CPE Provisioning Overview

This chapter describes the management of customer premises equipment (CPE) using the technologies that the CPE supports for the Prime Cable Provisioning. It features:

Overview

Prime Cable Provisioning provides provisioning and managing of residential devices, namely DOCSIS cable modems and set-top boxes, PacketCable eMTAs, CableHome devices, eRouters, and computers.

Prime Cable Provisioning supports provisioning and managing the following device types:

- Cable modems and STBs compliant with DOCSIS 1.0, 1.1, 2.0, 3.0, and 2.1
- Embedded Multimedia Terminal Adapters (eMTAs) compliant with PacketCable versions
- Devices compliant with CableHome 1.0
- Computers
- eRouters
- Any STB compliant with CableLabs OpenCable Application Platform.
- Variants of eSAFE (embedded Service/Application Functional Entities) devices, such as mixed-IP mode PacketCable Multimedia Terminal Adapters (MTAs). A mixed-IP mode MTA is an eSAFE device that consists of an IPv6 embedded cable modem and an IPv4 eMTA. This class of devices embeds additional functionality with cable modems, such as packet-telephony, home networking, and video.

Device Object Model

The device object model in Prime Cable Provisioning is crucial in controlling the configuration that is generated for the DPE to manage devices. The process of generating a device configuration occurs at the RDU, and is controlled through named attributes and relationships.

The main objects in the device object model are:

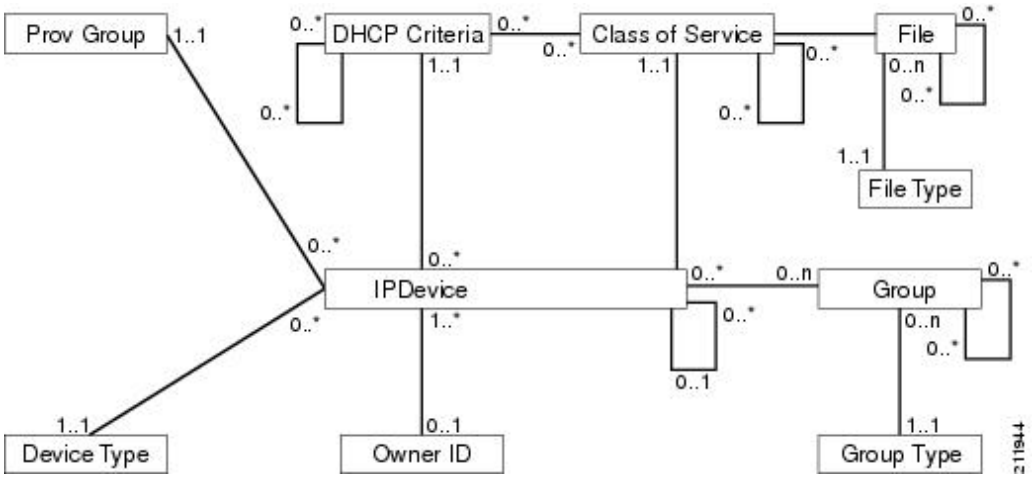
- IPDevice—Represents a network entity that requires provisioning.
- Owner ID—Represents an external identifier for a subscriber.
- Device Type—Represents the type of the device.
- ProvGroup—Represents a logical grouping of devices serviced by a specific set of DPEs.
- Class of Service—Represents the configuration profile to be assigned to a device.
- DHCP Criteria—Represents the criteria for a device to determine the selection of an IP address within the Cisco Prime Network Registrar DHCP server.
- File—Serves as a container for files, including templates, used in provisioning.
- Node—Is a customer-specific mechanism for grouping devices.

Common among the various objects in the Prime Cable Provisioning device data model are:

- Name–For example, Gold Class of Service.
- Attributes–For example, Device ID and a fully qualified domain name (FQDN).
- Relationships–For example, the relationship of a device to a Class of Service.
- Properties–For example, a property that specifies that a device must be in a provisioning group.

