags (**dhcp set map-radius-class=append-to-tags**), then it is appended.
3. If a *dhcp-user-class-id* is available and the *map-user-class-id* DHCP attribute is set to append to the selection tags (**dhcp set map-user-class-id=append-to-tags**), then it is appended.

Troubleshooting Client-Classes

To troubleshoot 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. 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's client-class.
- **client-criteria-processing**—Logs a message whenever the server examines a scope 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 from.

- Is client-class enabled?

If client-class is enabled, but you are getting 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. Set the LDAP *can-query* attribute to true. You also must configure the DHCP server to use LDAP for queries.

- Is the server providing clients the right data, but you are seeing 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 what scopes the server is getting addresses from. If it is not getting them from the expected scopes, explicit relationships might be incorrectly defined. A scope that you thought was a secondary scope might not be defined that way.

- Did you set the include and exclude for scope-selection tags properly?

If you define a series of scope-selection tags to include, a scope's tags must match those of the client. If you define a series to exclude, a scope 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.

Setting Client Properties

Set the DHCP client properties. These properties include the client's participating client-class, its associated policy, the action to perform, and the inclusion and exclusion criteria for scope-selection tags.

Adding and Editing Clients

A client inherits the properties from its client-class, which you may choose to override or supplement by specifying different ones for the client.

- Step 1** In the local cluster Web UI, click **DHCP**, then **Clients** to open the List/Add DHCP Clients page (see Figure 23-3).

Figure 23-3 List/Add DHCP Clients Page (Local)

Name*	Client-Class Name
<input type="text"/>	[none]
<input type="button" value="Add Client"/>	
Name	Client-Class Name
1,6,00:d0:ba:d3:bd:3b	[none]
client-1	[none]

Navigation: Home | Administration | Servers | Clusters | Routers | **DHCP** | DNS | Hosts | Address Space
 Sub-navigation: Scopes | Scope Templates | Prefixes | Links | Options | Policies | **Clients** | Client-Classes | VPNs | Networks | Failover | DNS | LDAP | Extensions | Traps | DHCP Server

Page controls: [Previous] [Next] [Search] [Name] [Page Size: 10] [Change Page Size]

- Step 2** Enter the client's 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*.)
- 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 23-4 for a partial view).

Figure 23-4 Add DHCP Client Page (Local)

Attribute	Value
Name *	<input type="text"/>
Client-class name	[none]
Host name	<input type="text"/>
Domain name	<input type="text"/>
Policy name	[none]

Attribute	Value	Data Type	Default	Unset?
selection-criteria	<input type="text"/>			<input type="checkbox"/>
selection-criteria-excluded	<input type="text"/>			<input type="checkbox"/>
action	<input type="checkbox"/> exclude <input type="checkbox"/> one-shot <input type="checkbox"/> use-release-grace-period <input type="checkbox"/> none	flags		<input type="checkbox"/>

Navigation: Home | Administration | Servers | Clusters | Routers | **DHCP** | DNS | Hosts | Address Space
 Sub-navigation: Scopes | Scope Templates | Prefixes | Links | Options | Policies | **Clients** | Client-Classes | VPNs | Networks | Failover | DNS | LDAP | Extensions | Traps | DHCP Server



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.

Step 5 On the Add Client page, add any attributes for the client, including scope selection criteria.

Step 6 Click **Add Client** at the bottom of the page.

In the CLI:

- Use **client name create**, specifying the name by client ID (MAC address, DUID, or lookup key).
 - 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 host name.
 - To set the client properties, use **client name set attribute=value**:
 - 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 23-10](#).
 - 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**.
 - To display properties of a specific client, use **client name [show]**.
 - To display properties for all the clients, use **client list**, or **client listnames** to list just the names.
 - To delete a client, use **client name delete**.
-

Configuring Embedded Policies for Clients

Network Registrar automatically creates an *embedded* policy for each client. The embedded policy has no properties or DHCP options associated with it until you enable or add them.

Step 1 In the local cluster Web UI, 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 Click **Edit Embedded Policy** to open the Edit DHCP Embedded Policy for Client page.

