Cisco Prime Cable Provisioning 5.0 User Guide

March 15, 2013

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number: OL-26274-01

Cisco Prime Cable Provisioning 5.0 User Guide

CONTENTS

Preface

Welcome to the *Cisco Prime Cable Provisioning 5.0 User Guide*. This guide describes concepts and configurations related to Cisco Prime Cable Provisioning.

Note For a complete understanding of the product, use this guide along with the documentation listed in Product Documentation, page xiii, and Related Documentation, page xiii.

Audience

System administrators use this guide to configure Prime Cable Provisioning for automating large-scale provisioning for broadband access. The administrator should be familiar with:

- Basic networking concepts and terminology
- Network administration
- Cable networks

Product Documentation

Note We sometimes update the documentation after original publication. Therefore, you should also look at the documentation on Cisco.com for any updates.

You can view the marketing and user documents for Prime Cable Provisioning at:

http://www.cisco.com/en/US/products/ps12728/tsd_products_support_series_home.html

See the *Cisco Prime Cable Provisioning 5.0 Documentation Overview* for the list of Prime Cable Provisioning guides.

Related Documentation

See the *Cisco Prime Network Registrar 8.x Documentation Overview* for the list of Prime Network Registrar guides.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see *What's New in Cisco Product Documentation* at: http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html.

Subscribe to *What's New in Cisco Product Documentation*, which lists all new and revised Cisco technical documentation, as an RSS feed and deliver content directly to your desktop using a reader application. The RSS feeds are a free service.

PART 1

Getting Started

This part contains the following sections:

- Introducing Prime Cable Provisioning
- Managing Licenses
- Understanding Prime Cable Provisioning Architecture
- Accessing Prime Cable Provisioning Interfaces
- Managing Database

CHAPTER 1

Introducing Prime Cable Provisioning

Cisco Prime Cable Provisioning automates the tasks of provisioning and managing customer premises equipment (CPE) in a broadband service-provider network.

With the high-performance capabilities of Prime Cable Provisioning, you can scale the product to suit networks of virtually any size, even those with millions of devices. It also offers high availability, which is made possible by the product's distributed architecture and centralized management.

Prime Cable Provisioning is designed to handle the rapid growth of service providers. It targets broadband service providers (including multiple service operators), internet, and voice service providers who want to deploy IP data, voice, and video on hybrid fiber and coaxial cable networks.

Prime Cable Provisioning provides such critical features as redundancy and failover. It can be integrated into new or existing environments through a provisioning application programming interface (API) that lets you control how Prime Cable Provisioning operates. You can use the provisioning API to register devices in Prime Cable Provisioning, assign device configurations, and configure the entire Prime Cable Provisioning provisioning system.

Technologies and Features

This section describes the technologies and features that Prime Cable Provisioning supports.

- Supported Technologies and Standards, page 1-1
- Supported Devices, page 1-2
- Key Features and Benefits, page 1-5
- What’s New in This Release, page 1-7

Supported Technologies and Standards

Prime Cable Provisioning facilitates support for many technologies to provide provisioning services for your network. These technologies include:

- DOCSIS High-Speed Data, page 1-2
- PacketCable Voice Services, page 1-2,
- CableHome, page 1-3

Supported Devices

Prime Cable Provisioning supports provisioning and managing of:

- IPv4 and IPv6 devices, which include:
  - Cable modems and STBs compliant with DOCSIS 1.0, 1.1, 2.0 and 3.0. Only DOCSIS 3.0 supports IPv6 devices.
  - Embedded Multimedia Terminal Adapters (eMTAs) compliant with PacketCable 1.0, 1.5 and 2.0. Only PacketCable 2.0 supports IPv6 devices.
  - Devices compliant with CableHome 1.0
  - Computers
  - Set-top boxes (STBs)
- 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.

DOCSIS High-Speed Data

The Data Over Cable Service Interface Specification (DOCSIS) defines functionality in cable modems that are involved in high-speed data distribution over cable television system networks. Using this feature, Multiple Systems Operators (MSOs) can provide a variety of services through an “always-on” Internet connection. These services include broadband Internet connectivity, telephony, real-time interactive gaming, and video conferencing.

