Managing Tunnel Provisioning

This section describes how to configure IoT FND for tunnel provisioning, and how to manage and monitor tunnels connecting FARs (CGRs and C800s) and HERs, and includes the following topics.

- Overview
- Configuring Tunnel Provisioning
- Monitoring Tunnel Status
- Reprovisioning CGRs

Overview

IoT FND sends the commands generated from processing the tunnel provisioning templates to FARs and HERs to provision secure tunnels between them. The default IoT FND templates contain CLI commands to set up and configure GRE and IPsec tunnels.

Figure 1 Tunnels Connect FARs and their Corresponding HERs

To provision tunnels between HERs and FARs, IoT FND executes CLI tunnel configuration commands on these devices. By default, IoT FND provides basic tunnel configuration templates containing the CLI tunnel configuration commands. You can also use your own templates. Although the tunnel provisioning process is automatic, you must first complete the configuration steps outlined in Tunnel Provisioning Configuration Process. After that, whenever a FAR comes online, IoT FND automatically provisions it with a tunnel. Before you configure IoT FND for tunnel provisioning, ensure that the IoT FND TPS Proxy is installed and running.

Tunnel Provisioning Configuration Process

You must generate the keystore files on the IoT FND and TPS Proxy before configuring tunnel provisioning. Then, you configure IoT FND and the TPS Proxy to talk to one another (Setting Up the TPS Proxy and Configuring IoT FND to Use the TPS Proxy).
To configure IoT FND for tunnel provisioning:

1. (CG-OS CGRs) Configure the DHCP servers. Configure DHCP servers to provide unique IP addresses to IoT FND. The default IoT FND tunnel provisioning templates configure a loopback interface and the IP addresses required to create the tunnels. Cisco IOS CGRs use FlexVPN. Ensure that the template only contains addresses for the loopback interface.

2. Configure the tunnel settings. Configure the NMS URL and the DHCP proxy client settings on the Provisioning Settings page in IoT FND (Admin > System Management > Provisioning Settings).

3. (CG-OS CGRs) Configure IoT FND to accept FAR registration requests on first contact (call home) to request tunnel provisioning. Cisco IOS CGRs use the CGNA service.

4. Configure HER management. Configure HERs to allow management by IoT FND using NETCONF over SSH.

5. Add HERs to IoT FND.

6. Review the IoT FND tunnel provisioning templates to ensure that they create the correct type of tunnel.

7. (Optional) If you plan to use your own templates for tunnel provisioning, create one or more tunnel provisioning groups and modify the default tunnel provisioning templates.

8. (CG-OS CGRs) Configure FARs to call home. Configure FARs to contact IoT FND over HTTPS through the IoT FND TPS proxy.

9. Add FARs to IoT FND. Import the FARs in to IoT FND using the Notice-of-Shipment XML file.

10. Map FARs to their corresponding HER.

After completing the previous steps, deploy the FARs and power them on. Tunnel provisioning happens automatically.

This is the sequence of events after a FAR is turned on:

1. Upon joining the uplink network after being turned on, the FAR sends a request for certificate enrollment.

2. The FAR then requests tunnel provisioning to IoT FND through the IoT FND TPS Proxy.

3. IoT FND looks up the FAR record in the IoT FND database and determines which tunnel provisioning templates to use. IoT FND also looks up which HERs to which to establish a tunnel.

4. For Cisco IOS CGRs, the default templates configure the CGR to use FlexVPN. The FlexVPN client is configured on the CGR that will contact the HER and ask for a FlexVPN tunnel to be dynamically constructed. This is how the HER dynamically adds a new tunnel endpoint interface for the CGR.

5. Before processing FAR templates, IoT FND processes the HER Tunnel Deletion template and sends the resulting commands to the HERs. This is done for each HER to remove existing tunnel configuration that may be associated with the FAR.
6. IoT FND uses the FreeMarker template engine to process the FAR Tunnel Addition template. The engine converts the templates to text, which IoT FND assumes to be CLI configuration commands (CG-OS or Cisco IOS, per the CGR). IoT FND uses these commands to configure and bring up one end of the tunnel on the FAR.

7. IoT FND uses the FreeMarker template engine to process the HER Tunnel Addition template. The engine converts the templates to text, which IoT FND assumes to be commands for configuring the tunnel on the HERs.

8. This step is OS-specific:

- For Cisco IOS CGRs, if no errors occurred applying the commands generated by the templates to the FAR and HERs, IoT FND configures a new active CGNA profile “cg-nms-register,” and deactivates the cg-nms-tunnel profile. That cg-nms-register profile uses the IoT FND URL.
- For CG-OS CGRs, IoT FND re-configures the call home URL to the IoT FND URL specified in the Provisioning Settings page (Admin > System Management > Provisioning Settings).

