Managing Scopes, Prefixes, and Link Templates

This chapter describes how to set up templates for scopes, prefixes, and links.

- Creating and Applying Scope Templates, on page 1
- Creating and Editing Prefix Templates, on page 3
- Creating and Editing Link Templates, on page 6
- Using Expressions in Link Templates, on page 8

Creating and Applying Scope Templates

Scope templates apply certain common attributes to multiple scopes. These common attributes include a scope name based on an expression, policies, address ranges, and an embedded policy option based on an expression (see Using Expressions in Scope Templates, on page 12).

Local Advanced and Regional Web UI

Scope templates you add or pull from the local clusters are visible on the List DHCP Scope Templates page. To get there, from the **Design** menu, choose **Scope Templates** from **DHCPv4** submenu. This functionality is available only to administrators assigned the dhcp-management subrole of the regional central-cfg-admin or local ccm-admin role.

To explicitly create a scope template, click **Add Scope Templates** on the Scope Templates pane. This opens the Add DHCP Scope Template dialog box, which includes the template name. You can also choose an existing policy for the scope template. The other fields require expression values (see the *"Create a Scope Template"* section in the *Cisco Prime Network Registrar 10.0 Administration Guide* that describes these fields).

Related Topics

Using Expressions in Scope Templates, on page 12

Additional Scope Template Attributes, on page 2

Editing Scope Templates, on page 2

Applying Scope Templates to Scopes, on page 2

Cloning a Scope Template, on page 3

CLI Commands

Create a scope template using **scope-template** *name* **create** [*attribute=value* ...]. For example:

```
nrcmd> scope-template example-scope-template create
```

You can also associate a policy with the scope template:

```
nrcmd> scope-template example-scope-template set policy=examplepolicy
```

Additional Scope Template Attributes

The optional additional attributes appear in functional categories. For a description of each attribute, click the attribute name to open a help window. For example, you might want to enable dynamic DNS updates for the scope, or set the main and backup DHCP failover servers.

After you complete these fields, click **Add Scope Template**.

Editing Scope Templates

To edit a scope template, select its name from the Scope Templates pane. The Edit DHCP Scope Template page is essentially the same as the Add DHCP Scope Template page (see Creating and Applying Scope Templates, on page 1) except for an additional attribute unset function. Make your changes, then click **Save**.

In the CLI, edit a scope template attribute by using **scope-template** *name* **set** *attribute*. For example:

```
nrcmd> scope-template example-scope-template set policy=default
```

Applying Scope Templates to Scopes

You can apply a scope template to a scope in a few ways.

Caution Be careful applying a scope template to an existing scope. The template overwrites all the scope attributes with its own, which can have a detrimental effect if the scope is active.

Local Advanced Web UI

- **When a template is applied to a target**—If the scope-template has an embedded policy, it is copied to the scope. This embedded policy may or may not have options. As the entire scope-template's embedded policy is used (if it exists), it will wipe out any existing options in the scope. If the scope-template has no embedded policy, the scope's embedded policy is retained. Next the scope-template's option expression, if any, is evaluated and the options are added to the embedded policy options in the scope (if no embedded policy exists, one is created).
- **While creating a scope, derive its name from the template**—If you set a Scope Name Expression for the scope template (see Using Expressions in Scope Templates, on page 12) on the List/Add DHCP Scope Template page, when you add a scope on the List/Add DHCP Scopes page, omit the name of the scope, but add its subnet and mask, then choose the scope template from the Template drop-down list. Clicking **Add DHCP Scope** creates a scope with a name synthesized from the scope name expression.

If you do not set a scope name expression in the template and apply it to the scope without specifying a name for the scope, you get an error. (Note that Basic mode does not provide this functionality.)

- **After creating a named scope**—On the Edit DHCP Scopes page, scroll to the bottom to find the **Apply Template** button. Choose a preconfigured template from the drop-down list, then click the button. Then click **Save**. (Be aware of the previous warning that the template attributes overwrite the existing ones of the scope.)

CLI Commands

To apply a template to the scope while creating the scope, use **scope** *name* **create** *address mask* [**template**=*template-name*] [*attribute*=*value* ...]. For example:

```
nrcmd> scope example-scope create 192.168.50.0 24 template=example-scope-template
```

To derive the scope name from the template during scope creation, use **scope-template** *name* **apply-to { all** | *scope1* **,** *scope2* ,...}. For example:

```
nrcmd> scope-template example-scope-template apply-to examplescope-1,examplescope-2
```

Cloning a Scope Template

In the CLI, you can also clone a scope template from an existing one by using **scope-template** *clone-name* **create clone**=*template*, and then make adjustments to the clone. For example:

```
nrcmd> scope-template cloned-template create clone=example-scope-template-1 ping-timeout=200
```

Creating and Editing Prefix Templates

You can create prefixes from predefined templates. The attributes you can set for a prefix template are the following (for the expression syntax, see Using Expressions in Prefix Templates, on page 18):

- ***name***—User-assigned name for the prefix template.
- ***description***—Descriptive text for the prefix template.
- ***dhcp-type***—Defines how DHCP manages address assignment for a prefix:
  - **dhcp** (preset value)—Uses the prefix for stateful address assignment.
  - **stateless**—Uses the prefix for stateless option configuration.
  - **prefix-delegation**—Uses the prefix for prefix delegation.
  - **infrastructure**—Uses the prefix to map a client address to a link, when the prefix does not have an address pool.
  - **parent**—Prefix is not used by DHCP. It is used as a container object, to group child prefixes.
- ***policy***—Shared policy to use when replying to clients.
- ***owner***—Owner of this prefix, referenced by name.
- ***region***—Region for this prefix, referenced by name.
- ***prefix-name-expr***—Expression that evaluates to a string value to use for the name of the prefix created. For example, you can have the prefix name prepended by **CM–** if you define *prefix-name-expr* as **(concat "CM-" prefix)**. In the CLI, you would include the expression in a file and point to that file:

```
> type prefix-name.txt
(concat "CM-" prefix)

nrcmd> prefix-template ex-template create prefix-name-expr=@prefix-name.txt
```