Prime Cable Provisioning, besides supporting DOCSIS 1.0, 1.1, and 2.0, provisions and manages CPE that is compliant with DOCSIS 3.0. The DOCSIS 3.0 specification defines the third generation of high-speed data-over-cable systems specification and provides for:

- Provisioning of IPv6 devices
- Expanded addressability of network elements
- Increased channel capacity via channel bonding
- Enhanced network security
- Enhanced multicast capabilities
- New service offerings

PacketCable Voice Services

PacketCable voice technology enables the delivery of advanced, real-time multimedia services over a two-way cable network. PacketCable is built on top of the infrastructure supported by cable modems to enable a wide range of multimedia services such as IP telephony, multimedia conferencing, interactive gaming, and general multimedia applications.

Using PacketCable voice technology, you can provide additional services, such as basic and extended telephony services, in a broadband network. For this purpose, PacketCable is an efficient and cost-effective option.

Prime Cable Provisioning supports the Secure and Basic variants of PacketCable and both these modes are much the same, except for reduced security found in the Basic variant. Prime Cable Provisioning supports PacketCable 1.0, 1.5, and 2.0 specifications.

Euro-PacketCable services are the European equivalent of the North American PacketCable standard. The only significant difference between the two is that Euro-PacketCable uses different MIBs.

CableHome

Non-secure CableHome 1.0 provisioning (hereafter referred to as home networking technology) is built on top of the existing DOCSIS standard and supports a ‘plug-and-play’ environment for residential broadband connectivity. This form of home networking technology encompasses a DOCSIS home access device with support for CableHome. This device is known as Portal Services and is considered to be the home’s entry point.

Supported Standards

Prime Cable Provisioning complies with these applicable Requests for Comments (RFCs), protocols, standards, and Internet Engineering Task Force (IETF) drafts:

- DHCPv6—Complies with RFC 3315 (DHCPv6 specification), 3633 (IPv6 Prefix Options), 3736 (Stateless DHCP Service for IPv6), 4014 (Remote Authentication Dial-In User Service–RADIUS–Attributes Suboption for the Relay Agent Information Option), 4580 (Relay Agent Subscriber-ID Option), 4649 (Relay Agent Remote-ID Option), and 4704 DHCPv6 Client Fully Qualified Domain Name (FQDN) Option.
- IPv6—Complies with RFC 2460 (IPv6 specification), 2461 (Neighbor Discovery Protocol), 2462 (Stateless Address Autoconfiguration), 2463 (Internet Control Message Protocol–ICMP), 3513 (Addressing Architecture).
- IPv4 and IPv6 interoperability—Complies with RFC 4038 (Application of IPv6 Transition) and 4472 (Operational Issues and Considerations with IPv6 DNS).
- TFTP and ToD servers—Complies with RFC 868 (Time Protocol), 2348 and 2349 (TFTP Blocksize Options), 1350 (TFTP Revision 2 protocol) and 2347 (TFTP Option Extension).

Additionally, Prime Cable Provisioning complies with these applicable CableLabs standards:

- Cross Project
  - CL-SP-CANN-I06-111117
  - CL-SP-CANN-DHCP-Reg-I08-111117
- DOCSIS
  - eDOCSIS
    - CM-SP-eDOCSIS-I23-120329