The following figure illustrates the interaction among the various objects in the device data model.

Figure 1: Device Object Model



The following table describes the attributes and relationships unique to each object in the data model.

Table 1: Device Object Relationships

Object	Related to...
IPDevice • Could be preprovisioned or self-provisioned (See Device Deployment in Prime Cable Provisioning). • Attributes include Device ID (MAC address or DUID) and FQDN	• Owner ID • Provisioning Group • Class of Service • DHCP Criteria • Device Type
Owner ID • Is associated with devices and, therefore, cannot exist without a device related to it. • Enables grouping; for example, you can group all devices belonging to *Joe*.	IPDevice
Device Type • Stores defaults common to all devices of a technology. • Enables grouping; for example, you can group all PacketCable devices.	IPDevice
File Stores files used in provisioning; for example, configuration files and templates.	Class of Service
Class of Service Attributes include Type, Name, and Properties. (For details, see Class of Service .)	• IPDevice • File • DHCP Criteria • Configuration Template (optional)
DHCP Criteria Enables grouping; for example, you can group devices within a specific technology to different classes of IP.	• IPDevice • Class of Service • Configuration Template (optional)

Class of Service

Class of Service is an RDU abstraction that represents the file configuration to be handed to a device as a static file or as a template file. It enables you to group devices into configuration sets, which are service levels or different packages that are to be provided to the CPE.

The different Classes of Service are:

- Registered—Specified by the user when the device is registered. This Class of Service is explicitly added to the device record via the application programming interface (API).
- Selected—Selected and returned by an RDU extension.
- Related—Related to the device by being registered, selected, or both. This Class of Service is selected by the RDU extensions.

If the selected Class of Service for a device is changed, it regenerates the device configuration. If the registered Class of Service for a device is changed, it regenerates the device configuration even if it is not the selected Class of Service because it could impose a policy that would change the selected Class of Service.

Discovered Data

During the provisioning process, Prime Cable Provisioning uses a set of properties to detect the device type (whether the device is a cable modem, a computer, and so on) and generate the configuration meant for that device type and technology. The information that Prime Cable Provisioning discovers using this set of properties is known as discovered data. Prime Cable Provisioning stores discovered data for each device in the RDU database.

When a device contacts the provisioning server, it provides details about itself, such as its firmware version, MAC address, mode of operation, and so on. In the case of cable modems that contact the provisioning server, these details are made available in the:

- Discover message for IPv4 devices
- Solicit message for IPv6 devices

Prime Cable Provisioning extensions installed on Network Registrar also retrieve discovered data and send it to the RDU when requesting a configuration for a device. For these devices, the discovered data depends on the Network Registrar settings. If an attribute or an option is configured for use in Network Registrar, then the extensions fetch the value for that attribute or option from the DHCP packet and include it as part of the data discovered for provisioning Prime Cable Provisioning.

The following table lists the data that Prime Cable Provisioning discovers for IPv4 devices.

Table 2: Data Discovered from IPv4 Devices

Option	Description
chaddr	Specifies the hardware address of the client
client-id	Identifies a sequence of bytes or a string defined on the client that uniquely identifies the client
client-id-created-from-mac-address	Identifies the client identifier that is created from the MAC address of the client
dhcp-message-type	Specifies the type of DHCP message, such as DHCP Discover, DHCP Ack, and so on

Option	Description
giaddr	Specifies the IP address to which the DHCP server should reply
hlen	Specifies the length of the hardware address
htype	Specifies the hardware type
relay-agent-circuit-id	Encodes an agent local identifier of the circuit from which a DHCP client-to-server packet is received
relay-agent-info	Used in accessing the CableLabs Relay Agent CMTS Capabilities Option
relay-agent-remote-id	Encodes information about the remote host end of a circuit
v-i-vendor-opts	Identifies the options requested by the client from the server
vendor-encapsulated-options	Defines options that are sent encapsulated in a standard DHCP option
vendor-class	Contains a string identifying capabilities of the DHCPv4 client and associated CPE

The following table lists the data that Prime Cable Provisioning discovers for IPv6 devices.