Step 4 Modify the fields, options, and attributes on this page. If necessary, unset attributes.

Step 5 Click **Modify Embedded Policy**.

In the CLI, use **client-policy**, using the client MAC address (or **default**) as the client policy name.

Setting Windows Client Properties

Windows 2003 and XP clients support class-based provisioning. You can set certain properties in the CLI that relate to client-class processing. These are:

- Looking up the client entry to determine the default client for client-class processing.
- Mapping the user class ID to the client-class or scope-selection tag.
- Whether to append the class ID to the scope-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 23-11. 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 of the local cluster Web UI 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 of the local cluster Web UI, 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 the lease is used by a different client. 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, 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 want not 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 not set, or if there is no client-class entry in it, the DHCP server ignores the request.

Here are the valid authentication values:

- **+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 time that is local to the **nrcmd** process. 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.

Here is an example 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's 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 scope-selection tags.
 - Step 4** Enable client-class processing for the server.
 - Step 5** Reload the DHCP server.
-

Setting Client Caching Parameters

A client's initial request for an address from a DHCP server often goes through a DHCPDISCOVER-DHCP OFFER-DHCP REQUEST-DHCP ACK cycle. This process requires that the DHCP server must consult the database twice for client data per request. 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 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 of the local cluster Web UI, or **dhcp set client-cache-count** in the CLI. By default, the maximum number to cache is 1000 clients. To disable the client cache, set the attribute to 0.

The client cache is usually valid for only ten seconds, called the cache time to live (TTL). After the TTL expires, the server reads the client information from the database, if need be. You can adjust this TTL using the *client-cache-ttl* attribute on the Edit DHCP Server page of the local cluster Web UI, 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's 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 provided by the DHCP server, but to limit the number of these addresses. One way to accomplish this is to use the client-class capability to register (or provision) each customer device. In this scenario, IP addresses are issued 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.

Limiting customer devices on a per-subscriber basis gives rise to the idea of limiting them based 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 suboptions of the *relay-agent-info* option 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, but it might not be unique across all subscribers on the switch.

The goal of this implementation is to allow the network administrator to configure limitations on subscriber use of the DHCP-allocated addresses without seriously impacting other capabilities of the DHCP server. 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.

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.

At a high level, this is configured by creating an *expression* that is evaluated for each incoming packet and returns the name of the client-class where the client should be placed. See [Chapter 24, “Using Expressions”](#) for details on the use of expressions.

To handle this, each client-class allows specification of a limitation identifier (ID), a key to be determined from the incoming packet and then used by the server in a later processing step to do the actual limitation on the number of devices. All devices with the identical limitation ID (the *limitation-id* property) are considered 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*), then the subscriber is a cable modem and should be assigned to 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 assigned to the *docsis-cm-client-class*.
3. If the *user-class* option value matches the string *alternative-class*, then the subscriber should be assigned to the *alternative-cm-client-class*.

Calculating Client-Classes and Creating Keys

The expression that determines the client-class is set by the *client-class-lookup-id* attribute of the DHCP server in the Web UI, 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 24-1).

Clients and client-classes also allow specification of 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. An expression is configured server-wide with the *client-class-lookup-id* attribute. This expression is executed 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 to be used for the packet, or the distinguishing string <none>, which indicates 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 packet is dropped (with a 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’s 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 limitation is enforced only within 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 chain 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’s embedded or named policy, a scope’s embedded or named policy, or the system’s *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.

Limitation processing is not currently available for DHCPv6 clients.

Expression Processing for Subscriber Limitation

There are expressions in several places in the limitation processing. Each expression either evaluates to null or to a string (typically to determine a client-class name when looking up a client-class), or evaluates to a series of bytes (a blob) when creating a *limitation-id*. Expressions are used in these places:

- Looking up a client-class.
- Creating the key to use in limiting 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

To configure option 82 limitation properly:

**Note**

If you are not registering clients explicitly, do not enable client-class as a DHCP server property when using option 82 data.