- ***prefix-description-expr***—Expression that evaluates to a string value to apply to the description on the prefix created when using the template.
- ***range-expr***—Expression that evaluates to an IPv6 prefix value to create an address range. In the CLI, you must use a file reference. For example:

```
> type subprefix-expr.txt
(create-prefix-range 1 0x1)

nrcmd> prefix-template ex-template set range-expr=@subprefix-expr.txt
```

- ***options-expr***—Expression that evaluates to embedded policy options to create. (Use the **list** function to create multiple options.)
- ***allocation-algorithms***—One or more algorithms the server uses to select a new address or prefix to lease to a client. The available algorithms are:
  - **client-request** (preset to off)—Controls whether the server uses a client-requested lease.
  - **reservation** (preset to on)—Controls whether the server uses an available reservation for the client.
  - **extension** (preset to on)—Controls whether the server calls extensions attached at the **generate-lease** extension point to generate an address or prefix for the client. When you use generate-lease extension point with DHCPv6 failover, the server uses the address or delegated prefix that the extension returns and does not perform a hash on this address or prefix as it does with the randomly generated addresses. If the extension is using some algorithmic method to generate the address or delegated prefix then the extension must be failover aware (extension will be able to determine if failover configuration is enabled and the role of the failover server). For details on extensions, see Using Extensions.
  - **interface-identifier** (preset to off)—Controls whether the server uses the interface-identifier from the client (link-local) address to generate an address; ignored for temporary addresses and prefix delegation.
  - **random** (preset to on)—Controls whether the server generates an address using an RFC 3041 algorithm; ignored for prefix delegation.
  - **best-fit** (preset to on)—Controls whether the server delegates the first, best-fit available prefix; ignored for addresses.

  When the server needs an address to assign to a client, it processes the flags in the following order until it finds a usable address: client-request, reservation, extension, interface-identifier, and random. When the server needs to delegate a prefix to a client, it processes the flags in the following order until it finds a usable prefix: client-request, reservation, extension, and best-fit.

- ***restrict-to-reservations***—Controls whether the prefix is restricted to client (or lease) reservations.
- ***max-leases***—Maximum number of nonreserved leases allowed on the prefix. When a new lease needs to be created, the server does so only if the limit is not exceeded. When the limit is exceeded, the server cannot create or offer new leases to clients. If you also enable SNMP traps, the *max-leases* value also calculates the percentage of used and available addresses.

Note Be sure to set the *max-leases* value to the expected maximum so that the SNMP address traps can return meaningful results.

- ***ignore-declines***—Controls whether the server responds to a DHCPv6 DECLINE message that refers to an IPv6 address or a delegated prefix from this prefix. If enabled, the server ignores all declines for leases in this prefix. If disabled (the preset value) or unset, the server sets to UNAVAILABLE every address or delegated prefix requested in a DECLINE message if it is leased to the client.
- ***deactivated***—Controls whether a prefix extends leases to clients. A deactivated prefix does not extend leases to any clients and treats all addresses in its ranges as if they were individually deactivated. The preset value is false (activated).
- ***expiration-time***—Time and date at which a prefix expires. After this date and time, the server neither grants new leases nor renews existing leases from this prefix. Enter a value in the format **"**[*weekday*] *month day hh* :*mm* [:*ss*] *year***"**; for example, **"Dec 31 23:59 2006"**. The reason for an expiration time is to support network renumbering events. The general idea is a new prefix is added and the old is taken away sometime at or after the *expiration-time*. Clients will be given leases on both prefixes. The server will automatically stop giving new clients leases once the configured valid lifetime before the expiration-time is reached. At this time, new clients will not get a lease on the prefix. Existing clients will continue to be able to use an existing lease, but will get shorter and shorter lifetimes (preferred and valid). The delta between the preferred and valid is always maintained. Thus if the preferred is 1day and the valid 2days, new clients will stop getting leases 2 days before the *expiration-time*, existing clients will continue to be able to renew leases with preferred lifetimes lesser than 1day and valid lifetimes greater than 2days. 1 day before the *expiration-time*, clients will get a 0 preferred lifetime.
- ***free-address-config***—Trap that captures unexpected free address events on the prefix.
- ***reverse-zone-prefix-length***—Prefix length of the reverse zone for ip6.arpa updates. (See Determining Reverse Zones for DNS Updates for details.)
- ***max-pd-balancing-length***—Controls the maximum prefix-delegation prefix length that the failover pool balancing will consider in balancing a prefix-delegation prefix. The default value is 64 and it should never be longer than the longest prefix length allowed for the prefix delegation.
- ***selection-tags***—List of selection tags associated with the prefix.
- ***allocation-group***—Allocation group to which the prefix belongs.
- ***allocation-group-priority***—Priority of the prefix over other prefixes in the same allocation group. The default value is zero.
- ***range-start-expr***—Defines an expression that evaluates to the *range-start* for the prefix.
- ***range-end-expr***—Defines an expression that evaluates to the *range-end* for the prefix.
- ***embedded-policy***—Policy embedded. When the template is applied, this will replace the entire embedded-policy in the prefix.

Local Advanced and Regional Web UI

Step 1 From the **Design** menu, choose **Prefix Templates** under the **DHCPv6** submenu. The List/Add DHCP v6 Prefix Templates page shows the existing templates.