The specified URL uses the IoT FND registration port (default 9121) instead of the tunnel provisioning port. The Fully Qualified Domain Name (FQDN) in that URL is different and resolves to an IP address that is only reachable through the tunnels.
Configuring Tunnel Provisioning

This section describes how to configure IoT FND for tunnel provisioning.

- Configuring the DHCP Server for Tunnel Provisioning
- Configuring DHCP for Tunnel Provisioning Using CNR

Configuring the DHCP Server for Tunnel Provisioning

For tunnel provisioning to succeed, configure the DHCP server used by IoT FND to supply addresses to create tunnels between the FARs and HERs. For example, configure the DHCP server to provide IP addresses for tunnel provisioning on a permanent-lease basis.

IoT FND makes the DHCP requests based on the settings defined in the tunnel provisioning templates. During tunnel provisioning, the IoT FND templates can make two kinds of DHCP requests:

- Request an IP address, and then make it available to the template.
- Request a subnet with two IP addresses, and then make both addresses available to the template.

IoT FND can make these requests for IPv4 addresses and IPv6 addresses.

The ability to request DHCP addresses from the template gives you maximum flexibility when defining tunnel configurations because you allocate the exact address needed for each FAR and corresponding interface on the HER. The default tunnel provisioning templates provided address the most common use case: one IPsec tunnel between the FAR and its corresponding HER. Each end of this IPsec tunnel gets a dynamically allocated IPv4 address:

- If your DHCP server supports subnet allocation, use it to obtain two addresses that belong to the same subnet.
- If your DHCP server only supports address allocation, configure it so that the two DHCP address requests return addresses that can be used as ends of an IPsec tunnel.
- If your routing plan calls for allocating unique IPv4 addresses for each FAR and assigning it to a loopback interface above the IPsec tunnel, allocate this address using the IoT FND template.

If you choose to build IPv6 GRE tunnels, allocate the IPv6 addresses for each end of the tunnel using DHCP prefix delegation or individual address requests.

This section describes example DHCP settings for tunnel provisioning. How you configure these settings depends on your installation. This section provides general guidelines for configuring the DHCP server for tunnel provisioning using the Cisco Network Registrar (CNR).

Configuring the DHCP Client Listen Address for Red Hat Linux 7.x Installations

In Red Hat Linux 7.x installations, you must configure specific IPv4 and IPv6 addresses on the IoT FND Linux host server for binding DHCP IPv4 and IPv6 clients by setting the following fields on the Admin > Provisioning Settings page (Figure 2):

- **DHCPv6 Proxy Client > Client Listen Address**: Set the value to the IPv6 address of the interface to use to obtain IPv6 DHCP leases from the DHCP server. The default value is “::”. Change the default setting to an actual IPv6 address on the Linux host machine.

- **DHCPv4 Proxy Client > Client Listen Address**: Set the value to the IPv4 address of the interface to use to obtain IPv4 DHCP leases from the DHCP server. The default value is “0.0.0.0”. Change the default setting to an actual IPv4 address on the Linux host machine.
Managing Tunnel Provisioning

Figure 2  Provisioning Settings Page Client Listening Fields

Configuring DHCP for Tunnel Provisioning Using CNR

The CNR CLI script in the following example configures the CNR DHCP server to service requests made by the default tunnel provisioning templates in IoT FND. When using this script, ensure that the subnets are appropriate for your DHCP server environment.

Example CNR DHCP Server Tunnel Provisioning Script

```bash
# These commented out commands support re-applying the configuration by first
# removing any previously applied configuration, in reverse order. This should
# not be done in a production environment, but may be useful when initially
# developing and testing a configuration.

# scope v4address-perm delete
# dhcp-address-block v4subnet-perm delete
# prefix v6subnet-perm delete
# prefix v6address-perm delete
# policy permanent delete

# Configure the server to automatically map any IPv4 or IPv6 user class
# option values to selection tags. By default CG-NMS includes a value of
# "CG-NMS" for the user class in its requests. The tag is used to insure
# prefixes and scopes configured to satisfy requests from CG-NMS are only
# used for that purpose.

dhcp set map-user-class-id=append-to-tags

# Since CG-NMS uses the leased addresses and subnets in router
# configuration the addresses and subnets must be permanently allocated
# for that purpose. Create a policy that instructs the DHCP server to
# offer a permanent lease.

policy permanent create
```
policy permanent set permanent-leases=enabled

# Configure DHCPv6.
# The default CG-NMS tunnel template will request IPv6 addresses for
# use with CGR loopback interfaces.
prefix v6address-perm create 2001:DB8:0:0:1::/80 dhcp-type=dhcp
prefix v6address-perm set description="Pool for leasing addresses for loopback interfaces."
prefix v6address-perm set policy=permanent
prefix v6address-perm set selection-tags=CG-NMS