-
- Step 1** Determine if you want to limit some clients and not others. If you want to limit some clients:
- Find some method to distinguish these clients from the others, based on some values contained in the DHCP requests from each class of clients.
 - 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 2** 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 into 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 3** 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 4** Create all these selection tags, client-classes, and scopes.
- Step 5** Configure the *limitation-count* in a policy, probably the named policy associated with the client-class for the clients to be limited.
- Step 6** Write the expression to separate the incoming clients into those to be limited and those not be limited. Configure it on the DHCP server by setting the *client-class-lookup-id* attribute.
- Step 7** Write the expression to determine the limitation ID for the devices to be limited, and configure it on the client-class for clients to be limited by setting the *limitation-id*. (See [Figure 23-2 on page 23-2](#) for a view of how to set this value on the Add DHCP Client-Class page.)
-

DHCP Renewal Processing

Remember that only packets that are broadcast from the DHCP client arrive at the DHCP 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. The first is to place all packets that do not have option 82 data into 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. The second approach is to place a renewal 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 appears to offer more security, 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 ever do this unless the server lost some of its state. In this case, you would want it to give the address to the client. 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 is logged in the DHCP log file whenever the client data is logged. The *limitation-id* used appears as "... LID: *nnn:nnn:nnn*..." in the log file. 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:

- In the local cluster Web UI, on the Manage DHCP Server page, click the Run icon () in the Commands column to open the DHCP Server Commands page (see [Figure 6-4 on page 6-3](#)). 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 is used just like a *giaddr* to determine the subnet. Any IP address that could appear in any scope (primary or secondary) for the network can be used to specify a subnet. If you specify only the *ipaddr*, it must be an address that is served by the DHCP server, and the command returns all of the clients and corresponding leases they use.

If a client is denied service due to a *limitation-count* overflow, a message such as this would appear 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 currently using up the *limitation-count*, thus causing a denial of service for the 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 using `dhcp setDebug VX=1` (`dhcp setDebug VX=0` 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 24-21.

When things seem to be going strangely, or when submitting log files to report a problem, it is important to enable some additional tracing using `dhcp setDebug QR57=9` (`dhcp setDebug QR57=0` 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, but the page is invaluable for understanding what is going on inside the server.

Expression Examples

For examples of how to use expressions for option 82 processing, see the “[Expression Examples](#)” section on page 24-17.

Configuring LDAP

This section describes the Lightweight Directory Access Protocol (LDAP) with which you can use directory services to integrate Cisco CNS 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’s database, you can ask the Network Registrar DHCP server to issue queries to one or more LDAP servers for information in response to DHCP client requests.

Network Registrar now uses the iPlanet LDAP Software Development Kit (SDK) version 5.0. Previous releases used SDK version 3.0.

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 is not dependent on any 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 if you disable 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 could 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 LDAP Remote Servers

To add LDAP remote servers in the local cluster 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 23-5 for a partial view). 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.

In the CLI, use `ldap name create domain-name`:

```
nrcmd> ldap myserver create myserver.example.com
```

Figure 23-5 Add LDAP Remote Server Page (Local)

Attribute	Value	Data Type	Default	Unset?
name*	<input type="text"/>	string		<input type="checkbox"/>
hostname*	<input type="text"/>	string		<input type="checkbox"/>
port	<input type="text"/>	unsigned 32-bit		<input type="checkbox"/>
username	<input type="text"/>	string		<input type="checkbox"/>
password	<input type="text"/>	string		<input type="checkbox"/>
can-create	<input type="radio"/> enabled <input checked="" type="radio"/> disabled	boolean	disabled	<input type="checkbox"/>
can-query	<input type="radio"/> enabled <input checked="" type="radio"/> disabled	boolean	disabled	<input type="checkbox"/>
can-update	<input type="radio"/> enabled <input checked="" type="radio"/> disabled	boolean	disabled	<input type="checkbox"/>
search-path	<input type="text"/>	string		<input type="checkbox"/>
search-scope	SUBTREE <input type="button" value="v"/>	32-bit enum	SUBTREE	<input type="checkbox"/>
search-filter	<input type="text"/>	string		<input type="checkbox"/>
Query Settings				
Attribute	Value	Data Type	Default	Unset?
query-dictionary	<input type="text"/>			<input type="checkbox"/>
dn-format	<input type="text"/>	string		<input type="checkbox"/>

149421

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's MAC address.

