# Managing Client-Classes and Clients

Use the Cisco Prime Network Registrar client and client-class concepts to provide differentiated services to users across a common network. You can group clients based on administrative criteria, and then ensure that each group receives its appropriate class of service (COS). Without client-class processing, the DHCP server provides client leases based solely on their network location.

## Configuring Client-Classes

You can differentiate client services in the following ways:

- Register clients using the Cisco Prime Network Registrar database (this section) or the Lightweight Directory Access Protocol (see Configuring Cisco Prime Network Registrar to Use LDAP, on page 17).
- Register intermediary devices (such as cable modems) so that you can differentiate their upstream clients by class of service.
- Use the contents of client packets without the foreknowledge of client data:
  - Known DHCP options that can be in the packet, such as the *dhcp-user-class-id* DHCP option (77), or the *radius-attribute* suboption of the *relay-agent-info* DHCP option (82, see Processing Client Data Including External Sources, on page 6).
  - Other data in the packet to extract using an expression in the *client-class-lookup-id* DHCP server attribute (see Calculating Client-Classes and Creating Keys, on page 14).
- Use a two-stage process of first creating a client-class to assign clients, then set a *client-lookup-id* for certain clients (see Expression Processing for Subscriber Limitation, on page 15).

## Client-Class Process

Enable or disable client-class processing for the DHCP server and apply a set of properties to groups of clients. With client-class enabled, the server assigns the client to an address from a matching DHCPv4 scope or DHCPv6 prefix. The server acts according to the data in the packet. To configure client-classes:

1. Enable client-class processing for the DHCP server.
2. Define client-classes that include or exclude selection tags (criteria).
3. Apply the selection tags to specific scopes or prefixes (or their templates).
4. Assign clients to these classes.

This process is for clients configured through Cisco Prime Network Registrar. For processing affected by data from external sources, see Processing Client Data Including External Sources, on page 6.

## Defining Client-Classes

You enable and define client-classes at the server level.

### Local Web UI

**Step 1** To enable client-classes, in the Basic or Advanced mode:

a) From the **Deploy** menu, choose **DHCP Server** under the **DHCP** submenu to open the Manage DHCP Server page.
b) Select the server on the DHCP Server pane.
c) On the Edit DHCP Server tab, enable the *client-class* attribute.
d) Click **Save**.

**Step 2** From the **Design** menu, choose **Client Classes** under the **DHCP Settings** submenu to open the List/Add DHCP Client Classes page.

**Step 3** Click the **Add Client Classes** icon in the Client Classes pane to open the Add DHCP Client Class dialog box.

**Step 4** Enter a name for the client-class.

**Step 5** Set other client-class properties. The hostname and domain name attributes are mainly used for DNS updates if not using a DNS update configuration (see Creating DNS Update Configurations). The hostname properties are described in Defining Client-Class Hostname Properties, on page 5. You can also choose the appropriate policy for the client-class.

**Step 6** Click **Add Client Class**.

**Step 7** Define the selection criteria.

The critical step in creating a client-class is defining its selection criteria so that you can associate the client-class with a DHCPv4 scope or DHCPv6 prefix. Use the *selection-criteria* attribute (see also Table 1: Selection Tag and Criteria Attributes Used ).

You can enter multiple selection tags by separating them with commas. The values have to match the selection tags set for the desired scope or prefix (see Setting Selection Tags on Scopes and Prefixes, on page 3).

**Step 8** To add an embedded policy to the client-class, see Editing Clients and Their Embedded Policies, on page 10.

**Step 9** Click **Save**.

**Step 10** Debug as needed. To debug client-class errors, enable the *client-criteria-processing* attribute in the Log Settings section of the Local DHCP Server page.

**Step 11** To delete a client-class, select the client and click the **Delete Client Classes** icon in the Client Classes pane on the left, and confirm the deletion.

### CLI Commands

Enable client-classes by using **dhcp enable client-class**. To create the client-class, use **client-class** *name* **create**. The name should clearly identify its intent. It is not case-sensitive; classPC is the same as Classpc.

Set properties of the clients in the client-class by using **client-class** *name* **set** *attribute*=*value*. For example, set the desired policy to associate with the client-class by using **client-class** *name* **set policy-name**=*value*. Associate a scope with the client-class by using **client-class** *name* **set selection-criteria**. (See the Setting Selection Tags on Scopes and Prefixes, on page 3).

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

## Configuring DHCPv6 Client-Classes

You can configure DHCPv6 client-class attributes, which are:

- *v6-client-lookup-id*—Key value to use to look up the DHCPv6 client in the client database (locally or through LDAP), specified as an expression that evaluates to a string (or a blob as a valid string).
- *v6-override-client-id*—Value that replaces any client-identity value in an incoming packet, specified as an expression that evaluates to a blob.

### Local Advanced Web UI

From the **Design** menu, choose **Clients** under the **DHCP Settings** submenu to open the List/Add DHCP Clients page. Select an existing client to open the Edit DHCP Client page or click the **Add Clients** icon on Clients pane to add a new client-class, choose the client-class that includes the DHCPv6 attributes that were set (see Configuring DHCPv6 Client-Classes, on page 3), then click **Save**.

**Tip** Disable the *validate-client-name-as-mac* attribute for the DHCP server.

### CLI Commands

Use **client list** or **client** *name* **show** to show the existing clients. To set the client-class name for the client, use **client** *name* **set client-class-name**=*value*. Also ensure that the *validate-client-name-as-mac* attribute is disabled for the DHCP server.

## Setting Selection Tags on Scopes and Prefixes

To assign clients to different address pools, you must define the DHCPv4 scope (or template) or DHCPv6 prefix (or template) with the selection tags that you specified in the selection-criteria for the client-class. All the selection-criteria tags that the client-class has must match the tags the scope or prefix has, even though the scope or prefix might have additional tags. If the client-class omits all selection-criteria, no limitations apply to the scope or prefix selection.

For example:

Scope A has tag1, tag2

Scope B has tag3, tag4

Assuming both scopes are on the same network, a client in a client-class with:

- Tag1, tag2, or both, gets leases from scope A.
- Tag3, tag4, or both, gets leases from scope B.
- One or more tags from both scopes (such as tag1 and tag3) does not get leases from either scope.
- No tags gets leases from either scope.

The table below describes the attributes Cisco Prime Network Registrar uses to refer to selection tags or selection criteria for network objects.

***Table 1: Selection Tag and Criteria Attributes Used***

| Object | Attribute |
|---|---|
| Client | *selection-criteria* |
| Client-class | *selection-criteria* |
| Scope | *selection-tag-list* |
| Scope template | *selection-tag-list* |
| Prefix | *selection-tags* |
| Prefix template | *selection-tags* |
| Address block | *selection-tags* |
| Subnets | *selection-tags* |

## Local Web UI

Create or edit a scope or prefix or its template; on the Add or Edit page for the scope or prefix (or its template), find the *selection-tags* attribute and enter a list of comma-separated selection tags created in the *selection-criteria* attribute for the client-class that you want to associate with this scope or prefix (or its template). Then save the changes and reload the DHCP server, if necessary.

## CLI Commands

Use **scope** *name* **set selection-tag-list**. For a scope template, use **scope-template** *name* **set selection-tag-list**. For a prefix, use **prefix** *name* **set selection-tags**. For a prefix template, use **prefix-template** *name* **set selection-tags**.

## Defining Client-Class Hostname Properties

You can specify the hostname that each client should adopt, using the Hostname (*host-name*) attribute of the client-class. This can be an absolute, valid DNS value to override the one included in the DHCP client request, or can be any of these:

- ***@host-name-option***—The server uses whatever hostname option the client sent.
- ***@no-host-name-option***—The server ignores the hostname that the client sends. If DNS name generation is in effect, the server uses a generated name, if set up as such for dynamic DNS updating.
- ***@use-macaddress***—The server synthesizes a hostname from the client MAC address, hyphenates the octets, then adds an **x** at the front. For example, if a client MAC address is 1,6:00:d0:ba:d3:bd:3b, the synthesized hostname would be x1-6-00-d0-ba-d3-bd-3b.

If you omit a value, the hostname is unspecified. You can also synthesize hostnames by using a DNS update configuration (see Creating DNS Update Configurations).

## Editing Client-Classes and Their Embedded Policies

Editing a client-class involves the same attributes as creating a client-class. You can also add and modify an embedded policy for the client-class so that you can set its policy options. The embedded policy has no properties or DHCP options associated with it until you add them. (See also Creating and Editing Embedded Policies). The client-class embedded policy setting is the third priority the DHCP server uses in its policy selection, after that set for the client itself (see DHCPv4 Policy Hierarchy).