Table 3: Data Discovered from IPv6 Devices

Option	Description
peer-address	Specifies the IPv6 address of the client that originally sent the message or the previous relay agent that relayed the message
link-address	Specifies the non-link-local address that is assigned to an interface connected to the client subnet
client-identifier	Specifies the DHCP Unique Identifier (DUID) of the client for the lease. Because the client hardware address (chaddr) is not available for DHCPv6 clients, a DUID is used to uniquely identify a device in an IPv6 environment. This information is made available in a DHCP Solicit message.
oro	Identifies the options requested
vendor-opts	Identifies the vendor-specific information option that is used by clients and servers to exchange vendor-specific information. This information is made available in a DHCP Solicit message.

Option	Description
vendor-class	Identifies the vendor that manufactured the hardware on which the client is running. This information is made available in a DHCPv6 Solicit message.

You can view discovered data using the administrator user interface on the Device Details page. For more information on viewing device details, see Viewing Device Details.

For a list of properties that Prime Cable Provisioning extensions use to discover data for DHCPv4 and DHCPv6, see Configuring Prime Network Registrar Extension.

DUID versus MAC Address

The DHCPv4 standard uses the client identifier, or the MAC address, as the primary device identifier for DHCP clients. DHCPv6 introduces a new primary device identifier: the DHCP Unique Identifier (DUID).

DHCPv4 uses the hardware address and an optional client identifier to identify the client for assigning an address. DHCPv6 basically follows the same scheme but makes the client identifier mandatory, consolidating the hardware address and the client ID into one unique client identifier.

The client identifier in DHCPv6 consists of:

- DUID—Identifies the client system (rather than just an interface, as in DHCPv4).
- Identity Association Identifier (IAID)—Identifies the interface on that system. As described in RFC 3315, an identity association is the means used for a server and a client to identify, group, and manage a set of related IPv6 addresses.

Each DHCP client and server has a DUID. DHCP servers use DUIDs to identify clients to select configuration information and in the association of IAs with clients. DHCP clients use DUIDs to identify a server in messages where a server needs to be identified.

Configuration Generation and Processing

When a device is activated in a Prime Cable Provisioning deployment, it initiates contact with the Prime Cable Provisioning server. Once contact is established, the device's preconfigured policy, based on configuration templates associated with the device, determines the DPE’s provisioning and managing of the device. Authoritative provisioning information for the device is forwarded to DPEs from the RDU as a device configuration. The DPE caches the device configuration and uses it to service requests from the device.

Device configurations can include customer-required provisioning information such as:

- DHCP IP address selection
- Bandwidth
- Data rates
- Flow control
- Communication speeds
- Level of service (also known as Class of Service)

A configuration includes an identifier (a MAC address or file name) and a revision number that is incremented each time the configuration is regenerated.

The RDU regenerates the configuration for a device when:

- Certain provisioning API calls, such as changing the device Class of Service, are made.
- Validation for a configuration fails. This occurs, for example, when certain parameters of a DHCP request from a device change from initial request parameters.

Every time the RDU regenerates a configuration for a device, the updated configuration is forwarded to the appropriate DPEs and cached.

This section also describes these related concepts:

- Static Files versus Dynamic Files
- Property Hierarchy
- Templates and Property Hierarchy
- Custom Properties

Static Files versus Dynamic Files

You can provision devices with Prime Cable Provisioning using two types of configuration files: static files and dynamic files.

When using static configuration files, you enter them into the Prime Cable Provisioning system. They are then delivered via TFTP to the specific device to generate its configuration. Prime Cable Provisioning treats static configuration files like any other binary file. Static files are identified by a *.cm* extension.