  - DOCSIS 2.0
    - CM-SP-RFIv2.0-C01-081104
    - CM-SP-DOCSIS2.0-IPv6-I05-111117
  - DOCSIS 3.0
    - CM-SP-MULPIv3.0-I18-120329
    - CM-SP-SECv3.0-I13-100611
    - CM-SP-OSSIv3.0-I18-120329
  - Business Services over DOCSIS
    - CM-SP-L2VPN-I09-100611
  - DOCSIS Set-top Gateway (DSG)
    - CM-SP-DSG-I20-120329
- DOCSIS Provisioning of EPON (DPoE)
  - DPoE-SP-MULPIv1.0-I03-120830
  - DPoE-SP-OSSIv1.0-I03-120830
  - DPoE-SP-SECv1.0-I02-120607
- PacketCable
  - PacketCable 1.5
    - PKT-SP-PROV1.5-I04-090624
    - PKT-SP-SEC1.5-I03-090624
  - PacketCable 2.0
    - PKT-SP-EUE-PROV-I07-110825
    - PKT-SP-EUE-DATA-I08-121030
    - PKT-SP-UE-PROV-I02-100527
    - PKT-SP-UE-DATA-I02-100527
    - PKT-SP-RST-EUE-PROV-I08-121030
    - PKT-SP-RST-UE-PROV-I02-100527
    - PKT-SP-RST-E-DVA-I10-121030
- OpenCable
  - OC-SP-HOST2.1-CFR-I15-120112
- CableHome
  - CH-SP-CH1.0-C01-060728
  - CH-SP-CH1.1-C01-060728

Key Features and Benefits

Table 1-1 outlines the important features and benefits of Prime Cable Provisioning.

Table 1-1 Prime Cable Provisioning Features and Benefits

Feature	Benefit
Easy integration with back-end systems	• The Prime Cable Provisioning Java API, which can be used to perform all provisioning and management operations. It also provides easy integration to customer OSS, billing, or workflow and mediation software. • The Prime Cable Provisioning publishing extensions, which are useful in writing RDU data into another database. • The SNMP agent, which simplifies integration for monitoring Prime Cable Provisioning. • The DPE command-line interface (CLI), which allows you to configure the DPE to suit your requirements via a "services" interface, and which simplifies local configuration when you use the CLI to copy and paste commands.
Improved management	• Provisioning group capabilities—Allows you to control the device type support that must be enabled for the provisioning groups in your deployment. • Property hierarchy—For better flexibility, Prime Cable Provisioning property hierarchy allows you to define properties at different levels.
Increased security	• User-configurable IP addresses and ports to provide multipathing, multi-interface binding, and firewall compatibility. • DOCSIS 3.0 for the Extended CMTS MIC Configuration Setting, enabling Prime Cable Provisioning to use advanced hashing techniques to detect unauthorized modification or corruption of the cable modem configuration file. • A password policy to access the RDU from the Administrator user interface. The Radius authentication provides increased security by authenticating the users accessing the network services via the Radius server, using the Radius standard protocol. • Secure access, enhanced administrator user interface access over HTTPS.

Table 1-1 Prime Cable Provisioning Features and Benefits (continued)

Enhanced troubleshooting and diagnostics	• Device troubleshooting to provide detailed records of device interactions with Prime Cable Provisioning servers using the IDs of the devices designated for troubleshooting. Using this feature, you can focus on a single device, identified by its MAC address or its DHCP Unique Identifier (DUID), and use that diagnostic information for further analysis. • Server troubleshooting using diagnostics scripts to collect performance statistics—down to a specific type of statistic—for the servers. Prime Cable Provisioning provides many scripts to collect server and system configuration data that may be required for support escalations. You can use the bundleState script to collect the diagnostics data.
DOCSIS 3.0 and IPv6 support	DOCSIS 3.0 channel bonding allows increased data speed for subscribers. Support for IPv4 and IPv6 cable modems and IPv4/IPv6 mixed device environment.
Distributed architecture with high availability and disaster recovery	Offers true scalability, failover, and high reliability to manage a growing subscriber base while helping to ensure minimum subscriber service disruption. Allows a simple way to extend provisioning to additional subscribers and new markets, and dramatically simplifies capacity upgrade and lowers maintenance costs. Distributed provisioning engines allow you to put them in different data centers for disaster recovery.
Integrated Kerberos Protocol server (KDC) for PacketCable voice service provisioning	Provides a single platform with all the necessary security components for PacketCable provisioning.
Templates and MACRO for better flexibility	You can include an existing template and use MACRO for better flexibility in managing template parameters and in automating the template deployment.
Technology extensions	Provides an easy means to extend this single platform to provision new devices and technologies to meet changing network and subscriber requirements.
PacketCable compliant	Supports PacketCable 1.0, 1.1, 1.5 and 2.0 specifications for complete end-to-end IP voice service provisioning and meets all PacketCable security specifications.
Dynamic file generation	Offers a means to build unique files for individual subscriber devices to meet needs of tiered service provisioning and true IP voice requirements.
Safe failover	High uptime and service reliability through DPE and DHCP failover as well as TFTP redundancy.