### Local Advanced Web UI

**Step 1** Create the client-class.

**Step 2** Select the client-class in the Client Classes pane on the left to open the Edit DHCP Client Class page.

**Step 3** Make changes to attribute settings as required.

**Step 4** To add a new embedded policy for the client-class, click **Create New Embedded Policy**. If there is an existing embedded policy that you want to edit, click **Edit Existing Embedded Policy**. (If you want to unset the existing embedded policy, click **Unset** on the Edit DHCP Client-Class page; this resets the button to **Create New Embedded Policy**.)

a) Modify the fields, options, and attributes on this page. For example, under the DHCPv4 Options, set the client lease time by choosing **dhcp-lease-time [51]** from the drop-down list, enter a lease interval value in the Value field, then click **Add Option**. If necessary, unset attribute values.

**Step 5** Click **Save**.

### CLI Commands

To check if there are any embedded policy values already set for a client-class, use **client-class-policy** *client-class-name* **show**. To set the attributes for the embedded policy, use **client-class-policy** *client-class-name* **set** *attribute* =*value*.

To set the DHCP options, use one of these commands:

```
nrcmd> client-class-policy client-class-name setOption {opt-name | id} value [-blob]
[-roundrobin]
nrcmd> client-class-policy client-class-name setV6Option {opt-name | id}[.instance] value
[-blob] [-roundrobin]
```

```
nrcmd> client-class-policy client-class-name setVendorOption {opt-name | id} opt-set-name
value [-blob]
nrcmd> client-class-policy client-class-name setV6VendorOption {opt-name | id} opt-set-name
 value [-blob]
```

To set the lease time, use **client-class-policy** *client-class-name* **setLeaseTime** *value*.

## Processing Client Data Including External Sources

Information about network hosts running DHCP clients and their users can arrive at the DHCP server from several external sources. The server can use this data as part of client-class processing, and capture it in its lease database to make it available to the Cisco Prime Network Registrar management system.

Recently introduced external factors that can influence client definitions are:

- A *subscriber-id* suboption of the *relay-agent-info* DHCP option (82), whereby a network administrator defines a network subscriber or client and sends this data to the DHCP server.
- RADIUS authentication server data, as part of 802.1x protocol deployments where the RADIUS data can be helpful in DHCP decision making. In this case, a device can send the data as part of *radius-attribute* suboption attributes in the *relay-agent-info* DHCP option (82).

Both these external options use DHCP option 82, as described in Subscriber Limitation Using Option 82, on page 13. The RADIUS source can end the following attributes:

- Client user or account name (the *user* attribute)
- Administratively defined class string (the *class* attribute)
- Vendor-specific data (the *vendor-specific* attribute)
- Session timeout value (the *session-timeout* attribute)
- IP address pool to use for the client (the *framed-pool* attribute)
- IPv6 address pool to use for the client (the *framed-ipv6-pool* attribute)

Cisco Prime Network Registrar provides extension support for the *subscriber-id* suboption and the *user*, *class*, and *framed-pool* attributes of the RADIUS suboption, and expression support for all of the suboptions (see Using Expressions). Additionally, the DHCP server now includes attribute settings to configure how the server handles the RADIUS *class* and *framed-pool* attributes. Cisco Prime Network Registrar can use the server attributes to map the RADIUS attribute value as a selection tag or client-class name, or append the value to the selection tag that it finds in its client database. For example:

```
nrcmd> dhcp set map-radius-class=append-to-tags
```

For client-classes and selection tags determined from external resources such as RADIUS, the processing order is slightly more complex than that described in Client-Class Process, on page 2. See the following subsections. Remember that to use the client-class feature, you must enable the DHCP server *client-class* attribute.

### Processing Order to Determine Client-Classes

The order in which the DHCP server uses possible sources to determine client-class names is as follows:

**1.** It uses the client-class name in the extension environment dictionary.

2. If it finds a real client-entry in the database, it uses its *client-class-name*. (If you believe that looking up clients in a database is unnecessary, you can prevent the database lookup by enabling the *skip-client-lookup* DHCP server attribute; see Skipping Client Entries for Client-Classing, on page 12.)
3. If you map the RADIUS framed-pool value to a client-class (by using **dhcp set map-radius-pool-name=map-as-class**), it uses the framed-pool value.
4. If you map the RADIUS class value to a client-class (by using **dhcp set map-radius-class=map-as-class**), it uses the class value.
5. If you map the *dhcp-user-class-id* DHCP option (77) to a client-class (by using **dhcp set map-user-class-id=map-as-class**), it uses the option value. (Note that you can alternatively use a lookup ID expression instead of this mapping; see Client-Class Lookup Expression Processing, on page 14.)
6. If it finds no mapping or user-class ID, it uses the default-client-class-name from the environment dictionary.
7. If it finds no default-client-class-name or client-class configured in a client-entry, it uses the client-class-name from the client named **default** (if found).

## Processing Order to Determine Selection Tags

The order in which the server uses the possible sources to determine selection tags (it uses the first nonnull source) is as follows:

1. Selection tags in the extension environment dictionary.
2. If it finds a real client-entry in the database, it uses the client-entry *selection-tags*. (To prevent this unnecessary database read, enable the *skip-client-lookup* DHCP server attribute; see Skipping Client Entries for Client-Classing, on page 12.)
3. Selection tags in the client-class.
4. If you map an available RADIUS framed-pool value to a tag (by using **dhcp set map-radius-pool-name=map-as-tag**), it uses that tag.
5. If you map an available RADIUS class value to a tag (by using **dhcp set map-radius-class=map-as-tag**), it uses that tag.
6. If you map an available *dhcp-user-class-id* DHCP option (77) to a tag (by using **dhcp set map-user-class-id=map-as-tag**), it uses that tag.

Next, the server could append one of the following to the list of selection tags (if any):

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

## Troubleshooting Client-Classes

To troubleshoot a client-class, enable client-class logging using the *log-settings* attribute on the Edit DHCP Server page of the web UI, or **dhcp set log-settings=***setting* in the CLI, then reload the DHCP server (if in staged dhcp edit mode). The recommended settings are:

- **client-detail**—Logs a single line at the end of every client-class client lookup operation. This line shows all the data found for the client as well as the data that was found in the client-class.
- **client-criteria-processing**—Logs a message whenever the server examines a scope or prefix to find an available lease or to determine if a lease is still acceptable for a client that already has one.
- **ldap-query-detail**—Logs messages whenever the DHCP server initiates a lease state entry creation to an LDAP server, receives a response from an LDAP server, or retrieves a result or error message from an LDAP server.
- If the problem could be related to your LDAP server, also enable the LDAP *can-query* setting.

These logs will help answer these questions:

- Is the server reading the client entry from the expected database?

  The server can read the client entry from LDAP or CNRDB (the Cisco Prime Network Registrar internal database). The *client-detail* log shows you from where the server is reading the client entry.

- Is client-class enabled?

  If enabled but you are getting unexpected results, verify from which database is your Cisco Prime Network Registrar server reading clients. Is it reading from LDAP or CNRDB? The *ldap-query-detail* log tells you if it is reading from LDAP. If not, enable the DHCP *use-ldap-client-data* property.

**Note** Using LDAP requires configuring the LDAP server for queries. Enable the LDAP *can-query* attribute. You also must configure the DHCP server to use LDAP for queries.

- Is the server providing clients the right data, but you see the wrong results from that data (for example, clients are not receiving the expected IP addresses)?

  Verify the explicit relationships on your network. The *client-criteria-processing* log shows from which scopes or prefixes the server is getting addresses. If it does not get addresses from the expected sources, explicit relationships might be incorrectly defined. A scope that you thought was a secondary scope might not be defined that way.

- In Expert mode, did you set the include and exclude criteria for selection tags properly?

  If you define a series of selection tags to include, the tags of a scope or prefix must match those of the client. In Expert mode, you can also use a *selection-criteria-excluded* attribute on the client-class to exclude selection tags. If you define a series to exclude, a scope or prefix must have none of these tags defined so that the client can get configuration parameters from it. Avoid complex inclusion and exclusion scenarios as you begin working with selection tags.

# Configuring Clients

DHCP client properties include the participating client-class and associated policy for a client, the action to perform, and the inclusion and exclusion criteria for selection tags. A client inherits the properties from its client-class, which you may choose to override or supplement by specifying different properties for the client.

## Local Web UI

**Step 1** From the **Design** menu, choose **Clients** under the **DHCP Settings** submenu to open the List/Add DHCP Clients page.