Dynamic files can be generated from either templates or Groovy scripts.Templates are text files containing DOCSIS, PacketCable, or CableHome options and values that, when used with a particular Class of Service, provide dynamic file generation. Prime Cable Provisioning ships with a configuration file utility that helps you test, validate, and view configuration and template files for DOCSIS, PacketCable, and CableHome. For detailed information on using the configuration file utility, see Using Configuration File Utility for Template. Template files are identified by a *.tmpl* extension.

For a summary of static provisioning versus dynamic provisioning, see Static versus Dynamic Provisioning.

Property Hierarchy

Prime Cable Provisioning properties provide a means to access and store data in Prime Cable Provisioning via the API. Preprovisioned, discovered, and status data can be retrieved via properties of corresponding objects via the API. Properties also enable configuration of Prime Cable Provisioning at the appropriate level of granularity (from system level to device group and to individual device).

Device-related properties can be defined at any acceptable point in the Prime Cable Provisioning property hierarchy. For details on whether you can assign the property at any level, see the API Javadoc.

The Prime Cable Provisioning property hierarchy gives you the flexibility to define properties for individual devices or groups of devices. The properties are looked up on a device and its associated objects until they are found in the following order:

1. Device registered properties—Specifies properties configured via the API or the administrator user interface.
2. Device selected properties—Specifies properties that are stored on the device record by the service-level selection process.
3. Group—Specifies properties of the group to which the device is related.
4. Device-detected properties—Specifies properties that are stored on the device record by the device detection process.
5. Provisioning Group—Specifies properties of a device's provisioning group.
6. Class of Service—Specifies properties that are configured on a device's Class of Service. If the service-level selection process determines a Selected Class of Service for a device, the properties from that object are used. Otherwise, the properties are looked up from the Registered Class of Service configured for a device via the API or the administrator user interface.
7. DHCP Criteria—Specifies properties that are configured on a device's DHCP Criteria. If the service-level selection process determines a Selected DHCP Criteria for a device, the properties from that object are used. Otherwise, the properties are looked up from the Registered DHCP Criteria configured for a device via the API or the administrator user interface.
8. Technology Defaults—Specifies the properties that are configured in the device's technology defaults. For example, technology defaults for DOCSIS modems, PacketCable MTAs, or computers.
9. System Defaults—Specifies the properties that are configured in system defaults.

Templates and Property Hierarchy

Generating configurations dynamically involves processing the text description of a device configuration file (which is also known as a template) into a binary device configuration. The binary configuration file is essentially a list of type-length-value (TLV) tuples, each of which contains a device configuration setting. The resulting binary configuration is then forwarded via TFTP to the device.

Dynamic configuration generation offers immense flexibility using a macro capability. Macros allow values from the Prime Cable Provisioning property hierarchy to be substituted into templates. This substitution is used for values that are commonly overridden, such as:

- Downstream or upstream bandwidth
- Number of devices behind a cable modem

In this way, Prime Cable Provisioning uses a single template to generate configuration from a few templates to any number of devices.

Scripts and Property Hierarchy

The Dynamic Configuration File Generation with Groovy scripting offers increased functionality over template-based file generation.

To support dynamic configuration, the Groovy script uses device-discovered data, at run time, through APIs. Using the bindings that are passed to the Groovy script, variables can be substituted with values from the Prime Cable Provisioning property hierarchy.

Similar to templates, Prime Cable Provisioning uses scripts to generate configuration for any number of devices.

Custom Properties

Prime Cable Provisioning allows you to define new properties within the RDU that can then be stored on any object via the API. These properties enable substitution of values into templates.

Custom properties are variable names defined in the RDU, and must not contain any spaces.

For details on how to create custom properties, see Configuring Custom Properties.

Device Deployment in Prime Cable Provisioning

A Prime Cable Provisioning deployment is divided into provisioning groups, with each provisioning group responsible only for a subset of the devices. All services provided by the provisioning group are implemented to provide fault tolerance (see Provisioning Groups).