What’s New in This Release

- New Web UI:
  - Cisco Broadband Access Center has been re-branded as Prime Cable Provisioning and offers a new Prime Admin UI that is similar to other Cisco Prime products and ensures ease of use.
- 64-bit support:
  - In earlier versions of Prime Cable Provisioning, the RDU and DPE servers were shipped with 32-bit processing capabilities with reliability and scalability limitations. A 32-bit application could address only a 4-GB address process space. To mitigate this performance issue, Prime Cable Provisioning is now enhanced to 64-bit processing capabilities, which also includes 64-bit JVM migration. With RDU and DPE servers supporting 64-bit mode, JRE can have expanded heap sizes that allows better leverage of existing resources.
    Prime Cable Provisioning supports migration of 32-bit RDU and DPE to 64-bit. Also, 64-bit RDU is backward compatible with 32-bit DPE and other components. The CNR-EP continues to be in 32-bit.
- RDU Redundancy:
  - The RDU redundancy feature involves setting up the RDU in High Availability (HA) mode where a two node failover pair is configured for the RDU.
- CableLab enhancements:
  - Supports PacketCable 2.0 on IPv4 mode on both Basic and Secure mode.
  - Supports DPoE 1.0 standard.
- Role Based Access Control:
  - Provides better user management and security with the help of finer grain privileges.
  - Provides an admin role as well as a set of roles that partition the responsibility of managing Prime Cable Provisioning, such as COS Admin Role, File Admin Role, Provisioning Group Admin Role, Device Admin Role.
- Secure connection through SSL:
  - Secures all inbound communication within Prime Cable Provisioning components through the use of Secure Socket Layer (SSL) 3.0 or and TLS 1.0 protocols.
  - Supports encryption of default and custom properties.
  - Supports both signed and unsigned certificates.
- Provisioning Web Services Interface:
  - Provides a SOAP based web service interface that facilitates device provisioning functions.
  - Operates in both secure and non-secure mode.

CHAPTER 2

Managing Licenses

Software licenses are used to activate specific features or to increase the functionality of your installation. This chapter describes Prime Cable Provisioning handling of different licenses. For details on the licensing changes in this release and how to obtain your license file, see the *Cisco Prime Cable Provisioning 5.0 Quick Start Guide*. To access Prime Cable Provisioning 5.0, you must procure a new license for this release. You cannot use any old licenses of any earlier releases.

Prime Cable Provisioning licenses are available either as a permanent or an evaluation license.

- Permanent—A permanent license is purchased for use in your network environment and activates the specific features for which it is intended.
- Evaluation—An evaluation license enables functionality for a specific length of time.

  Prime Cable Provisioning evaluation licenses become invalid on a predetermined date. As such, evaluation licenses must be created when needed. To create your evaluation license, contact your Cisco representative, who will generate the necessary license file online and forward it to you via e-mail.

Prime Cable Provisioning enables licensing using a Service License file. Each license translates to a DOCSIS IP device. The license file that you receive will contain the number of DOCSIS IP devices that are licensed.

Prior to Prime Cable Provisioning 5.0, every device type instance such as, DOCSIS, Computer, PacketCable, CableHome, etc. consumed a separate license. In addition, with dual-stack Computers, this approach created two IPDevice records, one each for IPv4 and IPv6, resulting in two licenses being consumed instead of one. In Prime Cable Provisioning 5.0, the licensing scheme counts only the DOCSIS IP devices irrespective of whether the device is a stand-alone CM or an embedded eCM, each DOCSIS IP device consumes one license. Apart from the DOCSIS IP device, all other device types consume no license. In other words, Prime Cable Provisioning 5.0 consumes only one license:

- if a subscriber has a CM to which a Computer and a PacketCable devices are connected.
- if a subscriber has a CM to which a dual-stack Computer and a PacketCable devices are connected.
- if a subscriber has an e-MTA or e-STB.