**Step 2** Click the **Add Clients** icon in the Clients pane to open the Add DHCP Client dialog box, enter the client identity, typically a MAC address, but it can also be a DUID or lookup key. (Note that you can set up the DHCP server to validate the client name as a MAC address by enabling the server attribute *validate-client-name-as-mac*.)

You can also create a client named **default** that does not have a specific client configuration. For example, you can have a client always use its MAC address for its hostname.

**Step 3** Select a client-class name, if desired, from the drop-down list of predefined client-classes.

**Step 4** Click **Add DHCP Client**. This opens the Edit DHCP Client page.

The critical step in creating a client is defining its selection criteria so that you can associate the client with a scope or prefix (unless the selection criteria were already set up for the client-class associated with the client).

Use the *selection-criteria* attribute under the Attribute list (see also Table 1: Selection Tag and Criteria Attributes Used , on page 4). You can enter multiple selection tags by separating them with commas. The values have to match the selection tags set for the desired scope or prefix (see Setting Selection Tags on Scopes and Prefixes, on page 3).

**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** Set other attributes as desired. For example:

- Set the *host-name* attribute to *@no-host-name-option* to provide provisional addresses to unknown clients.
- 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).
- Choose the number of time units (seconds, minutes, hours, days, weeks), or UNIX style date (such as Mar 24 12:00:00 2002) to indicate when the authentication expires, or use **forever**.

**Step 6** Click **Save** at the bottom of the page.

**Step 7** Debug as needed. To debug client errors, set the DHCP log settings to **client-criteria-processing**.

**Step 8** To delete a client, click the **Delete Clients** icon in the Clients pane on the left, and confirm the deletion.

## CLI Commands

To create a client, use **client** *name* **create**. To associate a client-class with the client, use **client** *name* **set client-class-name**=*value*. To set selection criteria for scopes or prefixes, use **client** *name* **set selection-criteria**. To set other attributes, use **client** *name* **set** *attribute*=*value*.

To display client properties, use **client** *name* [**show**]. To display properties for all the clients, use **client list**, or **client listnames** to list just the names. To debug clients, use **dhcp set log-settings=client-detail**. To delete a client, use **client** *name* **delete**.

## Editing Clients and Their Embedded Policies

Editing a client involves the same attributes as creating a client. You can also add and modify an embedded policy for the client so that you can set its policy options. The embedded policy has no properties or DHCP options associated with it until you add them. (See also Creating and Editing Embedded Policies.) The client embedded policy setting is the first priority the DHCP server uses in its policy selection (see DHCPv4 Policy Hierarchy).

### Local Web UI

**Step 1** Create the client.

**Step 2** Select the client from the Clients pane on the List/Add DHCP Clients page to open the Edit DHCP Client page.

**Step 3** Make changes to attribute settings as required.

**Step 4** To add a new embedded policy for the client-class, click **Create New Embedded Policy**. If there is an existing embedded policy that you want to edit, click **Edit Existing Embedded Policy**. Both actions open the Edit DHCP Embedded Policy for Client page. (This page is almost identical to the Edit DHCP Embedded Policy for Client-Class page.)

a) Modify the fields, options, and attributes on the Edit DHCP Embedded Policy for Client page. For example, under the DHCPv4 Options, set the client lease time by choosing **dhcp-lease-time [51]** from the drop-down list, enter a lease interval value in the Value field, then click **Add Option**. If necessary, unset attribute values.

If you want to unset the existing embedded policy, click **Unset** on the Edit DHCP Client page; this resets the button to **Create New Embedded Policy**.

**Step 5** Click **Save**.

### CLI Commands

To see if there are any embedded policy values already set for a client, use **client-policy** *client-name* **show**. To create an embedded policy, use **client-policy** *client-name* **set** *attribute* =*value*.

To set the DHCP options, use one of these commands:

```
nrcmd> client-policy client-name setOption <opt-name | id> value [-blob] [-roundrobin]
nrcmd> client-policy client-name setV6Option <opt-name | id>[.instance] value [-blob]
[-roundrobin]
nrcmd> client-policy client-name setVendorOption <opt-name | id> opt-set-name value [-blob]
nrcmd> client-policy client-name setV6VendorOption <opt-name | id> opt-set-name value [-blob]
```

To set the lease time, use **client-policy** *client-name* **setLeaseTime** *value*.

## Configuring DHCPv6 Clients

You can configure DHCPv6 clients.

### Local Advanced Web UI

From the **Design** menu, choose **Clients** under the **DHCP Settings** submenu to open the List/Add DHCP Clients page. Select an existing client to open the Edit DHCP Client page or click the **Add Clients** icon on Clients pane to add a new client-class, choose the client-class that includes the DHCPv6 attributes that were set (see Configuring DHCPv6 Client-Classes, on page 3), then click **Save**.

**Tip** Disable the *validate-client-name-as-mac* attribute for the DHCP server.

### CLI Commands

Use **client list** or **client** *name* **show** to show the existing clients. To set the client-class name for the client, use **client** *name* **set client-class-name=***value*. Also ensure that the *validate-client-name-as-mac* attribute is disabled for the DHCP server.

## Setting Windows Client Properties

Windows clients support class-based provisioning. You can set certain properties that relate to client-class processing. These are:

- Look up the client entry to determine the default client for client-class processing.
- Map the user class ID to the client-class or selection tag.
- Set whether to append the class ID to the selection tag name.

### Settings in Windows Clients

On the Windows client host, use **ipconfig /setclassid** to set the class ID. If you plan to map this client ID to a client-class or selection tag, it must have the same name. Then confirm by using **ipconfig /showclassid**. For example:

```
DOS> ipconfig /setclassid adapter engineering

DOS> ipconfig /showclassid adapter
```

### Settings in DHCP Servers

You must set Windows client properties in the DHCP server.

Use DHCP server attributes in the local cluster web UI or **dhcp set** command attributes in the CLI to set the Windows client properties for the server. If you set the *skip-client-lookup* attribute to true (the default is false), the DHCP server skips the client entry for client-class processing. (See Skipping Client Entries for Client-Classing, on page 12.) 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 selection tag
- **2**—Map the user class ID to the client-class
- **3**—Append the user class ID to the list of selection tags

## Skipping Client Entries for Client-Classing

You may not want to honor client entries for client-classing to prevent unnecessary database reads. To accomplish this, enable the *skip-client-lookup* DHCP server attribute (**dhcp enable skip-client-lookup** in the CLI).

## Limiting Client Authentication

By default, client entries get unlimited authentication. Using the *authenticate-until* attribute, you can limit authenticating a client entry by specifying an expiration time.

When a client entry is no longer authenticated, the DHCP server uses the *unauthenticated-client-class- name* attribute value for the name of the client-class entry to use in answering this DHCP request. If this attribute is unset, or if there is no client-class entry in it, the DHCP server ignores the request.

The valid client authentication values are:

- **+num unit**—Time in the future, where *num* is a decimal number and *unit* is *s*, *m*, *h*, *d*, or *w* for seconds, minutes, hours, days or weeks, respectively. For example, “+3w” is three weeks in the future.
- **date**—Month, day, 24-hour, and 2-or-4-digit-year. For example, “Jun 30 20:00:00 2002.” Enter the local process time. If the server runs in another time zone, disregard the time zone and use local time instead.
- **forever**—Does not expire the authentication for this client.

An example follows of using the *authenticate-until* attribute to distinguish between clients that are authenticated and those that are not authenticated. After the authentication expires and the client requests another address, the DHCP server assigns the client an address from the unauthenticated scope range:

**Step 1** Create an authenticated and an unauthenticated client-class. Set the selection criteria for each as appropriate.

**Step 2** Create the client and include the authenticate-until expiration time. Set the *client-class-name* and *unauthenticated-client-class-name* attributes as appropriate.

**Step 3** Create the authenticated and unauthenticated scopes, define their address ranges, and tie them to their respective selection tags.

**Step 4** Enable client-class processing for the server.

**Step 5** If necessary, reload the DHCP server.

## Setting Client Caching Parameters

The initial request from a client for an address from a DHCP server often goes through a DHCPDISCOVER-DHCPOFFER-DHCPREQUEST-DHCPACK cycle. This process requires that the server must consult the database twice per request for client data. If the client caching parameters are set, the DHCP server caches client data in memory so that it only needs to consult the database once. Client caching can provide a noticeable performance improvement in systems that store client information in LDAP. Client caching is enabled by default unless you unset the applicable attributes.