Step 2 Click the **Add Prefix Templates** icon in the **Prefix Templates** pane to open the Add Prefix Template dialog box.

Step 3 Enter the prefix template name and click **Add Prefix Template**.

Step 4 To edit a prefix template, select its name on the Prefix Templates pane. Set the attributes and add expressions for the templates that require expressions (see Using Expressions in Prefix Templates, on page 18).

Step 5 On the Edit DHCP v6 Prefix Template page, edit the template attributes, such as adding a selection tag, assigning a group and setting priorities, then click **Save**.

Step 6 In the regional web UI, you can pull replica prefix templates or push templates to local clusters:

- Click **Pull Data** to open the Select Replica Prefix Template Data to Pull page. Choose a pull mode for the cluster (ensure, replace, or exact), then click **Pull All Prefix Templates**. On the Report Pull DHCPv6 Prefix Template page, click **OK**.
- Click **Push** for a specific template (or **Push All**) to open the Push Data to Local Clusters page. Choose a data synchronization mode (ensure, replace, or exact), move the desired cluster or clusters to the Selected table, then click **Push Data to Clusters**.
- Click **Reclaim** to open the Reclaim Prefix Template page. Move the desired cluster or clusters to the Selected table, then click **Reclaim Data from Clusters**.

CLI Commands

To create the prefix template, use **prefix-template** *name* **create** [*attribute=value* ...]. For example:

```
nrcmd> prefix-template example-prefix-template create [attribute=value]
```

You can set and enable the aforementioned attributes in the usual way, and you can show and list prefix templates. In addition:

- To clone a prefix template, use **prefix-template** *name* **create clone=***name*.
- To apply a template to one or more prefixes, use **prefix-template** *name* **apply-to** {**all** | *prefix* [**,***prefix***,**...]}.
- The prefix-template includes an embedded-policy object. The prefix-template-policy CLI command and the Web UI supports the embedded policy on the prefix-template page.
- When connected to a regional cluster, you can use the following pull, push, and reclaim commands. For push and reclaim, a list of clusters or "all" may be specified.
  - **prefix-template** < *name* | **all** > **pull** < **ensure** | **replace** | **exact** > *cluster-name* [**-report-only** | **-report**]
  - **prefix-template** < *name* | **all** > **push** < **ensure** | **replace** | **exact** > *cluster-list* [**-report-only** | **-report**]
  - **prefix-template** *name* **reclaim** *cluster-list* [**-report-only** | **-report**]

Creating and Editing Link Templates

You can create links from predefined templates. The attributes you can set for a link template are as follows (for the expression syntax, see Using Expressions in Link Templates, on page 8):

- ***name*** —User-assigned name for the link template.
- ***description*** —Description of the link template itself.
- ***policy*** —Shared policy used when replying to clients, as applied to the link.

- ***owner*** —Owner of the link.
- ***region*** —Region for this link.
- ***link-name-expr*** —Expression to define the name of the link once the template is applied.
- ***link-description-expr*** —Expression to define the description on the link once applied.
- ***prefix-expr*** —Expression to create the list of associated prefixes once the template is applied. For example, you can specify creating prefixes based on defining *prefix-expr* as **@link-prefix-expr.txt** to point to the file that contains this expression (and assuming that the cm-prefix, cpe-address-prefix, and cpe-pd-prefix templates exist):

```
(list
(create-prefix "cm-prefix" (create-prefix-range 32 0x1))
(create-prefix "cpe-address-prefix" (create-prefix-range 32 0x2))
(create-prefix "cpe-pd-prefix" (create-prefix-range 16 0x1))
)
```

- ***options-expr*** —Expression to define the list of embedded policy options to create with the link.
- ***free-address-config*** —Trap that captures unexpected free address events on this link
- ***type*** —Type of the link (topological, location-independent, universal).
- ***group-name***—Link group to which the link belongs.
- ***embedded-policy***—Policy embedded. When the template is applied, this will replace the entire embedded-policy in the link.

Local Advanced and Regional Advanced Web UI

Step 1 From the **Design** menu, choose **Link Templates** under the DHCPv6 submenu. The List/Add DHCP v6 Link Templates page appears. The page displays the existing templates.

Step 2 Click the **Add Link Templates** icon in the **Link Templates** pane to open the Add Link Template dialog box.

Step 3 Enter a link template name and click **Add Link Template**.

Step 4 Enter an optional description, and optionally choose a preconfigured policy from the drop-down list.

Step 5 Add expressions for the *link-name-expr*, *link-description-expr*, *prefix-expr*, or *options-expr* field attributes (see Using Expressions in Link Templates, on page 8).

Step 6 If the link template is for Prefix Stability, select the link type (type) and specify a link group name (group-name). You can find these attributes in the Prefix Stability block in the Add DHCP v6 Link Template page (see Prefix Stability for details on link types and link groups).

Step 7 Click **Save**.

Step 8 In the regional web UI, you can pull replica link templates, push templates to local clusters, or reclaim the link templates:

- Click **Pull Data** to open the Select Replica Link Template Data to Pull page. Choose a pull mode for the cluster (ensure, replace, or exact), then click **Pull All Link Templates**. On the Report Pull DHCPv6 Link Template page, click **OK**.
- Click **Push** for a specific template (or **Push All**) to open the Push Data to Local Clusters page. Choose a data synchronization mode (ensure, replace, or exact), move the desired cluster or clusters to the Selected table, then click **Push Data to Clusters**.

- Click **Reclaim** to open the Reclaim Link Template page. Move the desired cluster or clusters to the Selected table, then click **Reclaim Data from Clusters**.