To know how the license mechanism for Prime Cable Provisioning 5.0 is different from BAC 4.x, see the BAC 4.x and Prime Cable Provisioning 5.0 Licensing Model document.

The Prime Cable Provisioning Admin UI shows the available and used licenses. You can also see the available and used licenses count through the API client.

Caution Do not edit your license file. Changing the data in any way invalidates the license file.

Note You still require separate licenses for the following Prime Cable Provisioning components:

- The DPE
- The KDC, if you configure your network to support secured voice technology

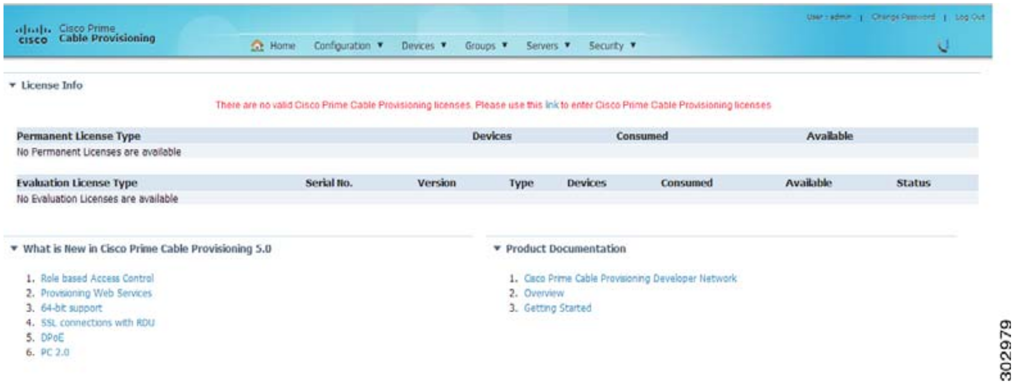
While you must install the DPE license from the Admin UI, the KDC license continues to be proprietary as in previous Prime Cable Provisioning releases, and is licensed during Prime Cable Provisioning installation.

Prime Cable Provisioning enables you to install permanent and evaluation licenses at the same time. In addition, it also allows you to install more than one evaluation license. This enables you to increase your device limit when you are in short of licenses till you purchase a permanent license.

While you can install more than one evaluation license, an expired license can be deleted and a new evaluation license (with a later expiry date), or a permanent license can be added. While deleting the expired license, ensure that the device limit of the new license at least equals the number of devices that is currently stored in the database.

Figure 2-1 shows a sample Manage License Keys page, which displays a list of service licenses that has been entered for your implementation.

Figure 2-1 Manage License Keys Page



▾ License Info

There are no valid Cisco Prime Cable Provisioning licenses. Please use this link to enter Cisco Prime Cable Provisioning licenses

Permanent License Type	Devices	Consumed	Available
No Permanent Licenses are available			

Evaluation License Type	Serial No.	Version	Type	Devices	Consumed	Available	Status
No Evaluation Licenses are available							

▾ What is New in Cisco Prime Cable Provisioning 5.0

1. Role based Access Control
2. Provisioning Web Services
3. 64-bit support
4. SSL connections with RDU
5. DPoE
6. PC 2.0

▾ Product Documentation

1. Cisco Prime Cable Provisioning Developer Network
2. Overview
3. Getting Started

Adding a License

Obtain your new license file as described in the *Cisco Prime Cable Provisioning 5.0 Quick Start Guide*. After you receive your license file, save each file to the system on which you plan to launch the Prime Cable Provisioning Admin UI.

To add a permanent or evaluation license:

Step 1 From the Admin UI, choose **Configuration > License Keys**.

> **Note** The License link appears on the Home page whenever there are no licenses added to the RDU.

Step 2 In the License File field, enter the complete path to the location of the permanent or evaluation license file on your local system.

Step 3 Click **Add**.

The Manage License Keys page appears with the details of the services or the DPEs that are licensed to be used.

Deleting a License

You can choose to delete any license—evaluation or permanent—that appears on the Manage License Keys page.

> **Note** You cannot delete a license even if doing so brings the licensed capacity of the system below the number of devices provisioned in the system.