# The default CG-NMS tunnel template will request IPv6 prefixes for
# use with GRE tunnels. Force use of a /127 prefix.
prefix v6subnet-perm create 2001:DB8:0:0:2::/80 dhcp-type=prefix-delegation
prefix v6subnet-perm set description="Pool for leasing prefixes for GRE tunnels."
prefix v6subnet-perm set policy=permanent
prefix v6subnet-perm set selection-tags=CG-NMS
prefix-policy v6subnet-perm set default-prefix-length=127
prefix-policy v6subnet-perm set shortest-prefix-length=127

# Configure DHCPv4.
# The default CG-NMS tunnel template will request IPv4 subnets for
# use with IPsec tunnels. Note that currently address pools for
# IPv4 subnet allocation can only be configured using the CLI as the
# CNR Web UI does not currently support them.
# If CNR allowed you to set a description on DHCP address blocks it would be:
# "Pool for leasing subnets for IPsec tunnels."
dhcp-address-block v4subnet-perm create 192.0.2.0/24
dhcp-address-block v4subnet-perm set default-subnet-size=31
dhcp-address-block v4subnet-perm set policy=permanent
dhcp-address-block v4subnet-perm set selection-tags=CG-NMS

# The default CG-NMS tunnel template will request IPv4 addresses for
# use with loopback interfaces.
scope v4address-perm create 198.51.100.0 255.255.255.0
scope v4address-perm set description="Pool for leasing addresses for loopback interfaces."
scope v4address-perm set policy=permanent
scope v4address-perm addRange 198.51.100.2 198.51.100.254
scope v4address-perm set selection-tag-list=CG-NMS

# Configure detailed logging of incoming and outgoing packets. This is useful when
# debugging issues involving DHCP, however this level of logging will lower the
# performance of the DHCP server. If this is a production server under heavy load
# it may be necessary to forgo detailed packet logging.
dhcp set

# Save the changes and reload the server to have them take effect.
save
dhcp reload

# List the current configuration.
policy list
prefix list
dhcp-address-block list
scope list
Managing Tunnel Provisioning

Configuring Tunnel Group Settings

You use groups in IoT FND to bulk configure tunnel provisioning for FARs. By default, all FARs added to IoT FND (see Adding Devices in Bulk) the appropriate default group: default-cgr1000 or default-c800. Default groups contain the three templates IoT FND uses for tunnel provisioning.

Topics in this section include the following:

- Creating Tunnel Groups
- Deleting Tunnel Groups
- Viewing Tunnel Groups
- Moving FARs to Another Group
- Renaming a Tunnel Group

Creating Tunnel Groups

If you plan to use one set of templates for all FARs, whether using the default templates, modified default templates or custom templates, do not create additional groups. To define multiple sets of templates, create groups and customize the templates for these groups.

Note: CGRs and C800s can be in the same tunnel provisioning group if your custom templates are applicable to both router types.

To create a tunnel group:

1. Choose Config > Tunnel Provisioning.
2. Click Add Group (➕).
3. Enter a name of the new group, and then click OK.

The group appears in the TUNNEL GROUPS pane.

After creating a tunnel group, the next step is to move FARs from other groups to it, as described in Moving FARs to Another Group.

Deleting Tunnel Groups

Only empty groups can be deleted. Before you can delete a tunnel group, you must move the devices it contains to another group.

To delete an empty tunnel group:

1. Choose Config > Tunnel Provisioning.
2. In the TUNNEL GROUPS pane, select the tunnel group to delete.
3. Click Delete Group (➖).
4. Click Yes.
Viewing Tunnel Groups

The Tunnel Provisioning page lists information about existing tunnel groups. Follow these steps to view the tunnel groups defined in IoT FND:

1. Choose Config > Tunnel Provisioning.
2. Click Group Members.
3. In the TUNNEL GROUPS pane, select a group.

IoT FND displays a list of all FARs in the group. Use the list navigation buttons to scroll through the list. Table 1 describes the list fields.

Table 1  Tunnel Group Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>FAR EID (device identifier).</td>
</tr>
<tr>
<td>Status</td>
<td>Status of the FAR:</td>
</tr>
<tr>
<td></td>
<td>■ Unheard—The FAR has not contacted IoT FND yet.</td>
</tr>
<tr>
<td></td>
<td>■ Unsupported—The FAR is not supported by IoT FND.</td>
</tr>
<tr>
<td></td>
<td>■ Up—The FAR is in operation.</td>
</tr>
<tr>
<td></td>
<td>■ Down—The FAR is turned off.</td>
</tr>
<tr>
<td>Last Heard</td>
<td>Last time the router contacted or sent metrics to IoT FND. If the router never contacted IoT FND, never appears in this field. Otherwise, IoT FND displays the date and time of the last contact, for example, 4/10 19:06.</td>
</tr>
<tr>
<td>Tunnel Source Interface 1</td>
<td>FAR interface used by the tunnel.</td>
</tr>
<tr>
<td>Tunnel Source Interface 2</td>
<td></td>
</tr>
<tr>
<td>OSPF Area 1</td>
<td>Open shortest path first (OSPF) areas 1 and 2.</td>
</tr>
<tr>
<td>OSPF Area 2</td>
<td></td>
</tr>
<tr>
<td>OSPFv3 Area 1</td>
<td>OSPFv3 area 1.</td>
</tr>
</tbody>
</table>
Managing Tunnel Provisioning