You can adjust the maximum cache count and time-to-live (TTL) parameters based on the expected rate of client requests. If you expect an onslaught of requests, you might want to increase the cache count, up to a limit based on your available memory. If you expect a longer request cycle, you might want to increase the TTL. The aim is to have the server consult the client cache once during the request cycle.

To set the limit on the number of entries that the server keeps in the client cache, use the *client-cache-count* attribute on the Edit DHCP Server page, or **dhcp set client-cache-count** in the CLI. By default, the maximum number to cache is 1000 clients. To disable the cache, set the attribute to 0.

The client cache is usually valid for only ten seconds, called the cache TTL. After the TTL expires, the server reads the client information from the database, if necessary. You can adjust the TTL using the *client-cache-ttl* attribute on the Edit DHCP Server page, or **dhcp set client-cache-ttl** in the CLI.

When the client cache count reaches the specified maximum, the server cannot cache any more clients until a client-entry TTL expires, after which it reads from the database and begins caching again.

The DHCP server, by default, caches client data while processing DISCOVER message only. If you want to cache client data during REQUEST (Renew or Rebind) message, you need to set the *cache-client-for-requests* attribute to true. This attribute can be configured on the Edit DHCP Server page, or using **dhcp set cache-client-for-requests** in the CLI. This attribute should be set to true only if the duration between the two REQUEST (Renew or Rebind) messages are lesser than the cache TTL.

## Subscriber Limitation Using Option 82

In many situations, service providers want to limit the number of IP addresses the DHCP server should give out to devices on customer premises. They want these devices to have “real” addresses that the DHCP server provides, but limit their number. One way is to use the client-class to register (or provision) each customer device so that the server issues IP addresses only to devices that are registered in the client-entry database. The major drawback to this approach is that it requires registering every customer device, which involves knowing its MAC address. Service providers often do not want to know about each device, but simply that there are not too many of them per customer.

Another approach is to limit customer devices on a per-subscriber basis on values in the *relay-agent-info* DHCP option (option 82, as described in RFC 3046) that the DHCP relay agent sends in a DHCPDISCOVER message. This option includes data about the port on a switch over which the customer device is attached. In a cable modem scenario, one of the option 82 suboptions usually contains the MAC address of the cable modem when the DHCP request comes from a device attached beyond the cable modem. In general, many devices that generate option 82 data place some values in its suboptions such that the value varies per subscriber on the same upstream device. In some cases, this value is unique across all possible subscribers (such as the MAC address of the cable modem). In others, it can be a port on a switch and thus unique across the other subscribers attached to that switch. However, it might not be unique across all subscribers on the switch.

Using this approach, the network administrator can configure limitations on subscriber use of the DHCP-allocated addresses without seriously impacting other DHCP server capabilities. In many environments, network administrators might want to use option 82 limitation for some class of devices and not others. A key aspect of this support is to allow network administrators to separate the devices for which they want to use option 82 limitation from those for which they do not.

### 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 CNRDB or LDAP). However, there is still a requirement that the specific number to which a subscriber is limited should be configurable and override the default number given to all unregistered subscribers.

**Note** Limitation processing is not currently available for DHCPv6 clients.

At a high level, you can configure subscriber limitation by creating an *expression* that the server evaluates for each incoming packet and returns the name of the client-class where you want the client to go (see Using Expressions for details on the use of expressions.). Each client-class allows specification of a limitation identifier (ID), a key the server determines from the incoming packet and uses in later processing to actually limit the number of devices. The server considers all devices with the same limitation ID (the *limitation-id* property) to come from the same subscriber.

## Typical Limitation Scenario

For example, an incoming packet might be evaluated such that:

1. If the *remote-id* suboption of option 82 matches the client hardware address (*chaddr* ), the subscriber is a cable modem and should be in the **cm-client-class**.
2. If the first six bytes in the *dhcp-class-identifier* option value match the string **docsis**, then the subscriber is a DOCSIS modem and should be in the **docsis-cm-client-class**.
3. If the *user-class* option value matches the string **alternative-class**, then the subscriber should be in the **alternative-cm-client-class**.

## Calculating Client-Classes and Creating Keys

You set the expression that determines the client-class for the *client-class-lookup-id* attribute of the DHCP server, or **dhcp set client-class-lookup-id=***expression* in the CLI. Include simple expressions in the attribute definition or more complex ones in a file referenced in the attribute definition (see Using Expressions).

Clients and client-classes also allow specifying a *limitation-id* value for the client or client-class. The server uses this identifier (ID) value to set the address limit on the number of devices with the identical ID on the same network or LAN segment. If a requesting client oversteps the limit of available addresses for that ID, the server assigns it to an *over-limit-client-class-name* (if set); otherwise, it drops the packet. The *limitation-id*, in effect, defines a subscriber.

## Client-Class Lookup Expression Processing

The initial client-class lookup is to allow you to decide whether the client should participate in some sort of limitation. Configure an expression server-wide with the *client-class-lookup-id* attribute. The server executes this expression on every incoming packet with the goal of determining the client-class of the packet.

The expression should return a string that is the client-class name for the packet, or the distinguishing string <none> indicating that no client-class value was considered for the client request. Returning the <none> string is equivalent to not configuring a *client-class-lookup-id* value and that no client-class processing should occur. If the expression returns null or there is an error evaluating the *client-class-lookup-id*, the server drops the packet (with an accompanying log message).

## Limitation Processing

The DHCP server limits the number of IP addresses allocated to DHCP clients with the same *limitation-id* value in the same network or LAN segment. In cases where the server finds that allocating another address to the client would go over the limit, it places the client packet in the *overflow-client-class* (if any is specified). This allows special handling for clients that are over the configured limit. Handling these clients in some self-provisioning way is one of the benefits of using limitation on the DHCP server instead of in the hardware (should it even be supported).

If there is no over-limit client-class, the server drops a packet where allocating an address for that packet would exceed the allowed *limitation-count* for that *limitation-id*. Note that the server enforces the limitation only in a single network or LAN segment. This is hardly a restriction, because network managers tend to see a single subscriber connecting only over one LAN segment at a time.

Configure the *limitation-count* with an identical *limitation-id* in a DHCP policy. The limitation code searches up the policy hierarchy for the *limitation-count* just as it does for any other policy item. This means that you can configure the *limitation-count* in a client-class embedded or named policy, a scope embedded or named policy, or the system *system_default_policy*.

When you configure a *limitation-id* on a client-class, you thereby signal to pursue limitation processing for the client-class. When you do not configure a *limitation-id*, you thereby signal not to pursue it. When executing the expression to determine the *limitation-id*, if the expression returns null, this signals that limitation processing should occur and to use the *limitation-id* saved in the lease state database.

## Expression Processing for Subscriber Limitation

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

- Looking up a client-class
- Creating the key to limit clients of the same subscriber (the *limitation-id*)
- Creating the key to look up in the client-entry database (the *client-lookup-id*)

## Configuring Option 82 Limitation

**Step 1** If you do not register clients explicitly, do not enable client-class as a DHCP server property when using option 82 data.

**Step 2** Determine if you want to limit some clients and not others. If you want to limit some clients:

a) Find some method to distinguish these clients from the others, based on some values contained in the DHCP requests from each class of clients.
b) Determine the names of the client-classes into which you want to put the clients that are not limited, and the selection tag and scope or scopes you want to use for these unlimited clients.

**Step 3** Decide if you want to put clients that are over-limit into a different client-class or just drop their packets. If you want to put them in an over-limit client-class, determine the client-class name and the selection tag and scope or scopes into which you want to put the over-limit clients.

**Step 4** Determine the client-class into which you want to put clients that you intend to limit and the selection tags and scope or scopes you want to use for these clients.

**Step 5** Create all these selection tags, client-classes, and scopes.

**Step 6** Configure the *limitation-count* in a policy, probably the named policy associated with the client-class for the clients to limit.

**Step 7** Write the expression to separate the incoming clients into those to be limited and those not to be limited. Configure it on the DHCP server by setting the *client-class-lookup-id* attribute.

**Step 8** Write the expression to determine the limitation ID for the devices to limit, and configure it on the client-class for clients to limit by setting the *limitation-id*.

## Lease Renewal Processing for Option 82 Limitation

Only packets that the DHCP client broadcasts arrive at the server with option 82 data attached. The BOOTP or DHCP relay agent adds the option 82 data in the first upstream router from the client device. A DHCPRENEW packet is unicast to the server and arrives without option 82 data. This can pose a problem when trying to configure the server for subscriber limitation.