CLI Commands

To create the link template, use **link-template** *name* **create** [*attribute*=*value* ...]. For example:

```
nrcmd> link-template example-link-template create [attribute=value]
```

You can set and enable the aforementioned expression setting attributes in the usual way, and you can show and list link templates. For example, to set a prefix expression for the link template, use the following file definition and pointer to the file (and assuming that the cm-prefix, cpe-address-prefix, and cpe-pd-prefix templates exist):

```
> type link-prefix-expr.txt
(list (create-prefix "cm-prefix" (create-prefix-range 32 0x1))
(create-prefix "cpe-address-prefix" (create-prefix-range 32 0x2))
(create-prefix "cpe-pd-prefix" (create-prefix-range 16 0x1)) )

nrcmd> link-template example-link-template set prefix-expr=@link-prefix-expr.txt
```

In addition:

- To clone a link template, use **link-template** *name* **create clone**=*name*.
- To apply a template to one or more links, use **link-template** *name* **apply-to** {**all** | *link* [**,***link***,**...]}. You can create prefixes by using **link-template** *name* **apply-to** *link* [*prefix*], but only with one link specified.
- The link-template includes an embedded-policy object. The link-template-policy CLI command and the Web UI supports the embedded policy on the link-template page.
- When connected to a regional cluster, you can use the following pull, push, and reclaim commands. For push and reclaim, a list of clusters or "all" may be specified.
  - **link-template** < *name* | **all** > **pull** < **ensure** | **replace** | **exact** > *cluster-name* [**-report-only** | **-report**]
  - **link-template** < *name* | **all** > **push** < **ensure** | **replace** | **exact** > *cluster-list* [**-report-only** | **-report**]
  - **link-template** *name* **reclaim** *cluster-list* [**-report-only** | **-report**]

Using Expressions in Link Templates

You can specify expressions in a link template to dynamically create prefix names, IP address ranges, and embedded options when creating a link. Expressions can include context variables and operations.

Note Expressions are not the same as DHCP extensions. Expressions are commonly used to create client identities or look up clients. Extensions (see Using Extension Points) are used to modify request or response packets.

When a template is applied to a link, if the link-template has an embedded policy, it is copied to the link. This embedded policy may or may not have options. As the entire link-template's embedded policy is used (if it

exists), it will wipe out any existing options in the link. If the link-template has no embedded policy, the link's embedded policy is retained. Next the link-template's option expression, if any, is evaluated and the options are added to the embedded policy options in the link (if no embedded policy exists, one is created).

The table below lists the link template predefined variables and Table 2: Link Template Expression Operators lists the link template operators. Note that these variables and operators are not case-sensitive. Table 5: Prefix Template Expression Operators lists the prefix template operators. The link template operators table and prefix template operations table both have same operators, except that only a link template can use **Create Prefix Operator** and prefix template can not use the operator.

Table 1: Link Template Expression Predefined Variables

Predefined Variable	Description
mask-length	Number of prefix mask bits (with a *template-root-prefix* defined).
prefix	Network number and length (with a *template-root-prefix* defined).
prefix-addr	Address portion of the prefix (with a *template-root-prefix* defined).
prefix-length	Number of prefix address bits (with a *template-root-prefix* defined).
template.*attribute*	Attribute of the link template. **Note** The attribute must be explicitly set. Otherwise, the expression will fail to evaluate.
this.*attribute*	Attribute of the link. **Note** The attribute must be explicitly set. Otherwise, the expression will fail to evaluate.
vpn	VPN of the link.

Table 2: Link Template Expression Operators

Expression Operator	Description
Arithmetic Operators (unsigned integer arguments only)	
(+ *arg1 arg2***)**	Adds the two argument values, such as (+ 2 3).
(– *arg1 arg2***)**	Subtracts the second argument value from the first one.
(* *arg1 arg2***)**	Multiplies the values of two arguments.
(/ *arg1 arg2***)**	Divides the value of the first argument by that of the second one (which cannot be zero).
(% *arg1 arg2***)**	Modulo arithmetic operator to determine the remainder of the result of the first argument divided by the second one.
Concatenation Operator	

Expression Operator	Description
(concat *arg1 ... argn***)**	Concatenates the arguments into a string.
List Operator	
(list *oper1 ... opern***)**	Creates an options list or list of prefixes. Required if you need more than one option for a link or prefix, or more than one prefix for a link. All arguments must be **create-v6-option** operation. Nesting is not supported. For example: `(list (create-prefix " cm-prefix" (create-prefix-range 32 0x1)) (create-prefix "cpe-address-prefix" (create-prefix-range 32 0x2)) (create-prefix "cpe-pd-prefix" (create-prefix-range 16 0x1)) )`
Create Prefix Operator	
(create-prefix *template prefix***)**	Creates a prefix based on a predefined prefix template name and the prefix, including the link VPN (assuming that a *template-root-prefix* is defined). The *prefix* argument can be the prefix name, but also the **create-prefix-addr** or **create-prefix-range** operator value. You can use the **list** function to combine multiple operations. For example: `(create-prefix "cm-prefix" (create-prefix-range 32 0x1))`
Create IP Operator	
(create-prefix-addr *prefix interface-id***)**	Creates an IPv6 address string (assuming that a *template-root-prefix* is defined) based on the prefix name and interface ID (an IPv6 address that you can specify as a string), which is the lower 64-bit address in the prefix (which need not be contained in the parent prefix). Used in the *prefix-expr* and *options-expr* attributes.
Create Range Operator	

Expression Operator	Description
(create-prefix-range *size n***)**	Creates an address range (child) for the prefix, used in the *prefix-expr* attribute. The prefix value that the function is based on is either the *template-root-prefix* if applying a link template to a link, or the prefix address, if applying a prefix template to a prefix. Range value—An increase in the prefix length. Size—The number of bits by which you can increase the prefix length. Must be a value from 1 through 32. Must be less than the parent prefix length. n—The nth occurrence of the child prefix. Value can be 0, but is limited to less than two to the power of the size. Must be less than or equal to the size. The size and *n* must be greater than zero. The *n* must be less than or equal to the *size*, and the *size* must by less than the parent prefix length. For example: `(create-prefix-range 32 0x1)`
Create Option Operator	