Renaming a Tunnel Group

You can rename a tunnel group at any time. Cisco recommends using short, meaningful names. Names cannot be more than 250 characters long.

To rename a tunnel group:

1. Choose Config > Tunnel Provisioning.
2. In the TUNNEL GROUPS pane, roll over the tunnel group to rename and click the Edit pencil icon ( ).
3. Enter the new name, and then click OK.

Note: As shown above, when you enter an invalid character entry (such as, @, #, !, or +) within a field, highlights the field in red, and disables the OK button.

Moving FARs to Another Group

You can move FARs to another group in two ways:

- Moving FARs to Another Group Manually
- Moving FARs to Another Group in Bulk
Moving FARs to Another Group Manually

To move FARs to another group manually:

1. Choose **Config > Tunnel Provisioning**.
2. Click the **Group Members** tab.
3. In the TUNNEL GROUPS pane, select the tunnel group with the routers to move.
4. Choose the device type from the **Select a device type** drop-down menu.
5. Check the check boxes of the FARs to move.
   
   To select all FARs in a group, click the check box at the top of the column. When you select devices, a yellow bar displays that maintains a count of selected devices and has the Clear Selection and Select All commands. The maximum number of devices you can select is 1000.
6. Click the **Change Tunnel Group** button.
7. From the drop-down menu, choose the tunnel group to which you want to move the FARs.
8. Click **Change Tunnel Group**.
9. Click **OK** to close the dialog box.

Moving FARs to Another Group in Bulk

You can move FARs in bulk to another group by importing a CSV or XML file containing the names of the FARs to move. Ensure that the file contains entries in the format shown the following example:

```
eid
CGR1210/k9+JSM1
CGR1210/k9+JSM2
CGR1210/k9+JSM3
```
The first line is the header, which tells IoT FND to expect FAR EIDs in the remaining lines (one FAR EID per line).

To move FARs to another group in bulk:

1. Create a CSV or XML file with the EIDs of the devices to move to a different group.
2. Choose **Config > Tunnel Provisioning**.
3. Click **Assign Devices to Group**.
4. Click **Browse** and locate the file that contains the FARs that you want to move.
5. From the **Group** drop-down menu, choose the destination tunnel group.
6. Click **Change Group**.
7. Click **Close**.

**Configuring Tunnel Provisioning Templates**

IoT FND has three default tunnel provisioning templates:

- Field Area Router Tunnel Addition—IoT FND uses this template to generate the CLI configuration commands for creating one end of an IPsec tunnel on the FAR.
- Head-End Router Tunnel Addition—IoT FND uses this template to generate the CLI configuration commands for creating the other end of the IPsec tunnel on the HER.
- Head-End Router Tunnel Deletion—IoT FND uses this template to generate the CLI configuration commands for deleting any existing tunnel to the FAR at the other end of the tunnel.
Managing Tunnel Provisioning

Tunnel Provisioning Template Syntax

The IoT FND tunnel provisioning templates are expressed with the FreeMarker syntax. FreeMarker is an open-source Java-based engine for processing templates and is built into IoT FND. As shown in Figure 3, FreeMarker takes as input the tunnel provisioning template and data supplied by IoT FND, and generates CLI commands that IoT FND runs on the FARs and HERs in the “configure terminal” context.

Figure 3  CLI Command Generation from Templates in IoT FND

In IoT FND, the tunnel provisioning templates consist of router CLI commands and FreeMarker variables and directives. The use of FreeMarker syntax allows IoT FND to define one template to provision multiple routers.

This section describes the basic FreeMarker syntax in the tunnel provisioning templates. For information about FreeMarker visit http://freemarker.sourceforge.net/.

- Template Syntax
- Data Model

Template Syntax

Table 2 describes the syntax in the default tunnel provisioning templates.