Prime Cable Provisioning provides two device deployment options:

- Preprovisioned—The RDU is populated with configurations and rules for the various device types. When the device record is added to the RDU, it maps to a configuration specific to the device type.
- Self-provisioned—The device makes first contact with the provisioning group before the device record is added to the RDU. The preprovisioned rules, however, determine the configuration of the device.

CPE Registration Modes

Registration modes allow the service provider to control the number of interactions with the subscriber. For any registered device, the service provider must be prepared to process any change to the device. There is a significant difference between registering 100 cable modems with unregistered computers behind them, and registering 100 cable modems, each of which has a potentially large number of registered computers behind it. For this reason, the service provider must carefully choose among the standard, promiscuous, roaming, and mixed modes.

Standard Mode

When operating in the standard mode (sometimes called the fixed mode), a computer is registered and, when it is behind the correct cable modem, it receives registered access. When it is moved behind a different cable modem, however, it receives unprovisioned access.

Promiscuous Mode

When operating in the promiscuous mode, only DOCSIS modems are registered; the DHCP server maintains lease information about a device operating behind another device. All devices of specified types behind a registered device receive network access.

Roaming Mode

When operating in the roaming mode, a registered device receives its assigned service behind any other registered device. For example, this mode permits the use of a laptop moving from location to location and obtaining service from multiple cable modems.

Mixed Mode

When operating in the mixed mode, any mode is used at any time in a single deployment (with different devices).

CPE Provisioning Flows

This section describes the provisioning workflows for devices:

- Initial Configuration Workflows
- Configuration Update Workflow

Initial Configuration Workflows

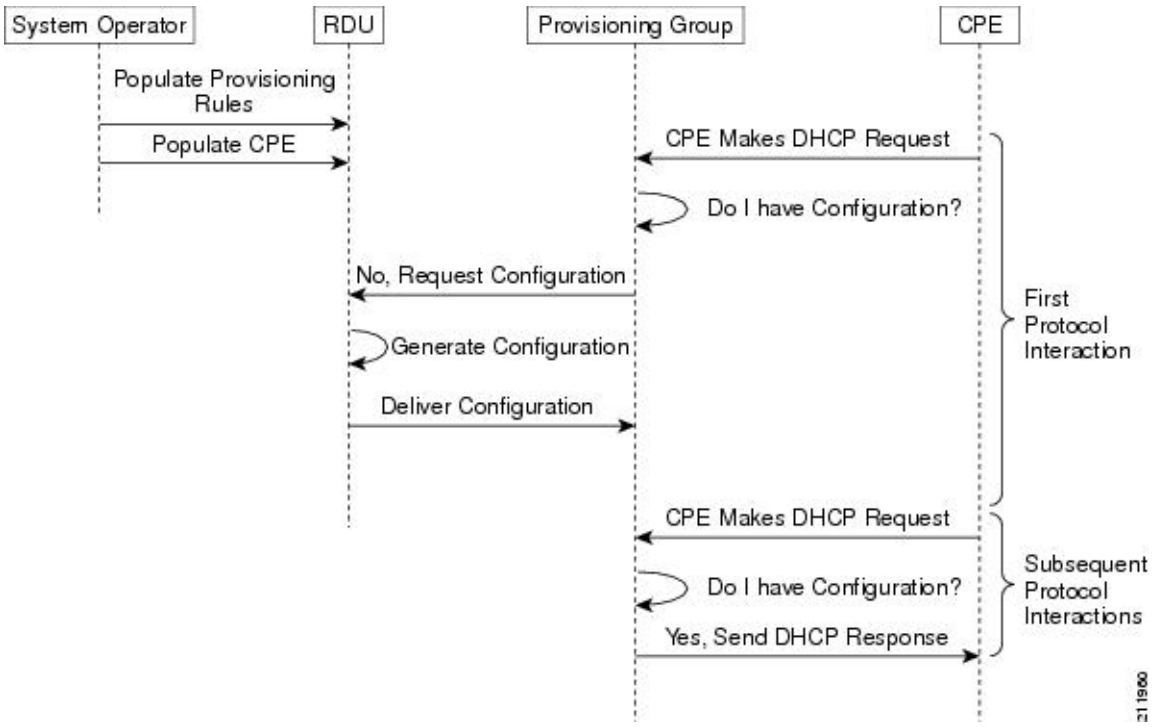
This section describes the configuration workflow when a device is initially installed and booted. The workflows differ based on deployment and registration mode and include:

- Preprovisioned Device Workflow
- Self-Provisioned Device Workflow

Preprovisioned Device Workflow

This section describes the workflow for a preprovisioned device. The following figure shows a common initial configuration workflow.

Figure 2: Workflow of Initial Device Configuration – Preprovisioned Mode



1. From the Prime Cable Provisioning API, the RDU is populated with specifically defined configurations and rules for various types of devices. The device is preconfigured and associated with a Class of Service, and preregistered in the RDU database.

Note Preconfiguring CPE involves populating the device information, such as the MAC address and the Class of Service, in Prime Cable Provisioning via the API. In the preprovisioned mode, this task occurs before the device has booted on the network and in the self-provisioned mode, this task occurs after the device has booted on the network.

2. When the device is booted, it discovers its provisioning group and initiates its autoprovisioning flow with DPEs in the provisioning group. The cable modem termination system (CMTS) relays broadcast traffic to the DHCP server. It is the DHCP server, or Prime Cable Provisioning extensions on the Network Registrar DHCP server, that requests configurations from the DPE.

Note When a device roams to a new provisioning group using the roaming mode, it goes through a similar flow except that its old configuration is removed from the provisioning group that it used to belong to.

3. The DPE, on receiving the device request, looks up its cache for a configuration for the device. Because the device has never previously contacted the provisioning group, no configuration is found. The Network Registrar extensions in the provisioning group then request the RDU to generate a configuration for the device.

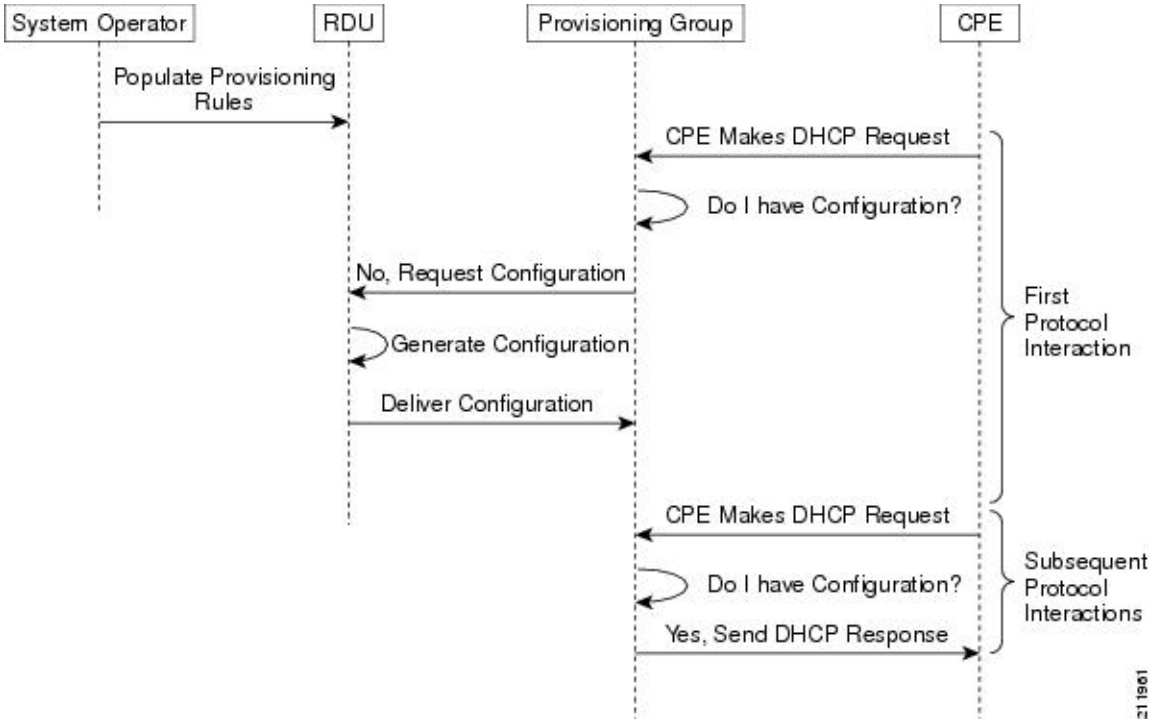
Depending on the time the RDU takes to process the request, the provisioning group may decide not to respond to the device request.