There are generally two approaches to take when dealing with renewals:

- Place all packets that do not have option 82 data in a client-class with no associated selection tags. This is equivalent to a wildcard selection and means that any packet with no option 82 data is accepted.
- Place a DHCPRENEW in the same client-class as you would place a packet that has option 82 data, and have its *limitation-id* evaluate to null. This is a signal that when checking for limitation, the DHCP server should use a previously stored *limitation-id* instead of one from the packet.

Both approaches work. The second one can be more secure, but in practice, it is not much better than the first. This is because you have to use an IP address for the DHCP server to respond to a DHCPRENEW, and most clients would not do this unless the server loses some of its state. In this case, you would want it to give the client the address. In the case of a malicious client, it would still have to use the address to get the server to give the address to the client, thereby limiting the exposure for this case.

## Administering Option 82 Limitation

Whenever a client is involved in limitation because of its inclusion in a client-class with a *limitation-id*, the limitation ID used appears in the DHCP log file whenever client data logging occurs as "... LID: *nnn* :*nnn* :*nnn* ...." The data is logged only for clients with active leases that are currently occupying one of the *limitation-count* counts.

You can determine all the clients using a *limitation-id* in a subnet. In the CLI, use **dhcp limitationList**:

```
nrcmd> dhcp limitationList ipaddr [limitation-id] show
```

If you specify both the *ipaddr* and *limitation-id*, the *ipaddr* value, the server uses it just like a *giaddr* to determine the subnet. You can use any IP address that could appear in any scope (primary or secondary) for the network to specify a subnet. If you specify only the *ipaddr*, it must be an address that the DHCP server serves, and the command returns all the clients and corresponding leases they use.

If a client is denied service due to a *limitation-count* overflow, a message such as this appears in the DHCP server log file:

```
Warning Server 0 05646 Could not add Client MAC: '1,6,01:02:03:04:0c:03'  with
limitation-id: 01:02:03 using Lease: 10.0.0.23, already 3 Clients with that id.
No over-limit client class specified! Dropping packet!
```

You can determine which clients are extended beyond the *limitation-count*, thus causing a denial of service for any new client, by using **dhcp limitationList**. The *ipaddr* value in the command should be the "using Lease:" value, and the limitation-id should be the "limitation-id:" value, in the log file. Using the log file example, the command would be:

```
nrcmd> dhcp limitationList 10.0.0.23 01:02:03 show
```

## Troubleshooting Option 82 Limitation

There are several ways that you can debug limitation support. First, you might want to turn on packet tracing by setting the DHCP server debug value to **VX=1** (or using **dhcp setDebug VX=1**). (The **VX=0** debug value disables packet tracing.) Then, you probably want to enable client-class debugging by adding *client-criteria-processing* and *client-detail* to your log settings.

There is also a server-wide expression trace level, *expression-trace-level*, that you can set to various levels. Setting it to 6 gives you a details trace of every expression evaluation. This can take a bit of space in the log, and slows down the server considerably as well, but is invaluable in the process of getting familiar with expression evaluation. See Debugging Expressions.

When things seem to be going strangely, or when submitting log files to report a problem, it is important to enable some additional tracing by setting the DHCP server debug value to **QR57=9** (or using **dhcp setDebug QR57=9**. (The **QR57=0** debug value disables this tracing). Note that the Q and R are both uppercase. The Q is client-class debugging and the R is response debugging (required to get the flow of control clear in the log). The 5 is expression processing and the 7 is *client-class-lookup* processing. This generates a page or so of output for each packet, which will help you understand what is going on inside the server.

## Expression Examples

See Using Expressions to Limit IP Addresses Leased to Subscribers.

# Configuring Cisco Prime Network Registrar to Use LDAP

The Lightweight Directory Access Protocol (LDAP) provides directory services to integrate Cisco Prime Network Registrar client and lease information. By building on your existing standard schema for objects stored in LDAP directories, you can handle information about DHCP client entries. Thus, instead of maintaining client information in the DHCP server database, you can ask the Cisco Prime Network Registrar DHCP server to issue queries to one or more LDAP servers for data in response to DHCP client requests, or write lease data to an LDAP server.

Cisco Prime Network Registrar uses the OpenLDAP client available with Linux.

## 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 Cisco Prime Network Registrar does not depend on specific LDAP object classes or schema:

- You can store DHCP client information in unused attributes. For example, you could use the *givenname* attribute to hold the DHCP *client-class name* value.
- You can add new attributes to an object class without altering your LDAP schema if you disable LDAP schema checking. For example, you could add the *client-class-name* attribute to the organizational person object class.
- You can create a new object class and define the appropriate attributes. For example, you can create the DHCP client object class and define the client attributes that you want to use.

When you configure the DHCP server to read from LDAP, a query dictionary tells the server which LDAP attributes to query for. The server converts the resulting data into DHCP client data attributes.

**Tip** You can configure Cisco Prime Network Registrar to generate SNMP traps when an LDAP server stops responding or resumes responding to requests from the DHCP server.

## Adding and Editing LDAP Remote Servers

You must add a remote LDAP server so that you can begin using the LDAP services.

### Local Advanced Web UI

From the **Deploy** menu, choose **LDAP** under the **DHCP** submenu to open the List/Add LDAP Remote Servers page. Click the **Add LDAP** icon in the LDAP pane to open the Add DHCP LDAP Server dialog box. To edit the remote server, select the LDAP in the LDAP pane to open the Edit LDAP Remote Server page.

On this page, you must specify at least the name and fully qualified domain name or IP address (IPv4 or IPv6) of the LDAP server. The username and password are required for successful operation.

**Note** The Query Settings and the Create Settings are used in the local for DHCP lease creation.

### CLI Commands

Use **ldap** *name* **create** *domain-name*. For example:

```
nrcmd> ldap ldap-1 create ldap.example.com
```

You can also use **ldap server** IP address (IPv4 or IPv6). For example:

```
nrcmd> ldap ldap-1 create 192.0.2.1
nrcmd> ldap ldap-1 create 2001:DB8:1::1
```

## LDAP over TLS

Before Cisco Prime Network Registrar 11.2, the communication between LDAP server and CPNR happens in clear-text. Starting Cisco Prime Network Registrar 11.2, TLS support is added for this communication based on RFC 4511 and RFC 4513 (StartTLS Operation). Both Server and Client Identity Check are supported along with data encryption. In Cisco Prime Network Registrar, the LDAP server listens on configurable port 389 for TLS.

**Note** LDAPS (LDAP over SSL) is not supported.

The server looks for managed certificates for its component type (that is, certificates of type=dhcp). When enabling TLS, set the *server-ca-certificate* and *client certificate* attribute. To know more about the managed certificates, see the *"Certificate Management" section in Cisco Prime Network Registrar 11.2 Administration Guide*.

LDAP Remote Server configuration has option to specify LDAP Server CA Certificate. The Server Identity Check happens based on the steps that follows:

- If LDAP Server CA Certificate is configured, Server Identity Check happens during TLS handshake using specified certificate. If certificate is not valid, connection fails.
- If LDAP Server CA Certificate is not configured, Server Identity Check does not happen during TLS handshake. CPNR tries to establish TLS connection without LDAP server identity check.

LDAP Remote Server configuration has option to specify LDAP Client Certificate (and Private Key). The Client Identity Check happens based on the steps that follows:

- If LDAP Client Certificate is configured, Client Identity Check happens during TLS handshake. If certificate and/or private key is not valid, connection establishment depends on LDAP server configuration (olcTLSVerifyClient/TLSVerifyClient).
- If LDAP Client Certificate is not configured, Client Identity Check does not happen during TLS handshake. CPNR tries to establish TLS connection without client identity check and connection establishment depends on LDAP server configuration (olcTLSVerifyClient/TLSVerifyClient).
- To perform Client Identity Check, both Client Certificate and Private Key are required. Hence configuring a Certificate object, without either 'certificate-contents' or 'key-contents', as LDAP Client Certificate fails.

**Note** Cisco Prime Network Registrar does not support a command for generating self-signed certificates. However, they can be generated using readily available command line tool like openssl. For example:

```
# openssl req -new -x509 -days 365 -nodes -out public.pem -keyout private.pem
```

For more information on generating the certificates for server and client and configuring the LDAP Server Certificate and other TLS settings, see the Certificate generation and LDAP Server Certificates Configuration and TLS settings in the appendix section.