To delete a license, from the Admin UI, choose **Configuration > License Keys** and click the **Delete** button corresponding to the permanent or evaluation license that you want to delete.

If you delete a license that contains multiple keys, the list of permanent licenses appears. Click the **Delete** button corresponding to the license that you want to delete.

The license key disappears from the Manage License Keys page.

> **Note** To confirm if the license has been deleted, verify if the action has been recorded in *audit.log*. The *audit.log* file is available at /var/CSCObac/rdu/logs/audit.log.

Understanding Prime Cable Provisioning Architecture

This chapter describes the system architecture implemented in Prime Cable Provisioning.

Deployment

Figure 3-1 represents a typical, fully redundant deployment in a Prime Cable Provisioning network.

Figure 3-1 Deployment Using Prime Cable Provisioning

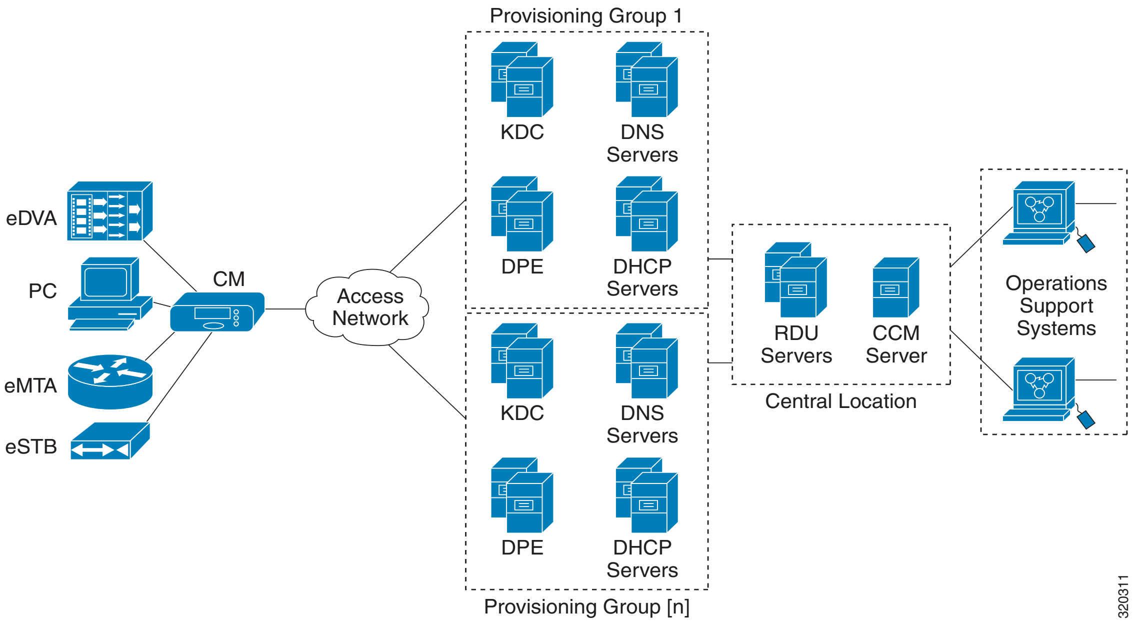


Figure 3-2 shows a typical provisioning group in a Prime Cable Provisioning network.