Table 2  Tunnel Provisioning Template Syntax

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Unmarked text is carried through as CG-OS CLI configuration commands for FARs and Cisco IOS CLI commands for HERs.</td>
</tr>
</tbody>
</table>
| Interpolations    | ${variable}  
FreeMarker replaces this construct with the value of a string variable that IoT FND supplies. In this example, IoT FND provides the EID of the FAR:  
description IPsec tunnel to ${far.eid} |
| Default Values    | ${variable!"Default"}  
FreeMarker replaces this construct with the value of a string variable. If the variable is not set, FreeMarker replaces this construct with Default. |
| Conditionals      | <#if condition> output1 <#else> output2 </if>  
FreeMarker uses this construct to determine the text to use in the output. For example:  
<#if far.ipsecTunnelDestAddr1??>  
<#assign destinationAddress=far.ipsecTunnelDestAddr1>  
<#else>  
<#assign destinationAddress=her.interfaces("GigabitEthernet0/0/0")[0].v4.addresses[0].address>  
</#if> |
Table 2    Tunnel Provisioning Template Syntax (continued)

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration over lists</td>
<td><code>&lt;#list as variable&gt; ${variable} &lt;/#list&gt;</code></td>
</tr>
<tr>
<td></td>
<td>FreeMarker uses this construct to iterate over a list.</td>
</tr>
<tr>
<td>Comments</td>
<td><code>&lt;#-- this is a comment --&gt;&gt;</code></td>
</tr>
<tr>
<td></td>
<td>FreeMarker allows comments, but does not retain them in the output.</td>
</tr>
<tr>
<td>Assign statements</td>
<td><code>&lt;#assign name=value&gt;</code></td>
</tr>
<tr>
<td></td>
<td>This construct declares a local variable within the template and assigns a value to it. After that, use this construct to reference the variable:</td>
</tr>
<tr>
<td></td>
<td><code>${name}</code></td>
</tr>
<tr>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;#assign interfaceNumber=0&gt;</code></td>
</tr>
<tr>
<td></td>
<td>…</td>
</tr>
<tr>
<td></td>
<td>interface Tunnel<code>${interfaceNumber}</code></td>
</tr>
<tr>
<td>Macros</td>
<td>These constructs are similar to function calls.</td>
</tr>
<tr>
<td></td>
<td><code>&lt;#macro name(param1,param2, … ,paramN)&gt;</code></td>
</tr>
<tr>
<td></td>
<td>… ${param1} …</td>
</tr>
<tr>
<td></td>
<td>&lt;/#macro&gt;</td>
</tr>
<tr>
<td></td>
<td>Here is an example of a macro definition:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;#macro configureTunnel(interfaceNamePrefix,ospfCost)&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;#assign wanInterface=far.interfaces(interfaceNamePrefix)&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;#if (wanInterface[0].v4.addresses[0].address)?&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;#assign interfaceName=wanInterface[0].name&gt;</code></td>
</tr>
<tr>
<td></td>
<td>interface Tunnel<code>${her.unusedInterfaceNumber()}</code></td>
</tr>
<tr>
<td></td>
<td>description IPsec tunnel to <code>${far.eid}</code></td>
</tr>
<tr>
<td></td>
<td>…</td>
</tr>
<tr>
<td></td>
<td>ip ospf cost <code>${ospfCost}</code></td>
</tr>
<tr>
<td></td>
<td>…</td>
</tr>
<tr>
<td></td>
<td>&lt;/#macro&gt;</td>
</tr>
<tr>
<td>Macro calls</td>
<td>To call macros in a tunnel provisioning template:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;@name param1, param2 ... paramN&gt;</code></td>
</tr>
<tr>
<td></td>
<td>FreeMarker replaces the macro call with the output of the macro after resolving all variables.</td>
</tr>
<tr>
<td></td>
<td>For example:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;@configureTunnel far.tunnelSrcInterface1!&quot;Wimax&quot;, 100/&gt;</code></td>
</tr>
</tbody>
</table>
Managing Tunnel Provisioning