*Table 2: TLS attributes in the DHCP LDAP server*

| Attribute | Description |
|---|---|
| TLS | Enables or disables TLS support (using Start TLS mechanism defined in RFC 4511 and 4513) for communication between DHCP and LDAP Server. |
| LDAP Server CA Certificate (*server-ca-certificate*) | Specifies the LDAP server CA certificate for Server Identity Check. If it is configured, server certificate is requested during connection establishment. If no certificate is provided by LDAP server, or a bad certificate is provided, the session is immediately terminated. If server CA certificate is not configured, DHCP server does not request or check any LDAP server certificate. The specified certificate should be a managed DHCP certificate. |
| LDAP Client Certificate (*client-certificate*) | Specifies the LDAP client certificate for Client Identity Check during TLS session establishment. The specified certificate should be a managed DHCP certificate, with both certificate and private key configured. |

### Local Advanced Web UI

To enable TLS support for the LDAP server, do the following:

**Step 1** From the **Deploy**menu, choose **LDAP** under the **DHCP** submenu to open the List/Add LDAP Remote Servers page.

**Step 2** Click **Add DHCP LDAP Server** icon in the LDAP pane to open the Add DHCP LDAP Server dialog box.

**Step 3** Enter the required fields to create the LDAP object:

- name: An arbitrary name used to refer to an individual server.
- host name: Sets the hostname, IPv4 or IPv6 address of the server to connect to.

**Note** If CA certificate is generated using a hostname as common name, we must give the host name only in this field.

**Step 4** Click **Add DHCP LDAP Server**.

**Step 5** To edit the LDAP remote server, select the LDAP remote server object created in the LDAP pane to open the **Edit LDAP Remote Server** page.

**Step 6** In the **Edit LDAP Remote Server** page, under the **TLS Settings** section, enable the *TLS* attribute by selecting the **enabled** option and set the LDAP Server CA certificate and LDAP Client certificate.

**Step 7** Click **Save** to save the changes.

## Configuring DHCP Client Queries in LDAP

You can configure and unprovision DHCP client queries, and configure embedded policies, in an LDAP client entry.

### Configuring DHCP-Server-to-LDAP Client Queries

To enable the DHCP server to query your LDAP server for client data, perform the following steps. Like local client entries, LDAP client entries are keyed by the client MAC address.

**Note** When connecting to an LDAP server, use the *distinguished name* (*dn*) of the user. It uniquely identifies an object in the LDAP schema, and is like a unique key in a database or a fully qualified path name for a file. For example, a dn for a person might be dn: cn=Beth Jones, ou=Marketing, o=Example Corporation. In this company, there may be many people named Beth and many people named Jones, but no one else named Beth Jones works in Marketing at Example Corporation.

**Step 1** Supply a hostname for the LDAP server. On the Add LDAP Remote Server page, enter a value in the name field. In the local CLI, use this command:

```
nrcmd> ldap ldap-1 create ldap.example.com
```

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

**Step 2** Configure the connection credentials. Use the distinguished name (*dn*) for the user. Enter a value in the username field. In the CLI, use this command, for example:

```
nrcmd> ldap ldap=1 set username="cn=joe,o=Example Corp,c=US" password=access
```

**Step 3** Set the search path (and, if necessary, the search scope). The path is a point in the directory from which to start searches. If the search scope is:

- SUBTREE, the server searches all the children of the search path.
- ONELEVEL, the server searches only the immediate children of the base object.
- BASE, the server searches only the base object itself.

This example sets the base of the search to be the organization Example Corp and the country US, with a subtree search scope. Enter a value in the search-path field. In the CLI, use this command, for example:

```
nrcmd> ldap ldap-1 set search-path="o=Example Corp,c=US" search-scope=SUBTREE
```

**Step 4** Set the search filter to be the attribute for which DHCP substitutes the client's MAC address (for DHCPv4) or DUID (for DHCPv6). In this example, the attribute is the common name (*cn*). Enter a value in the search-filter field. In the CLI, use this command, for example:

```
 nrcmd> ldap ldap-1 set search-filter=(cn=%s)
```

**Step 5** Configure a query dictionary that contains all the LDAP-to-DHCP mappings. Use **ldap** *servername* **setEntry** to set these mappings.

**a.** Retrieve the DHCP surname from the *sn* LDAP attribute:

```
nrcmd> ldap ldap-1 setEntry query-dictionary sn=host-name
```

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

```
nrcmd> ldap ldap-1 setEntry query-dictionary givenname=client-class-name
```

**c.** Retrieve the domain name from the *localityname* LDAP attribute:

```
nrcmd> ldap ldap-1 setEntry query-dictionary localityname=domain-name
```

**d.** If you need to unset any of the entries, use **ldap** *server* **unsetEntry** *attribute key*. You can also check any of the settings using **ldap** *server* **getEntry** *attribute key*.

**Step 6** Enable queries for the LDAP server. This example enables queries for *myserver*. Set the *can-query* attribute to enabled. In the CLI, use this command:

```
 nrcmd> ldap ldap-1 enable can-query
```

**Step 7** Enable client-class processing for the DHCP server. On the Edit DHCP Server page, set the *client-class* attribute to enabled. In the CLI, use this command:

```
 nrcmd> dhcp enable client-class
```

**Step 8** Enable the DHCP server to use LDAP for client entry queries. On the Manage DHCP Server page, set the client-class attribute to enabled. In the CLI, use this command:

```
nrcmd> dhcp enable use-ldap-client-data
```

**Step 9** If you have more than one LDAP server configured, you can also set them to operate in round-robin or failover mode:

- **round-robin**—The LDAP servers’ preference values are ignored and all servers that are configured to handle client queries and accept lease state updates are treated equally.

- **failover**—The DHCP server uses the active LDAP server with the highest preference (lowest preference number). If the preferred server loses its connection or fails, the DHCP server uses the next LDAP server of lower preference (increasing preference number). If the preference values are the same (or not set), the DHCP reverts to round-robin mode with these servers.

Set the LDAP server mode by setting the ldap-mode on the Edit DHCP Server page. LDAP failover mode actually performs preferential load balancing. The DHCP server assesses the LDAP connection and error states and how fast the LDAP server responds. In an optimal state, the DHCP server uses the LDAP server with the highest assigned preference (lowest preference number). In a less-than-optimal state, the DHCP server uses the next LDAP server of lower preference (increasing preference number). If the preference values are the same (or unset), the DHCP server reverts to round-robin mode.

In the CLI, use **dhcp set ldap-mode** to set the mode, and **ldap** *server* **set preference** to set the server preferences; for example:

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

Note also that, depending on how many threads you have open, as set by using the *connections* attribute (see Recommended Values for LDAP, on page 30) between the DHCP and LDAP servers, the DHCP server opens only as many threads as it can before the *query-timeout* expires. The LDAP server might be processing these threads, but it is not servicing the request, because the failover server has now taken over.

**Step 10** Enable the DHCP server to use LDAP for client entry queries. On the Manage DHCP Server page, set the *client-class* attribute to enabled. In the CLI, use this command:

```
 nrcmd> dhcp enable use-ldap-client-data
```

**Step 11** Show or list the LDAP configuration. Go to the List/Add LDAP Remote Servers page. In the CLI, use:

```
nrcmd> ldap ldap-1
nrcmd> ldap list
nrcmd> ldap listnames
```

**Step 12** Reload the DHCP server.

---

**Note** The DHCP server typically replaces **%s** with the client MAC address (for DHCPv4) or DUID (for DHCPv6). However, you can use other client-specifiers. When using other client-specifiers (such as generated by an extension), take care to assure that the string cannot be used to do LDAP injection. This means that, you must ensure that the following characters cannot be inserted by data sent from the client or are properly escaped if that is possible:

comma (,), backslash character (\), pound (hash) sign (#), plus sign (+), less than (<), greater than (>), semicolon (;), double quote ("), equal sign (=), and leading or trailing spaces

In some cases, other characters may also be an issue (check with your LDAP server or RFC 4514). Thus, in cases where data from a received packet is used, this could be an issue. The DHCP server will not alter the provided string. It assumes that the provided string is safe to use as is.

## 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 in Configuring DHCP-Server-to-LDAP Client Queries, on page 20 of the preceding section so that the LDAP server does not return a client entry containing a specified attribute with a value.

If you want to unprovision the LDAP entry *givenname*, configure the search filter accordingly. For example:

```
nrcmd> ldap ldap-1 set search-filter=(&(cn=%s)(!(givenname=unprovision)))
```

Whenever the *givenname* attribute in the LDAP client entry is set to the "unprovision" string, the LDAP server does not return the client entry to the DHCP server. In other words, the DHCP server treats the client as if it has no LDAP client entry. This procedure has no measurable performance impact on either the DHCP or the LDAP server.