**Note**

When connecting to an LDAP server, specify the user's *distinguished name*. The distinguished name is the way to uniquely identify an object in the LDAP schema. It is like a unique key in a database or a fully qualified path name for a file. For example, a distinguished name 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 Tell DHCP about your LDAP server by supplying a host name. In the local Web UI, on the LDAP Remote Server page (see [Figure 23-5 on page 23-18](#)), enter a value in the name field. In the local CLI, use this command, for example:

```
nrcmd> ldap myserver create myserver.example.com
```

Later, if you need to delete the server, use `ldap server delete`.

Step 2 Configure the connection credentials. Use the distinguished name for the user. In the Web UI, enter a value in the username field. In the CLI, use this command, for example:

```
nrcmd> ldap myserver 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 you specify the search scope to be `SUBTREE`, the server searches all the children of the search path. If you specify `ONELEVEL`, the server searches only the immediate children of the base object. If you specify `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:

In the Web UI, enter a value in the search-path field. In the CLI, use this command, for example:

```
nrcmd> ldap myserver 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`). In the Web UI, enter a value in the search-filter field. In the CLI, use this command, for example:

```
nrcmd> ldap myserver set search-filter=(cn=%s)
```

Step 5 Configure a query dictionary, which contains all the LDAP-to-DHCP mappings. Use `ldap servername setEntry` to set these mappings (you can do this in the CLI only):

- a. Retrieve the DHCP surname from the `sn` LDAP attribute:

```
nrcmd> ldap myserver setEntry query-dictionary sn=host-name
```

- b. Retrieve the client-class name from the first `givenname` LDAP attribute:

```
nrcmd> ldap myserver setEntry query-dictionary givenname=client-class-name
```

- c. Retrieve the domain name from the `localityname` LDAP attribute:

```
nrcmd> ldap myserver 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`. In the Web UI, set the `can-query` attribute to enabled. In the CLI, use this command:

```
nrcmd> ldap myserver enable can-query
```

Step 7 Enable client-class processing for the DHCP server. In the Web UI, 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. In the Web UI, 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 servers' preference values are ignored and all LDAP servers that are configured to handle client queries are treated equally, as are all servers configured to accept lease state updates.
- **failover**—The DHCP server always uses the active LDAP server with the lowest preference. If the preferred server loses its connection or fails, DHCP uses the next server in preference order, except that LDAP servers with equal preference are treated as being in round-robin mode.

You set the LDAP server preference (the lower the number, the higher the preference) in the Web UI by setting the preference attribute in the Advanced Settings section of the Add LDAP Remote Server page. Be sure that *ldap-mode* in the Miscellaneous attributes on the Edit DHCP Server page is set to failover.

In the CLI, use **ldap server set preference** and set the LDAP mode for the DHCP server by using **dhcp set ldap-mode**; for example:

```
nrcmd> ldap myserver set preference=1
nrcmd> ldap otherserver set preference=2
nrcmd> dhcp set ldap-mode=failover
```

Step 10 Show or list the LDAP configuration. In the Web UI, go to the List LDAP Remote Servers page, or in the CLI use:

```
nrcmd> ldap myserver
nrcmd> ldap list
nrcmd> ldap listnames
```

Step 11 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.

For example, if you want to unprovision the LDAP entry *givenname*, configure the search filter accordingly, as with this CLI command:

```
nrcmd> ldap myserver 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.



Note

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.
- For example, create an embedded client policy for 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
```
  - Get the client's embedded policy data:
 

```
nrcmd> client 1,6,00:d0:ba:d3:bd:3b get embedded-policy
100 Ok
embedded-policy="((ClassName Policy)(name
client-policy:1,6,00:d0:ba:d3:bd:3b)(v4reply-options [routers])(version
1)(option-list [((ClassName Option)(Number 3)(value
01:02:03:04:06:07:08:09)(option-definition-set-name dhcp-config)]))"
```
  - Extract what is between the quotes in the client output in the previous substep to the LDAP attribute (`businessCategory` in the example):
 