Data Model

This section describes the data model in the tunnel provisioning templates. The `far` and `her` prefixes provide access to the properties of the FARs and HERs, respectively. These properties are stored in the IoT FND database. Table 3 describes referencing the information provided by the data model in tunnel provisioning templates.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>far.eid</td>
<td>Returns the EID of the FAR. For example: <code>${far.eid}</code></td>
</tr>
<tr>
<td>far.hostname</td>
<td>Returns the hostname of the FAR.</td>
</tr>
<tr>
<td>far.tunnelSrcInterface1</td>
<td>Returns the name of the FAR interface on which to establish the tunnel.</td>
</tr>
<tr>
<td>far.ipsecTunnelDestAddr1</td>
<td>Returns the name of the tunnel destination IP address on the HER.</td>
</tr>
</tbody>
</table>
| far.ipv4Address(clientId, linkAddress, userClass) | Returns an IPv4 address. The IPv4 address method takes these parameters as input:  
  - clientId — DCHP Client Identifier for the DHCP request  
  - linkAddress — Link address for the DHCP request  
  - userClass — Value for the DHCP User Class option (defaults to “CG-NMS”)  
  To establish a loopback interface and assign it an address:  
  ```
  interface Loopback0  
  ip address `${far.ipv4Address(dhcpClientId(far.enDuid, 0), far.dhcpV4LoopbackLink).address}/32  
  ipv6 address `${far.ipv6Address(far.enDuid, 0, far.dhcpV6LoopbackLink).address}/128  
  ``` |
| far.ipv4Subnet() | Returns a DHCP IPv4 subnet lease. This call takes a clientId and linkAddress as arguments.  
  Construct the clientId from the FAR EID and interface ID number using the dhcpClientId() method provided in the template API. This method takes as input a DHCPv6 Identity Association Identifier (IAID) and a DHCP Unique Identifier (DUID) and generates the DHCPv4 client identifier, as specified in RFC 4361. This method provides consistency for how network elements are identified by the DHCP server.  
  For example:  
  ```
  <%=assign lease=far.ipv4Subnet(dhcpClientId(far.enDuid,iaId), far.dhcpV4TunnelLink)>
  ``` |
| far.[any device property] | Returns the value of the specified property.  
  For example, far.tunnelSrcInterface1 returns the value of the FAR tunnelSrcInterface1 property. |
| far.interfaces(interfaceNamePrefix) | Returns a list of interfaces discovered from the device that with that prefix (not case sensitive).  
  Use square brackets to index list members for example, [0], [1], [2], and so on. Use the `<#list>` construct to iterate list members.  
  For example:  
  ```
  <%=assign wanInterface = far.interfaces(interfaceNamePrefix)>
  <%=if (wanInterface[0].v4.addresses[0].address)??>
  ``` |
Addresses
Table 4 describes referencing addresses in the tunnel provisioning templates.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>address.address</td>
<td>Returns the address of the interface.</td>
</tr>
<tr>
<td>address.prefixLength</td>
<td>Returns the prefix length of the address.</td>
</tr>
<tr>
<td>address.prefix</td>
<td>Returns the address prefix.</td>
</tr>
<tr>
<td>address.subnetMask</td>
<td>Returns the subnet mask for the address.</td>
</tr>
<tr>
<td>address.wildcardMask</td>
<td>Returns the wildcard mask for the subnet.</td>
</tr>
</tbody>
</table>

Configuring the Field Area Router Tunnel Addition Template

To edit the FAR Tunnel Addition template to provide one end of an IPsec tunnel on FARs in the group:

1. Choose **Config > Tunnel Provisioning**.

2. In the TUNNEL GROUPS pane, select the tunnel group with the template to edit.

3. Click the **Field Area Router Tunnel Addition** tab.

4. Modify the default template.

   **Tip**: Use a text editor to modify templates and copy the text into the template field in IoT FND.

5. Click **Save Changes**.
6. Click **OK** to confirm the changes.

See also, *Tunnel Provisioning Template Syntax*.

**Configuring the Head-End Router Tunnel Addition Template**

*Note: To ensure that both endpoints are in a matching subnet, this template must use the same IAID as the FAR template.*

To edit the HER Tunnel Addition template to create the other end of the IPsec tunnel on HERs in the group:

1. Choose **Config > Tunnel Provisioning**.
2. In the TUNNEL GROUPS pane, select a tunnel group.
3. Click the **Head-End Router Tunnel Addition** tab.
4. Modify the default HER addition template.
5. Click **Save Changes**.
6. Click **OK** to confirm the changes.

**Configuring the HER Tunnel Deletion Template**

To edit the HER tunnel deletion template to delete existing tunnels to FARs at the other end of the tunnel:

1. Choose **Config > Tunnel Provisioning**.
2. In the TUNNEL GROUPS pane, select the tunnel group whose template to edit.
3. Click the **Head-End Router Tunnel Deletion** tab.

4. Modify the default HER deletion template.

5. Click **Save Changes**.

6. Click **OK** to confirm the changes.

**Monitoring Tunnel Status**

To view tunnel status, choose **Operations > Tunnel Status**. The Tunnel Status page lists devices and their provisioned tunnels and displays relevant information about tunnels and their status.
Table 5 describes the tunnel status fields. To change the sort order of tunnels in the list by name, click the HER Name column heading. A small arrow next to the heading indicates the sort order.

Note: It takes time for the status of the newly created tunnel to be reflected in IoT FND.