(create-option *opt val***)**	Creates a DHCPv6 option, used in the *options-expr* attribute. The opt can be the literal string or integer identifying the option. The val is the string representation of the option value, as defined by the option TLV value.
	You can use custom defined and unknown options. For undefined options, the option number must be specified and the data is used as is (as blob data). If the data is a string, the string is used as is and if the data is a number or address, it is used as is.
	For example:
	`(list (create-option "dns-servers" (create-prefix-addr prefix "::2"))`
	`(create-option "domain-list" "sales.example.com,example.com"))`
	Note (create-v6-option opt val) is a synonym for (create-option) and can be used instead; but we recommend that you use (create-option).

Create Vendor Option Operation	

Expression Operator	Description

(create-vendor-option *set-name opt val***)**	Creates a DHCPv6 vendor option, used in the *options-expr* attribute. The *set-name* specifies the option definition set for the vendor option. The opt can be the literal string or integer identifying the vendor option in the set. The val is representation of the option value.
	For example:
	`(list (create-option "dns-servers" (create-prefix-addr prefix "::2"))`
	`(create-vendor-option "dhcp6-cablelabs-config" 17`
	`"(enterprise-id 4491((tftp-servers 32 3800:0:0:180::6)`
	`(config-file-name 33 modem_ipv6.bin)(syslog-servers 34 3800:0:0:180::8)`
	`(rfc868-servers 37 3800:0:0:180::6)(time-offset 38 -5h)`
	`(cablelabs-client-configuration 2170 (primary-dhcp-server 1 10.38.1.5)`
	`(secondary-dhcp-server 2 10.38.1.6))))"))`
	Note (create-v6-vendor-option opt val) is a synonym for (create-vendor-option) and can be used instead; but we recommend that you use (create-vendor-option).

Using Expressions in Scope Templates

You can specify expressions in a scope template to dynamically create scope names, IP address ranges, and embedded options when creating a scope. Expressions can include context variables and operations.

Note Expressions are not the same as DHCP extensions. Expressions are commonly used to create client identities or look up clients. Extensions (see Using Extension Points) are used to modify request or response packets. If you apply the template to a scope that already has ranges defined, the address range expression of the scope template is not evaluated for that scope.

The following table lists the scope expression functions. Note that these functions are not case-sensitive.

Table 3: Scope Template Expression Functions

Expression Function	Description
Context Variables	
bcast-addr	Derived from the broadcast address in the subnet, such as 192.168.50.255. Use in any expression field.
first-addr	Derived from the first address in the subnet, such as the first address in 192.168.50.64/26 is 192.168.50.65. Use in any expression field.
last-addr	Derived from the last address in the subnet, such as the last address in 192.168.50.64/26 is 192.168.50.127. Use in any expression field.
mask-addr	Derived from the network mask address in the subnet, such as 255.255.255.0. Use in any expression field.
mask-count	Derived from the number of bits in the network address of the subnet, such as 24. Use in the Scope Name Expression or Embedded Policy Option Expression field.
naddrs	Derived from the number of IP addresses in the subnet, such as 255. Use in the Scope Name Expression field.
nhosts	Derived number of usable hosts in the subnet, such as 254. Use in any expression field.
subnet	Derived from the IP address and mask of the subnet, such as 192.168.50.0/24. Use in the Scope Name Expression or Embedded Policy Option Expression field.
subnet-addr	Derived from the subnet address, such as 192.168.50.0. Use in any expression field.
template.*attribute*	Attribute of the scope template, such as template.ping-timeout. Use in the Embedded Policy Option Expression field. **Note** The attribute must be explicitly set. Otherwise, the expression will fail to evaluate.
this.*attribute*	Attribute of the scope. **Note** The attribute must be explicitly set. Otherwise, the expression will fail to evaluate.
Arithmetic Operations (unsigned integer arguments only)	

Expression Function	Description
(+ *arg1 arg2*)	Adds the two argument values, such as (+ 2 3).
(– *arg1 arg2*)	Subtracts the second argument value from the first one, such as with *ping-timeout* defined as 100, (– template.ping-timeout 10) yields 90.
(* *arg1 arg2*)	Multiplies the values of two arguments.
(/ *arg1 arg2*)	Divides the value of the first argument by that of the second one (which cannot be zero).
Concatenation Operation	
(concat *arg1 ... argn*)	Concatenates the arguments into a string, to be used in the Scope Name Expression field. Examples: With subnet=192.168.50.0/24 and template.ping-timeout=100: **(concat "ISP-" subnet)** --> ISP-192.168.50.0/24 **(concat subnet "-" (+ template.ping-timeout 10))** --> 192.168.50.0/24-110 (concat "ISP-" subnet "-" (+ template.ping-timeout 10)) --> ISP-192.168.50.0/24-110 See also Scope Name Expression Example, on page 17.
Create Option Operation	

Expression Function	Description

(create-option *opt val***)**	Use create-option in the Embedded Policy Option Expression field to create new DHCP options for the scope. The first argument can be an integer or string to represent the option number or name. The second argument can be a string or blob to give the option a value.
	
	You can also specify custom defined and unknown options. For undefined options, the option number must be specified and the data is used as is (as blob data). If the data is a string, the string is used as is and if the data is a number or address, it is used as is.
	