```
businessCategory:((ClassName Policy)(name
client-policy:1,6,00:d0:ba:d3:bd:3b)(v4reply-options [routers])(version
1)(option-list [((ClassName Option)(Number 3)(value
01:02:03:04:06:07:08:09)(option-definition-set-name dhcp-config)]))
```

The option values are translated into hexadecimal field syntax, including multiple values that were originally comma-separated.
  - 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 use the CLI to delete the dummy client:
 

```
nrcmd> client 1,6,00:d0:ba:d3:bd:3b delete
nrcmd> save
```

## Configuring DHCP LDAP Update and Create Services

You can configure the DHCP LDAP service to copy lease state data to existing attributes in the LDAP server. The service converts the lease state data to string form, and uses an update dictionary to map the LDAP attributes to the DHCP data values.

Each time the state of a lease changes, the DHCP server makes a copy of the data and then transfers it to the LDAP server that you have configured to store lease state data.

## Lease State Attributes

You can store any of these attributes about the lease's 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's 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 LDAP for Lease State Updates

Follow this procedure to configure lease state updates:

- 
- Step 1** Choose the 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 to this method is that lease data is stored directly with other associated 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 a client is moved off a lease by the server.

**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 client's 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 entry's distinguished name.

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 distinguished name of the entry. When the DHCP LDAP service locates the entry, the distinguished name 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 23-24](#).

## 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 distinguished name.

## 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's 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 data 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 (distinguished name) for an update.
- *dn-format*—The server is provided with the distinguished name for an update. In other words, the DHCP performs a direct update without having to query before an update.

### Option 1: Using update-search-path Option

The following example illustrates the first option, *update-search-path*. It shows what to do when the LDAP object's distinguished name (*dn*) 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 distinguished name to issue an LDAP update.

The example shown in [Table 23-1](#) assumes that you are using the standard LDAP organizational person object class attributes to hold lease update data.

**Table 23-1 LDAP-to-DHCP Mapping, Example 2**

| 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's host name 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 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's 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's data should be substituted, as in this 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, as in this CLI example:
- ```
nrcmd> ldap myserver enable can-update
```
- Step 8** Reload the DHCP server.
-

## Option 2: Using dn-format Option

This example illustrates using the second option, *dn-format*:

- 
- Step 1** Tell DHCP about the LDAP server by supplying the server's host name 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 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, as in this 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's 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, as in this CLI example:
- ```
nrcmd> ldap myserver_option2 enable can-update
```
- Step 7** 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, as in this 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 update-search-path Option](#)” section on page 23-24). Skip this step if you configure lease state updates using the *dn-format* string. (See “[Option 2: Using dn-format Option](#)” section on page 23-26.)
- Step 2** Specify the distinguished name of the entry to be created when combined with the existing *dn-attribute* property, as in this CLI example:
- ```
nrcmd> ldap myserver set dn-create-format="cn=%s\",ou=IT,o=Example Corp,c=US"
```
- Network Registrar's *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 distinguished names.

- 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, as in this CLI example:

```
nrcmd> ldap myserver set create-object-classes=
"top,person,organizationalPerson,inetorgperson"
```

- Step 5** Enable entry creation for the LDAP server myserver, as in this 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 23-24.

- Step 6** 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.

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=(amp;macaddress=%s)(amp;csrcclassname=Computer)(amp;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=(amp;macaddress=%s)(amp;csrcclassname=Computer)(amp;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

```

Typical LDAP Attributes and Recommended Values

Table 23-2 shows typical values for the LDAP attributes.

Table 23-2 LDAP Parameters and Recommended Values

Recommended Setting	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.
timeout=3	Number of seconds an LDAP request remains on a connection queue before being declared stale and timing out. The default setting of 3 seconds is recommended. Any response the DHCP client receives after the client's timeout period is stale. Network Registrar DHCP servers fail over at the <i>timeout</i> interval if dhcp set ldap-mode=failover and dhcp enable can-update or dhcp enable can-create are set.
query-timeout=3	Network Registrar DHCP servers fail over at the <i>query-timeout</i> interval if dhcp set ldap-mode=failover and dhcp enable can-query are set. The default setting is 3 seconds and is recommended.