Table 5 Tunnel Status Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| HER Name           | The EID of the HER at one end of the tunnel. To view the HER details, click its EID.  
| Note:              | Because one HER can serve up to 500 FARs, there may be multiple tunnels in the list with the same HER EID. |
| HER Interface      | The name of the HER tunnel interface. These names are automatically generated when tunnels are created (Tunnel1, Tunnel2, Tunnel3, and so on). |
| Admin Status       | The administrative status of the tunnel (up or down). This indicates if the administrator enabled or disabled the tunnel. |
| Oper. Status       | The operational status of the tunnel (up or down). If the tunnel is down, traffic does not flow through the tunnel, which indicates a problem to troubleshoot. Ping the HER and FAR to determine if they are online, or log on to the routers over SSH to determine the cause of the problem. |
| Protocol           | The protocol used by the tunnel (IPSEC, PIM, or GRE). |
| HER Tunnel IP Address | The IP address of the tunnel at the HER side. Depending on the protocol used, the IP address appears in dotted decimal (IPv4) or hexadecimal (IPv6) slash notation. |
| HER IP Address     | The destination IP address of the tunnel on the HER side. |
| FAR IP Address     | The destination IP address of the tunnel on the FAR side. |
Reprovisioning CGRs

In IoT FND, CGR reprovisioning is a process for modifying the configuration files on CGRs.

- CGR Reprovisioning Basics
- Tunnel Reprovisioning
- Factory Reprovisioning

Note: C800s do not support reprovisioning.

CGR Reprovisioning Basics

- CGR Reprovisioning Actions
- CGR Reprovisioning Sequence

CGR Reprovisioning Actions

In IoT FND, you can perform the following two CGR reprovisioning actions in the Reprovisioning Actions pane of the Tunnel Provisioning page (Config > Tunnel Provisioning).

You can also activate the mesh firmware on this page.

<table>
<thead>
<tr>
<th>Reprovisioning Actions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Factory Reprovisioning</td>
<td>Change the express-setup-config file loaded on the CGR during factory configuration. This file contains a minimal set of information and is loaded on the CGR at the factory. This file provides the CGR with information to contact IoT FND (call home) through the TPS Proxy after the CGR is deployed and powered on.</td>
</tr>
<tr>
<td>Tunnel Reprovisioning</td>
<td>Change the golden-config file on a CGR. This file has the tunnel configuration defined on the CGR.</td>
</tr>
</tbody>
</table>
Table 6 describes the fields on the Reprovisioning Actions pane.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Action</td>
<td>The current reprovisioning action being performed.</td>
</tr>
<tr>
<td>Reprovisioning Status</td>
<td>The status of the reprovisioning action.</td>
</tr>
<tr>
<td>Completed devices /All Scheduled Devices</td>
<td>The number of CGRs that were processed relative to the number of all CGRs scheduled to be processed.</td>
</tr>
<tr>
<td>Error devices/ All Scheduled Devices</td>
<td>The number of CGRs that reported an error relative to the number of all CGRs scheduled to be processed.</td>
</tr>
<tr>
<td>Name</td>
<td>The EID of the CGR.</td>
</tr>
<tr>
<td>Reprovisioning Status</td>
<td>The status of the reprovisioning action for this CGR.</td>
</tr>
<tr>
<td>Last Updated</td>
<td>The last time the status of the reprovisioning action for this CGR was updated.</td>
</tr>
<tr>
<td>Template Version</td>
<td>The version of the Field Area Router Factory Reprovision template being applied.</td>
</tr>
<tr>
<td>Error Message</td>
<td>The error message reported by the CGR, if any.</td>
</tr>
<tr>
<td>Error Details</td>
<td>The error details.</td>
</tr>
</tbody>
</table>

CGR Reprovisioning Sequence

When you start tunnel or factory reprovisioning on a tunnel provisioning group, the reprovisioning algorithm sequentially goes through 12 CGRs at a time and reprovisions them.

After IoT FND reprovisions a router successfully or if an error is reported, IoT FND starts the reprovisioning process for the next router in the group. IoT FND repeats the process until all CGRs are reprovisioned.

There is a timeout of 4 hours when reprovisioning each CGR in the group. If the CGR does not report successful reprovisioning or an error within the timeout period, then IoT FND changes the Reprovisioning Status of the CGR to Error and displays a timeout error and any further information displays in the Error Details field.
Tunnel Reprovisioning

If you make changes to the Field Area Router Tunnel Addition template and want all CGRs already connected to IoT FND reprovisioned with new tunnels based on the modified template, use the tunnel reprovisioning feature of IoT FND.

Tunnel reprovisioning places the CGR in a state where no tunnels are configured, and then initiates a new tunnel provisioning request. To reprovision tunnels, IoT FND sequentially goes through the FARs (12 at a time) in a tunnel provisioning group. For every CGR, IoT FND rolls back the configuration of the CGR to that defined in the ps-start-config template file.

After a rollback to ps-start-config, the CGR contacts IoT FND to request tunnel provisioning. IoT FND processes the Field Area Router Tunnel Addition template and sends the resultant configuration commands for creating new tunnels to the CGR. As shown in Figure 4, the tunnel provisioning process includes updating the golden-config file to include the new configuration information.