	Examples:
	
	`(list (create-option "domain-name" "example.com")`
	`(create-option 3 "10.10.10.1"))`
	`(create-option "routers" "10.10.10.1,10.10.10.2,10.10.10.3")`
	`(create-option "routers" (create-ipaddr subnet 10))`
	
	See also Embedded Policy Option Expression Example, on page 18.

Create Vendor Option Operation	

(create-vendor-option *set-name opt val***)**	Use the create-vendor-option in the Embedded Policy Option Expression field to creates a DHCP vendor option. The set-name specifies the option definition set for the vendor option. The opt can be the literal string or integer identifying the vendor option in the set. The val is representation of the option value.
	
	For example:
	
	`(list (create-option "routers" (create-ipaddr subnet 1))`
	
	`(create-vendor-option "dhcp-cablelabs-config" 125`
	
	`(concat "(tftp-servers 2 " (create-ipaddr subnet 2)`
	`")")))`

Create Range Operation	

Expression Function	Description

(create-range *start end***)**	Use this operation in the Range Expression field. It creates an IP address range for the scope. The first argument is the start of the address range and can be an integer or IP address string. The second argument is the end of the range and can be an integer or IP address string. Do not include the local host or broadcast address determined by the mask (such as 0 and 255 for /24 subnets) in the range. Validation ensures that the range must be in the subnet defined by the template and that the first argument value must be lower than the second. An integer value determines the position of the address in the given subnet. Examples (with subnet=192.168.50.0/26):
	`(create-range "192.168.50.65" "192.168.50.74")`
	`--> 192.168.50.65 - 192.168.50.74`
	`(create-range 1 10)`
	`--> 192.168.50.65 - 192.168.50.74`
	See also Range Expression Example, on page 17.

Create IP Operation	
(create-ipaddr *net host***)**	Use this operation in the Embedded Policy Option Expression or Range Expression fields. It creates an IP address string. The net argument is a string or variable. The host argument is an integer. Example: `(create-ipaddr subnet 4)`
List Operation	

(list *oper1 ... opern***)**	Arguments must all be create-option or create-range operations. Nesting is not supported. Examples:
	`(list (create-option "routers" "10.10.10.1")`
	`(create-option "domain-name" "example.com"))`
	`(list (create-range 1 5) (create-range 10 20))`

Local Advanced and Regional Web UI

There are three fields on the Add DHCP Scope Template page for which you must specify an expression:

- **Scope Name Expression**—Must return a string
- **Range Expression**—Must return IP addresses
- **Embedded Policy Option Expression**—No requirements

CLI Commands

Use the following **scope-template** command attributes:

- **scope-name**
- **ranges-exp**
- **options-exp**

Scope Name Expression Example

You might want to set an expression so that the template constructs scope names starting with "ISP–" and followed by the subnet of the scope and a derivative of its ping timeout value. You would use the following expression in the Scope Name Expression field:

```
(concat "ISP-" subnet "-" (+ template.ping-timeout 10))
```

The elements of the example expression are:

- **(concat** ...**)**—Concatenation operation, which concatenates all the following values into one value.
- **"ISP–"**—String with which to start the scope name.
- **subnet**—Keyword variable that indicates to use the existing subnet defined for the scope.
- **"–"**—Indicates to include this hyphen to construct the value.
- **(+ template.ping-timeout 10)**—Indicates to add the *ping-timeout* property value for the scope to the number 10.

If the scope subnet happens to be 192.168.50.0/24 and its *ping-timeout* value 100, the resulting constructed scope name would be:

```
ISP-192.168.50.0/24-110
```

Range Expression Example

You might want to set an expression so that the template constructs only certain address ranges for scopes. You can either be explicit about the actual starting and ending addresses, or you can make them relative to the subnet. Here are two ways of requesting relative ranges in the Range Expression field:

```
(create-range first-addr last-addr)
(create-range 1 10)
```

The first **create-range** operation creates the address range based on the first through last usable address in the subnet. For the 192.168.50.0/24 subnet, for example, the address range would be 192.168.50.1 through 192.168.50.254. Because the second operation specifies integers instead of full IP addresses, it makes the range relative to the subnet based on its mask. If the template discovers the subnet to be 192.168.50.0/26, it takes the first through tenth address in this subnet, which would be 192.168.50.65 through 192.168.50.74.

To set the range expressions in the CLI, you should place the expression into a file and use a command such as:

```
nrcmd> scope-template example-template set ranges-expr=@ file
```

where *file* is the name of the file that you created with the expressions.

Embedded Policy Option Expression Example

An embedded policy is important because the DHCP server looks at it before it looks at the assigned, named policy of the scope. This is usually where you would set the DHCP options on a scope. You might want to set an expression so that the template constructs DHCP options for the scope embedded policy. Here are some examples:

```
(create-option "domain-name" "example.com")
(create-option 3 "10.10.10.1")
(create-option "routers" (create-ipaddr subnet 10))
```

The first **create-option** operation associates the value example.com with the *domain-name* option for the scope. The second operation associates the address 10.10.10.1 with the *routers* option (number 3). The third operation creates an IP address for the *routers* option based on the tenth address in a subnet.

To set the policy options expressions in the CLI, you should place the expression into a file and use a command such as:

```
nrcmd> scope-template example-template set options-expr=@ file
```

where *file* is the name of the file that you created with the expressions.

Note Trying to specify the expression directly on the CLI command line will likely fail because of embedded spaces and special characters such as the quotes. Use the ***@file*** syntax as it avoids any potential issues with the CLI command parser. But the WebUI does not support the @file syntax. You can enter complex expressions directly in the Web UI.

Using Expressions in Prefix Templates

You can specify expressions in a prefix template to dynamically create prefix names, IP address ranges, and embedded options when creating a prefix. Expressions can include context variables and operations.