## Configuring Embedded Policies in LDAP

**Step 1** Configure an LDAP server for the DHCP server, naming it myserver, for example.

**Step 2** Map the LDAP attribute that you want the DHCP server to interpret as the embedded policy to the internal embedded-policy property. This example maps the businessCategory LDAP attribute:

```
nrcmd> ldap myserver setEntry query-dictionary businessCategory=embedded-policy
```

**Step 3** Add a string to the LDAP attribute that the DHCP server can interpret as an embedded policy. The most practical way to determine what this string should look like is to create a dummy client in the Cisco Prime Network Registrar database and extract data from the client embedded policy setup. Note that this dummy client will never be used, because you are using LDAP, and you can subsequently delete it. Have the embedded policy include the option data types that you need.

**a.** For example, create an embedded client policy for dummy client 1,6,00:d0:ba:d3:bd:3b. Add some reply options and a multivalue option (routers) with an IP address data type:

```
nrcmd> client 1,6,00:d0:ba:d3:bd:3b create
nrcmd> client-policy 1,6,00:d0:ba:d3:bd:3b set v4-reply-options=routers
nrcmd> client-policy 1,6,00:d0:ba:d3:bd:3b setOption routers 1.2.3.4,5.6.7.8
nrcmd> save
```

**b.** Get the client embedded policy data so that you can display the values:

```
nrcmd> client 1,6,00:d0:ba:d3:bd:3b get embedded-policy
100 Ok
embedded-policy="((ClassName Policy)(name client-policy:00:d0:ba:d3:bd:3b)(option-list [((ClassName
 Option)(number 3)(option-definition-set-name dhcp-config)(value
01:02:03:04:05:06:07:08))])(v4-reply-options [routers ])"
```

**c.** Copy what is between the quotes in the client output in the previous substep and paste it in for the definition of the businessCategory LDAP attribute:

```
businessCategory:((ClassName Policy)(name client-policy:00:d0:ba:d3:bd:3b)(option-list [((ClassName
 Option)(number 3)(option-definition-set-name dhcp-config)(value
01:02:03:04:05:06:07:08))])(v4-reply-options [routers ])
```

**d.** Use the syntax as a model for each new embedded policy entry in LDAP. To see how other option data types appear in the LDAP string, add these options to the client or create further dummy clients with them. Once you extract the data, you can delete the dummy client:

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

### Configuring Embedded Policies in LDAP (with multiple option definitions)

Here is another example with multiple option definitions:

**Step 1** Create a dummy client 1,6,00:d0:ba:d3:bd:3b and an embedded policy attached to that client, with the following options and values:

```
3 routers 10.1.1.1,10.2.1.1
66 tftp-server tftp-server.com
67 bootfile device-boot-file.txt
```

**Step 2** Save the changes to the embedded policy, save the client, then extract the following output string into an LDAP client configuration:

```
nrcmd> client 1,6,00:d0:ba:d3:bd:3b get embedded-policy
100 Ok
embedded-policy="((ClassName Policy)(name client-policy:00:d0:ba:d3:bd:3b)(option-list [((ClassName
 Option)(number 3)(option-definition-set-name dhcp-config)(value 0a:01:01:01:0a:02:01:01))((ClassName
 Option)(number 66)(option-definition-set-name dhcp-config)(value
74:66:74:70:2d:73:65:72:76:65:72:2e:63:6f:6d))((ClassName Option)(number 67)(option-definition-set-name
 dhcp-config)(value 64:65:76:69:63:65:2d:62:6f:6f:74:2d:66:69:6c:65:2e:74:78:74))])"
```

## Configuring DHCP LDAP Update and Create Services

You can configure the Cisco Prime Network Registrar DHCP server to write lease and client data to an LDAP server. The DHCP server can use the client data when responding to DHCP client requests, through the use of the *query* configuration. You can configure the DHCP LDAP service to copy lease state data to attributes on client objects in the LDAP server. The DHCP server converts the lease state data to string form, and uses an update dictionary to map the DHCP data values to the LDAP attributes.

Each time the lease state changes, the DHCP server writes the change to the LDAP server that you configured to store the data. The lease data that the DHCP server writes to LDAP is “write-only” in that it is a copy of the authoritative data in the lease state database.

## Lease State Attributes

You can store any of these attributes about the lease state information in your LDAP server:

- *address*—IP address of this lease.
- *client-dns-name*—Name the DHCP server attempted to enter into the DNS server for this client.
- *client-domain-name*—Domain into which to put the client name.
- *client-flags*—A variety of flags relating to the client.
- *client-host-name*—DNS name that the client requested the DHCP server to place in the DNS server.

- *client-id*—Client-id specified by the client, or one synthesized by the DHCP server for this client.
- *client-mac-addr*—MAC address that the client presented to the DHCP server.

**Note** Although the MAC addresses in LDAP have to be formatted exactly the way they are formatted by Cisco Prime 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 Cisco Prime 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 Cisco Prime Network Registrar. In addition, the *lease-renewal-time* attribute may not be present if the *save-lease-renewal-time* property is disabled through DHCP. Similarly, the *vendor-class-identifier* property may not be present if the *save-vendor-class-id* property is disabled through DHCP, using the CLI.

## Configuring DHCP to Write Lease States to LDAP

To have DHCP write lease state updates to LDAP:

**Step 1** Choose the LDAP lease state update scheme.

**Step 2** Add entries to the directory or modify existing entries to store the lease state information. You may need to extend entries through the addition of attributes or custom object classes.

**Step 3** Configure Cisco Prime Network Registrar to perform the updates.

Given the flexibility of directories, there are many different ways in which you could choose to store a copy of lease state attributes in a directory. For example, you could choose to store the lease state data as part of an existing entry, or you could store the lease state data independently.

## Storing Lease State Data as Part of Existing Entries

You can store lease state data as part of an existing entry. It is even possible to store the client entry, lease state, and employee data in the same entry. As part of the setup for this method, you must decide how you want to store the lease data attributes. You can store data attributes using these methods:

- Map attributes from the entry
- Add attributes to the entry
- Extend the entry by creating a new object class

The advantage is that lease data is stored directly with other client information. The disadvantage is that there are scenarios, albeit unlikely, related to client-class and reservations that could result in stale data being in the directory for a short period of time when the server moves a client off a lease.

**Note** If the lease whose state is being updated does not have a client, it will not have an associated MAC address. This situation occurs when a client gets a lease, and then is moved off that lease by client-class processing. It can also occur when a client has a pre-existing lease and a reservation for a different lease in the same LAN segment. When the reserved lease is available, the server moves the client off its existing lease and onto the reservation. Both of these transfers result in an LDAP update for the old lease without a client MAC address. This is generally not a problem, because the update for the new lease (which has an associated MAC address) should come through.

Also, this method requires two LDAP interactions to write the lease information. When updating lease state information, the DHCP LDAP service contacts the directory twice because when updating an entry it is not enough just to know how to find the entry. You must specifically know the *dn* of the entry.

The DHCP LDAP service first finds the appropriate entry in the directory by using one of the lease state attributes that you chose (preferably the MAC address) as the search criteria. This is necessary because none of the lease state attributes is part of the *dn* of the entry. When the DHCP LDAP service locates the entry, the *dn* is returned. The DHCP LDAP service then updates that same entry with the appropriate information. For an example how to use this method, see Configuring LDAP State Updates, on page 27.

## Storing Lease State Data Independently

You can store lease state data by IP address in its own entries. This method results in a copy of the server lease database in a directory, and is the most straightforward way to configure the database. As part of the setup for this method, create new entries for each IP address that the server can serve. The advantage to this method is that there are no scenarios in which the lease state data in the directory will be stale. The disadvantage is that lease data is not stored directly with other associated client information.

To update the lease state information, the DHCP LDAP service contacts the directory service once. When performing the update, the service uses the IP address to construct the *dn*.

## Using LDAP Updates

There are two ways you can use the LDAP update feature:

- Keep track of clients that use LDAP client entry information and to associate some of the attributes of that LDAP host with lease state attributes.
- Create and update objects that can be located by their IP address. When Cisco Prime Network Registrar creates these objects, it can make a level of LDAP objects that matches (or is) the DHCP server lease state.

When using Cisco Prime Network Registrar, you should be aware that:

- The DHCP server only reads from a single object and writes to a single object. You can use separate objects to hold the client entry data read and the lease state date written, but Cisco Prime 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.
- LDAP attributes must either come preconfigured in the LDAP schema at server installation or be created by some other means outside Cisco Prime Network Registrar.

## Configuring LDAP State Updates

There are two options available for performing a lease state update to an LDAP server:

- ***update-search-path***—The DHCP server first queries to locate the *dn* for an update.
- ***dn-format***—The server is provided with the *dn* for an update. In other words, the DHCP performs a direct update without having to query before an update.