4. The RDU generates a configuration appropriate for the device. The resulting device configuration directs DPE responses to various CPE protocol events, such as a DHCP Discover.
5. The device configuration is forwarded to the DPE and cached there. Now, the DPE is programmed to handle subsequent CPE protocol interactions for the device autonomously from the RDU. Once the device is added to the network and a configuration is generated for the device, the device boots to allow the DPE to begin its interactions with the preregistered device.
6. During interactions with the device, additional information can be discovered and forwarded to the RDU. In this case, the RDU may decide to generate new configurations and forward them to all DPEs.

Self-Provisioned Device Workflow

This section describes the workflow for a self-provisioned device. The following figure shows a common initial configuration workflow.

Figure 3: Workflow of Initial Device Configuration – Self-Provisioned Mode



1. From the Prime Cable Provisioning API, the RDU is populated with specifically defined configurations and rules for various types of devices.

Note Preconfiguring CPE involves populating the device information, such as the MAC address and the Class of Service, in Prime Cable Provisioning via the API. In the self-provisioned mode, this task occurs after the device has booted on the network.

2. When the device is booted, it discovers its provisioning group and initiates its autoprovisioning flow with DPEs in the provisioning group. The cable modem termination system (CMTS) relays broadcast traffic to the DHCP server. It is the DHCP server, or Prime Cable Provisioning extensions on the Network Registrar DHCP server, that requests configurations from the DPE.

Note When a device roams to a new provisioning group using the roaming mode, it goes through a similar flow except that its old configuration is removed from the provisioning group that it used to belong to.

3. The DPE, on receiving the device request, looks up its cache for a configuration for the device. Because the device has never previously contacted the provisioning group, no configuration is found. The Network Registrar extensions in the provisioning group then request the RDU to generate a configuration for the device.

   Depending on the time the RDU takes to process the request, the provisioning group may decide not to respond to the device request.

4. The RDU generates a configuration appropriate for the device. The resulting configuration directs DPE responses to various CPE protocol events, such as a DHCP Discover.

5. The device configuration is forwarded to the DPE and cached there. Now, the DPE is programmed to handle subsequent CPE protocol interactions for the device autonomously from the RDU. Once the device is added to the network and a configuration is generated for the device, the device boots to allow the DPE to begin its interactions with the preregistered device.