Note Expressions are not the same as DHCP extensions. Expressions are commonly used to create client identities or look up clients. Extensions (see Using Extension Points) are used to modify request or response packets.

When a template is applied to a prefix, if the prefix-template has an embedded policy, it is copied to the prefix. This embedded policy may or may not have options. As the entire prefix-template's embedded policy is used (if it exists), it will wipe out any existing options in the prefix. If the prefix-template has no embedded policy, the prefix's embedded policy is retained. Next the prefix-template's option expression, if any, is evaluated and the options are added to the embedded policy options in the prefix (if no embedded policy exists, one is created).

The tables below lists the prefix template predefined variables and lists the operator. Note that these variables and operators are not case-sensitive.

Table 4: Prefix Template Expression Predefined Variables

Predefined Variable	Description
prefix	Network number and length, based on the template root prefix if applying a link template to a link, or the prefix address if applying a prefix template to a prefix.
vpn	VPN of the prefix.
prefix-addr	Address portion of the prefix.
prefix-length	Number of prefix address bits.
mask-length	Number of prefix mask bits.
template.*attribute*	Attribute of the prefix template. **Note** The attribute must be explicitly set. Otherwise, the expression will fail to evaluate.
this.*attribute*	Attribute of the prefix, such as this.link for the prefix's link name. **Note** The attribute must be explicitly set. Otherwise, the expression will fail to evaluate.

Table 5: Prefix Template Expression Operators

Expression Operator	Description
Arithmetic Operators (unsigned integer arguments only)	
(+ *arg1 arg2***)**	Adds the two argument values, such as (+ 2 3).
(– *arg1 arg2***)**	Subtracts the second argument value from the first one, such as with *ping-timeout* defined as 100, (– template.ping-timeout 10) yields 90.
(* *arg1 arg2***)**	Multiplies the values of two arguments.
(/ *arg1 arg2***)**	Divides the value of the first argument by that of the second one (which cannot be zero).
(% *arg1 arg2***)**	Modulo arithmetic operator to determine the remainder of the result of the first argument divided by the second one.
Concatenation Operator	
(concat *arg1 ... argn***)**	Concatenates the arguments into a string.
List Operator	

Expression Operator	Description
(list *oper1 ... opern***)**	Creates an options list or list of prefixes. Required if needing more than one option for a prefix. All arguments must be **create-v6-option** or **create-prefix-range** operations. Nesting is not supported.
Create IP Operator	
(create-prefix-addr *prefix-name interface-id***)**	Creates an IPv6 address string based on the prefix name and interface ID (an IPv6 address that you can specify as a string), which is the lower 64-bit address in the prefix (which need not be contained in the parent prefix). Used in the *range-expr* and *options-expr* attributes.
Create Range Operator	
(create-prefix-range *size n***)**	Creates an address range (child) for the prefix, used in the *range-expr* attribute. The prefix value that the function is based on is either the *template-root-prefix* if applying a link template to a link, or the prefix address if applying a prefix template to a prefix. Range value—An increase in the prefix length. Size—The number of bits by which you can increase the prefix length. Must be a value from 1 through 32. Must be less than the parent prefix length. n—The nth occurrence of the child prefix. Value can be 0, but is limited to less than two to the power of the size. Must be less than or equal to the size. The size and n must be greater than zero. The *n* must be less than or equal to the *size*, and the *size* must by less than the parent prefix length. For example: `(create-prefix-range 32 0x1)`
Create Option Operation	

Expression Operator	Description

(create-option *opt val***)**	Creates a DHCPv6 option, used in the *options-expr* attribute. The opt can be the literal string or integer identifying the option. The val is the string representation of the option value, as defined by the option TLV value.
	
	You can use custom defined and unknown options. For undefined options, the option number must be specified and the data is used as is (as blob data). If the data is a string, the string is used as is and if the data is a number or address, it is used as is.
	
	For example:
	
	`(list (create-option "dns-servers" (create-prefix-addr prefix "::2"))`
	
	`(create-option "domain-list" "sales.example.com,example.com"))`
	
	Note (create-v6-option opt val) is a synonym for (create-option) and can be used instead.

Create Vendor Option Operator	

Expression Operator	Description

(create-vendor-option *set-name opt val***)**	Creates a DHCPv6 vendor option, used in the *options-expr* attribute. The set-name specifies the option definition set for the vendor option. The opt can be the literal string or integer identifying the vendor option in the set. The val is representation of the option value.
	For example:
	`(list (create-option "dns-servers" (create-prefix-addr prefix "::2"))`
	`set-name`
	`opt val`
	`(create-vendor-option "dhcp6-cablelabs-config" 17`
	`"(enterprise-id 4491((tftp-servers 32 3800:0:0:180::6)`
	`(config-file-name 33 modem_ipv6.bin)`
	`(syslog-servers 34 3800:0:0:180::8)`
	`(rfc868-servers 37 3800:0:0:180::6)(time-offset 38 -5h)`
	`(cablelabs-client-configuration 2170 (primary-dhcp-server 1 10.38.1.5)`
	`(secondary-dhcp-server 2 10.38.1.6))))"))`
	Note (create-v6-vendor-option opt val) is a synonym for (create-vendor-option) and can be used instead.

Note We recommend that you use create-option and create-vendor-option for v4 and v6.

Using Expressions in Link Templates

You can specify expressions in a link template to dynamically create prefix names, IP address ranges, and embedded options when creating a link. Expressions can include context variables and operations.

Note Expressions are not the same as DHCP extensions. Expressions are commonly used to create client identities or look up clients. Extensions (see Using Extension Points) are used to modify request or response packets.