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

The following example illustrates the first option, *update-search-path*. It shows what to do when the distinguished name (*dn*) of an LDAP object cannot be constructed from data that is available in the lease state. The DHCP server creates an LDAP query based on the *update-search-xxx* information, locates the LDAP object, and uses its *dn* to issue an LDAP update.

The example shown in the table below assumes that you are using the standard LDAP organizational person object class attributes to hold lease update data.

*Table 3: LDAP-to-DHCP Mapping Example*

| Attribute | DHCP Lease Entry Mapping |
|---|---|
| *uid* | address (IP address) |
| *carlicense* | state (lease state) |

**Step 1** Tell DHCP about the LDAP server by supplying the server hostname in the LDAP configuration.

**Step 2** Configure the credentials to use when connecting to the LDAP server. This CLI example sets the administrator to joe and his password to access. Use the distinguished name (*dn*) for the user:

```
nrcmd> ldap myserver set username="cn=joe,o=Example Corporation,c=US" password=access
```

**Step 3** Configure the *update-search-path* attribute, which is the starting point in the directory for the objects that the DHCP server will update. You can also set the update search scope. This CLI example sets the search path to begin at the organizational unit (ou) IT, the organization Example Corporation, and country US. The update search scope is set to SUBTREE:

```
nrcmd> ldap myserver set update-search-path="ou=IT,o=Example Corp,c=US"
update-search-scope=SUBTREE
```

**Step 4** Set the ID of the attribute you want to use to search for the LDAP object that will be updated. This CLI example sets the search attribute to be the client MAC address:

```
nrcmd> ldap myserver set update-search-attribute=client-mac-addr
```

**Step 5** Configure a filter expression into which the *update-search-attribute* attribute should be formatted. This expression must contain a "%s," which indicates where the search attribute data should be substituted. Here is a CLI example:

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

**Step 6** Configure the *update-dictionary* attribute, which allows you to identify the LDAP attributes that you want set with the values of the corresponding lease state attributes. This example specifies that the LDAP UID should be updated to contain the IP address, and that the *carlicense* attribute should be updated to contain the DHCP lease state information. Using the CLI:

```
nrcmd> ldap myserver setEntry update-dictionary uid=address carlicense=state
```

**Step 7** Enable updates for the new LDAP server. Here is a CLI example:

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

**Step 8** Reload the DHCP server.

## Option 2: Using the dn-format Option

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

**Step 1** Tell DHCP about the LDAP server by supplying the server hostname in the LDAP configuration.

**Step 2** Configure the credentials to use when connecting to the LDAP server. This CLI example sets the administrator to joe and his password to access. Use the *dn* for the user:

```
nrcmd> ldap myserver_option2 set username="cn=joe,o=Example Corporation,c=US"
password=access
```

**Step 3** Use the *dn-format* string to specify where in the LDAP server database hierarchy you want to begin searching for the update. Here is a CLI example:

```
nrcmd> ldap myserver_option2 set dn-format="cn=\"%s\",ou=IT,o=Example Corp,c=US"
```

**Step 4** Set the *dn-attribute* attribute to which you want the *dn-format* string to refer. This CLI example sets the *dn-attribute* to be the client MAC address:

```
nrcmd> ldap myserver_option2 set dn-attribute=client-mac-addr
```

**Step 5** Specify the entries to be updated. Using the CLI:

```
nrcmd> ldap myserver_option2 setEntry update-dictionary uid=address carlicense=state
```

**Step 6** Enable the *can-update* attribute. Here is a CLI example:

```
nrcmd> ldap myserver_option2 enable can-update
```

**Step 7** Reload the DHCP server.

## Configuring LDAP Entry Creation

This section explains how to create LDAP entries. LDAP entry creation provides the ability to locate entries and update them with current lease information. Entries are created only if a state update operation fails because it cannot locate an entry.

After performing the steps in the previous example, follow these steps in the CLI:

**Step 1** Set the *dn-attribute* property for the LDAP server for the lease object attribute, such as the *client-mac-addr* field, and set the *dn-format* string. Here is a CLI example:

```
nrcmd> ldap myserver set dn-attribute=client-mac-addr dn-format="cn=\"%s\",ou=IT,o=Example Corp,c=US"
```

This step is required only if you configure the lease state updates using the *update-search-path* option. (See Option 1: Using the update-search-path Option, on page 27). Skip this step if you configure lease state updates using the *dn-format* string. (See Option 2: Using the dn-format Option, on page 28.)

**Step 2** Specify the *dn* of the entry to be created when combined with the existing *dn-attribute* property. Here is a CLI example:

```
nrcmd> ldap myserver set dn-create-format="cn=\"%s\",ou=IT,o=Example Corp,c=US"
```

The Cisco Prime Network Registrar *client-mac-addr* field uses the form 1,6:*xx*:*xx*:*xx*:*xx*:*xx*:*xx*. Since the comma character is a special separator in LDAP, you must use the \" characters to quote the *dn*.

**Step 3** Using the *create-dictionary* property, establish mappings between LDAP attributes and lease state attributes by entering a series of name=value pairs. The LDAP attributes indicate the entry attributes set to the value of their corresponding lease state attributes. In the CLI:

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

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

nrcmd> ldap myserver setEntry create-dictionary localityname=client-domain-name
```

**Step 4** Using the *create-object-classes* property, specify the object classes to be used when creating the entry. Here is a CLI example:

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

**Step 5** Enable entry creation for the LDAP server myserver. Here is a CLI example:

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

**Note** Enable the *can-update* attribute before you enable the *can-create* attribute. For an example, see Configuring LDAP State Updates, on page 27.

**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=(&(macaddress=%s)(|(crscclassname=Computer)(csrcclassname=Modem)))

nrcmd> ldap LDAP-Write set search-path=csrcprogramname=csrc,o=NetscapeRoot

nrcmd> ldap LDAP-Write set
username=uid=admin,ou=Administrators,ou=TopologyManagement,o=NetscapeRoot

nrcmd> ldap LDAP-Write disable can-query

nrcmd> ldap LDAP-Write enable can-create

nrcmd> ldap LDAP-Write enable can-update

nrcmd> ldap LDAP-Write enable limit-requests

nrcmd> ldap LDAP-Write set connections=2 max-requests=8 timeout=10s
```

This CLI example tunes read (query) operations, which require shorter server processing:

```
nrcmd> ldap LDAP-Read create csrc-ldap password=changeme port=389 preference=1

nrcmd> ldap LDAP-Read setEntry query-dictionary csrcclientclasas=client-class-name

nrcmd> ldap LDAP-Read set
search-filter=(&(macaddress=%s)(|(crscclassname=Computer)(csrcclassname=Modem)))

nrcmd> ldap LDAP-Read set search-path=csrcprogramname=csrc,o=NetscapeRoot

nrcmd> ldap LDAP-Read set
username=uid=admin,ou=Administrators,ou=TopologyManagement,o=NetscapeRoot

nrcmd> ldap LDAP-Read enable can-query

nrcmd> ldap LDAP-Read disable can-create

nrcmd> ldap LDAP-Read disable can-update

nrcmd> ldap LDAP-Read enable limit-requests

nrcmd> ldap LDAP-Read set connections=3 max-requests=12 timeout=4s
```

### Recommended Values for LDAP

The table below shows recommended values for some key LDAP attributes.

*Table 4: Recommended Values for LDAP Attributes*

| Attribute and Value | Description |
|---|---|
| *connections* =5 to 25 | Number of connections that the server should make to an LDAP server. This is primarily a performance tuning parameter. The default is one connection. In some cases, more than one connection can improve overall throughput. The amount depends on the load on the LDAP server. With many applications using LDAP, five connections would be appropriate; with just Cisco Prime Network Registrar using LDAP, 25 would be appropriate. |
| *threadwaittime* =2 | Interval (in milliseconds) at which each LDAP client connection polls for results, if it has outstanding queries or updates. |
| *query-timeout* =3 | Cisco Prime Network Registrar DHCP servers fail over at the *query-timeout* interval if *failover* and *can-query* are set. The default setting is 3 seconds and is recommended (in that it is less than the default 4-second *drop-old-packets* value for the DHCP server, after which time the connection is considered inactive and the LDAP server as “unhealthy”). |
| *timeout* =10 | Number of seconds an LDAP request remains in a connection queue before being declared stale and timing out. Any response the DHCP client receives after the client timeout period is stale. The default is 10 seconds, which is recommended. Cisco Prime Network Registrar DHCP servers fail over at the *timeout* interval if *failover* and *can-update* or *can-create* are enabled. |