**Figure 4  Tunnel Reprovisioning Process**

![Diagram of Tunnel Reprovisioning Process]

1. Roll back
2. Reload
3. Process template and push resulting commands
4. Save

**Note:** For CG-OS CGRs, a rollback results in a reload of the CGR. Also, when IoT FND rolls back a CGR, IoT FND removes the corresponding tunnel information from the HERs to which the CGR was connected.

You perform a configuration replace for Cisco IOS based CGRs.

**Note:** The Field Area Router Factory Reprovision template is not used when performing tunnel reprovisioning.

To configure and trigger tunnel reprovisioning:

1. Choose **Config > Tunnel Provisioning**.
2. In the TUNNEL GROUPS pane, select the tunnel group whose template to provision.
3. Click the **Reprovisioning Actions** tab.
4. From the Action drop-down menu, choose **Tunnel Reprovisioning**.

5. Click **Start**.

IoT FND changes the Reprovisioning Status field to Initialized, and then to Running.

**Note**: If you click **Stop** while tunnel reprovisioning is running, IoT FND stops the reprovisioning process only for the FARs in the queue that were not selected. However, for those CGRs in the queue that were selected for reprovisioning, the process completes (success or error) and cannot be stopped.

The reprovisioning process completes after IoT FND finishes attempting to reprovision each CGR in the tunnel provisioning group. If a CGR cannot be reprovisioned, IoT FND displays the error message reported by the CGR.
Factory Reprovisioning

Use the Factory Reprovisioning feature in IoT FND to change the factory configuration of CGRs (express-setup-config).

Factory Reprovisioning involves these steps:

1. Sending the roll back command to the CGR.
2. Reloading the CGR.
3. Processing the Field Area Router Factory Reprovision template, and pushing the resultant commands to the CGR.
4. Saving the configuration in the express-setup-config file.

After these steps complete successfully, IoT FND processes the Field Area Router Tunnel Addition, Head-End Router Tunnel Addition, and Head-End Router Tunnel Deletion templates and pushes the resultant commands to the CGR (see Tunnel Provisioning Configuration Process).

To configure and trigger factory reprovisioning:

1. Choose **Config > Tunnel Provisioning**.
2. In the TUNNEL GROUPS pane, select the tunnel group whose template you want to edit.
3. Click the **Field Area Router Factory Reprovision** tab and enter the template that contains the configuration commands to apply.

**Note:** The Field Area Router Factory Reprovision template is processed twice during factory reprovisioning; once when pushing the configuration and again before saving the configuration in express-setup-config. Because of this, when making your own template, use the specific if/else condition model defined in the default template.
4. Click Save Changes.

5. If needed, make the necessary modifications to the Field Area Router Tunnel Addition, Head-End Router Tunnel Addition, and Head-End Router Tunnel Deletion templates.

6. Click the Reprovisioning Actions tab.

7. From the Action drop-down menu, choose Factory Reprovisioning.

8. From the Interface drop-down menu, choose the CGR interface for IoT FND to use to contact the FARs for reprovisioning.
9. From the Interface Type drop-down menu, choose **IPv4** or **IPv6**.

10. Click the **Start** button.

   IoT FND changes the Reprovisioning Status field to Initialized, and then to Running.

   **Note:** If you click **Stop** while factory reprovisioning is running, IoT FND stops the reprovisioning process only for the FARs in the queue that were not selected. However, for those CGRs in the queue that were selected for reprovisioning, the process completes and cannot be stopped.

   The reprovisioning process completes after IoT FND has finished attempting to reprovision each CGR in the tunnel provisioning group. If a CGR cannot be reprovisioned, IoT FND displays the error message reported by the CGR.
**Sample Field Area Router Factory Reprovision Template**

This sample template changes the WiFi SSID and passphrase in the factory configuration.

```xml
<#--IMPORTANT: This template is processed twice during factory reprovisioning. The if/else condition described below is needed to determine which part of the template is applied. In this example, if no schedule name wimaxMigrationRebootTimer is found in runningConfig, then the if part of the if/else section is applied. During the second pass, this template runs the commands in the else section and the no scheduler command is applied. If modifying this template, do not remove the if/else condition or else the template fails. -->

<#if !far.runningConfig.text?contains("scheduler schedule name wimaxMigrationRebootTimer")>

<#--Comment: This is a sample of generating wifi ssid and passphrase randomly-->

wifi ssid ${far.randomSSID("PREFIX_")}
authentication key-management wpa2
wpa2-psk ascii ${far.randomPassword(10)}
exit

feature scheduler
scheduler job name wimaxMigration
reload
exit

scheduler schedule name wimaxMigrationRebootTimer
time start +02:00
job name wimaxMigration
exit

<#else>

no scheduler job name wimaxMigration
no scheduler schedule name wimaxMigrationRebootTimer

</#if>
```