When a template is applied to a link, if the link-template has an embedded policy, it is copied to the link. This embedded policy may or may not have options. As the entire link-template's embedded policy is used (if it

exists), it will wipe out any existing options in the link. If the link-template has no embedded policy, the link's embedded policy is retained. Next the link-template's option expression, if any, is evaluated and the options are added to the embedded policy options in the link (if no embedded policy exists, one is created).

The table below lists the link template predefined variables and Table 7: Link Template Expression Operators lists the link template operators. Note that these variables and operators are not case-sensitive. Table 5: Prefix Template Expression Operators lists the prefix template operators. The link template operators table and prefix template operations table both have same operators, except that only a link template can use **Create Prefix Operator** and prefix template can not use the operator.

Table 6: Link Template Expression Predefined Variables

Predefined Variable	Description
mask-length	Number of prefix mask bits (with a *template-root-prefix* defined).
prefix	Network number and length (with a *template-root-prefix* defined).
prefix-addr	Address portion of the prefix (with a *template-root-prefix* defined).
prefix-length	Number of prefix address bits (with a *template-root-prefix* defined).
template.*attribute*	Attribute of the link template. **Note** The attribute must be explicitly set. Otherwise, the expression will fail to evaluate.
this.*attribute*	Attribute of the link. **Note** The attribute must be explicitly set. Otherwise, the expression will fail to evaluate.
vpn	VPN of the link.

Table 7: Link Template Expression Operators

Expression Operator	Description
Arithmetic Operators (unsigned integer arguments only)	
(+ *arg1 arg2***)**	Adds the two argument values, such as (+ 2 3).
(– *arg1 arg2***)**	Subtracts the second argument value from the first one.
(* *arg1 arg2***)**	Multiplies the values of two arguments.
(/ *arg1 arg2***)**	Divides the value of the first argument by that of the second one (which cannot be zero).
(% *arg1 arg2***)**	Modulo arithmetic operator to determine the remainder of the result of the first argument divided by the second one.
Concatenation Operator	

Expression Operator	Description
(concat *arg1 ... argn***)**	Concatenates the arguments into a string.
List Operator	
(list *oper1 ... opern***)**	Creates an options list or list of prefixes. Required if you need more than one option for a link or prefix, or more than one prefix for a link. All arguments must be **create-v6-option** operation. Nesting is not supported. For example: `(list (create-prefix " cm-prefix" (create-prefix-range 32 0x1)) (create-prefix "cpe-address-prefix" (create-prefix-range 32 0x2)) (create-prefix "cpe-pd-prefix" (create-prefix-range 16 0x1)) )`
Create Prefix Operator	
(create-prefix *template prefix***)**	Creates a prefix based on a predefined prefix template name and the prefix, including the link VPN (assuming that a *template-root-prefix* is defined). The *prefix* argument can be the prefix name, but also the **create-prefix-addr** or **create-prefix-range** operator value. You can use the **list** function to combine multiple operations. For example: `(create-prefix "cm-prefix" (create-prefix-range 32 0x1))`
Create IP Operator	
(create-prefix-addr *prefix interface-id***)**	Creates an IPv6 address string (assuming that a *template-root-prefix* is defined) based on the prefix name and interface ID (an IPv6 address that you can specify as a string), which is the lower 64-bit address in the prefix (which need not be contained in the parent prefix). Used in the *prefix-expr* and *options-expr* attributes.
Create Range Operator	

Expression Operator	Description
(create-prefix-range *size n***)**	Creates an address range (child) for the prefix, used in the *prefix-expr* attribute. The prefix value that the function is based on is either the *template-root-prefix* if applying a link template to a link, or the prefix address, if applying a prefix template to a prefix. Range value—An increase in the prefix length. Size—The number of bits by which you can increase the prefix length. Must be a value from 1 through 32. Must be less than the parent prefix length. n—The nth occurrence of the child prefix. Value can be 0, but is limited to less than two to the power of the size. Must be less than or equal to the size. The size and *n* must be greater than zero. The *n* must be less than or equal to the *size*, and the *size* must by less than the parent prefix length. For example: `(create-prefix-range 32 0x1)`
Create Option Operator	

(create-option *opt val***)**	Creates a DHCPv6 option, used in the *options-expr* attribute. The opt can be the literal string or integer identifying the option. The val is the string representation of the option value, as defined by the option TLV value.
	You can use custom defined and unknown options. For undefined options, the option number must be specified and the data is used as is (as blob data). If the data is a string, the string is used as is and if the data is a number or address, it is used as is.
	For example:
	`(list (create-option "dns-servers" (create-prefix-addr prefix "::2"))`
	`(create-option "domain-list" "sales.example.com,example.com"))`
	Note (create-v6-option opt val) is a synonym for (create-option) and can be used instead; but we recommend that you use (create-option).

Create Vendor Option Operation	

Expression Operator	Description

(create-vendor-option *set-name opt val***)**	Creates a DHCPv6 vendor option, used in the *options-expr* attribute. The *set-name* specifies the option definition set for the vendor option. The opt can be the literal string or integer identifying the vendor option in the set. The val is representation of the option value.
	For example:
	`(list (create-option "dns-servers" (create-prefix-addr prefix "::2"))`
	`(create-vendor-option "dhcp6-cablelabs-config" 17`
	`"(enterprise-id 4491((tftp-servers 32 3800:0:0:180::6)`
	`(config-file-name 33 modem_ipv6.bin)(syslog-servers 34 3800:0:0:180::8)`
	`(rfc868-servers 37 3800:0:0:180::6)(time-offset 38 -5h)`
	`(cablelabs-client-configuration 2170 (primary-dhcp-server 1 10.38.1.5)`
	`(secondary-dhcp-server 2 10.38.1.6))))"))`
	Note (create-v6-vendor-option opt val) is a synonym for (create-vendor-option) and can be used instead; but we recommend that you use (create-vendor-option).