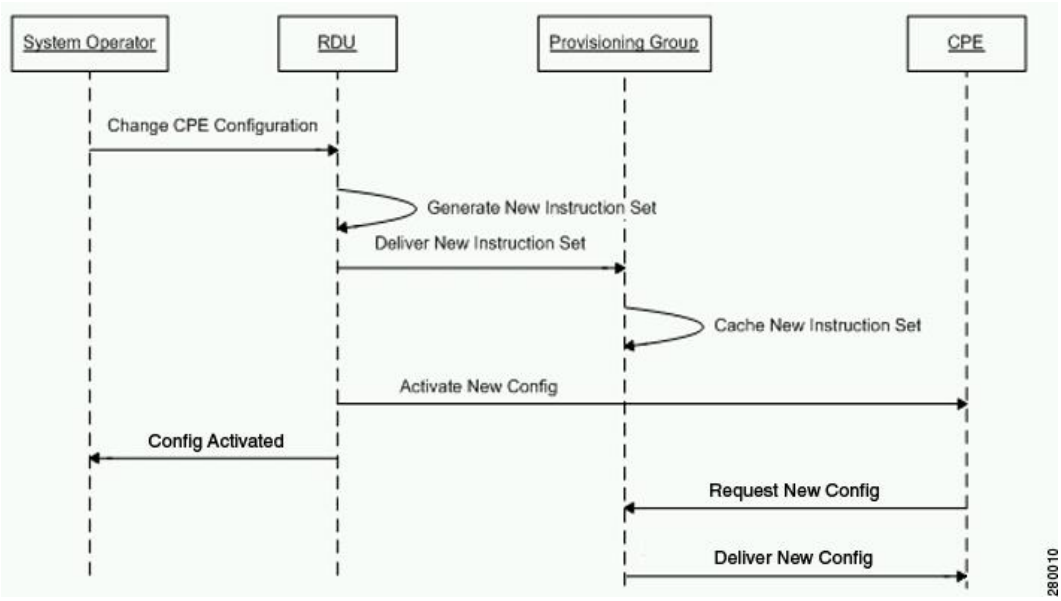
6. During interactions with the device, additional information can be discovered and forwarded to the RDU. In this case, the RDU may decide to generate new configurations and forward them to all DPEs.

Configuration Update Workflow

This section describes the workflows when a device configuration is updated.

The following figure shows the common configuration workflow when you change the configuration of a device that was previously configured.

Figure 4: Workflow of Device Configuration Update



1. From the Prime Cable Provisioning API, the device configuration at the RDU is updated.
2. The RDU generates a configuration for the device and delivers it to the DPEs in the provisioning group to which the device belongs.
3. The DPE caches the new configuration.
4. The RDU instructs the DPE to forward the new configuration to the device.
5. The RDU does an SNMP Set on the modem or MTA, causing the device to reboot.

Restrict number of CPEs behind CM

In addition to providing warning for CPE addition behind Cable Modem through the property, */rdu/log/cpeCountOnUpdate/enable*, you can restrict adding CPE behind Cable Modem using the property */rdu/cpe/restrict/enable*. The device count threshold value can be set using */rdu/log/cpe/threshold* in *rdu.properties*, beyond that assigned threshold value, the property */rdu/cpe/restrict/enable* will restrict adding CPEs. After adding these properties, you need to restart the RDU.

Table 4:

Property	Description

When only restriction property is added
 	
`/rdu/cpe/restrict/enable =true`
`/rdu/log/cpe/threshold = <integer>`
 	
Eg: **/rdu/log/cpe/threshold =3**
	Devices more than the threshold value will not be allowed to add behind the CM and **BATCH_ERROR** will be added in **rdu.log**.

Property	Description

When only warning property is added
 	
`/rdu/log/cpeCountOnUpdate/enable =true`
`/rdu/log/cpe/threshold=<integer>`
 	
For eg: /rdu/log/cpe/threshold =2
	Device will get added behind CM but **BATCH_WARNING** will be added in **rdu.log**.

`/rdu/log/cpeCountOnUpdate /enable =true`
`/rdu/cpe/restrict/enable =true`
`/rdu/log/cpe/threshold=<integer>`
 	
For eg: /rdu/log/cpe/threshold =3
	BATCH_WARNING will be added when the threshold value is reached and **BATCH_ERROR** will be added when the devices are added more than the threshold value.