Figure 3-2 Typical Provisioning Group

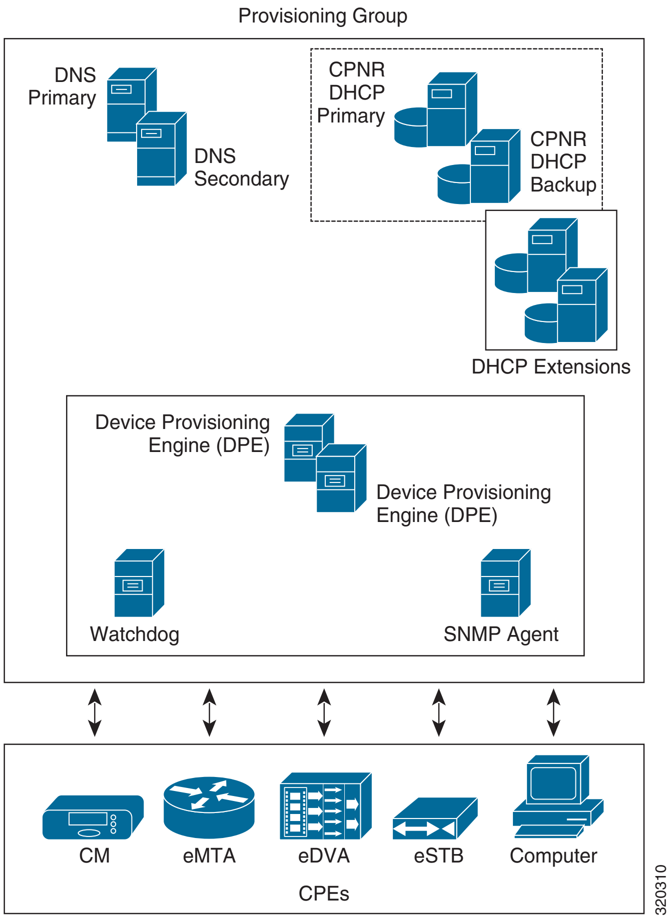
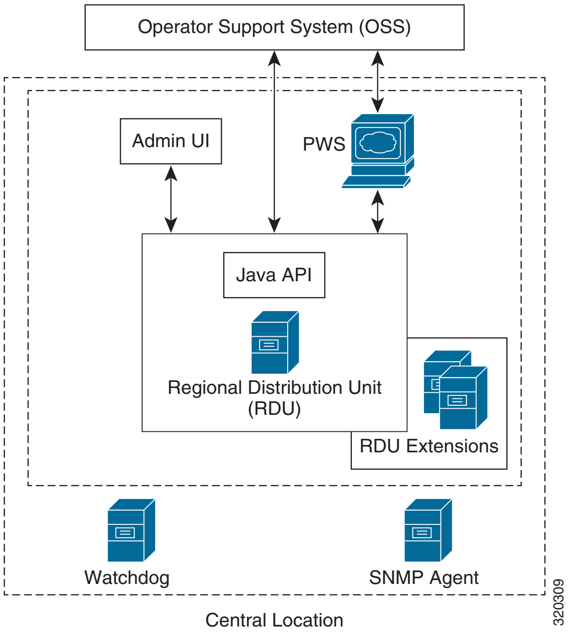


Figure 3-3 represents a central location in a Prime Cable Provisioning network.

Figure 3-3 OSS in a Prime Cable Provisioning Network



Architecture

This section describes the basic Prime Cable Provisioning architecture, such as:

Regional Distribution Unit (RDU) that provides:

- The authoritative data store of the Prime Cable Provisioning system.
- Support for processing application programming interface (API) requests.
- Monitoring of the system's overall status and health.
- RBAC for better user management.

See Regional Distribution Unit, page 3-5, for additional information.

Provisioning Web Services (PWS) that provides:

- A SOAP based web service for device provisioning functions.
- Support for both HTTP and HTTPS connectivity.
- Supports interacting with multiple RDU servers.

See Provisioning Web Service, page 3-24, for additional information.

Device Provisioning Engines (DPEs) that provide:

- Interface with customer premises equipment (CPE).
- Configuration cache.
- Autonomous operation from the RDU and other DPEs.
- PacketCable provisioning services.
- IOS-like command-line interface (CLI) for configuration.

See Configuring Device Provisioning Engines, page 6-7 and *Cisco Prime Cable Provisioning 5.0 DPE CLI Guide* for additional information.

Cisco Prime Cable Provisioning API that provides total client control over system capabilities.

See *Cisco Prime Cable Provisioning 5.0 Integration Developers Guide* for additional information about the APIs.

Cisco Prime Network Registrar servers that provide:

- Dynamic Host Configuration Protocol (DHCP).
- Domain Name System (DNS).

See Cisco Prime Network Registrar, page 3-33, for additional information.

Provisioning Groups that provide:

- Logical grouping of Network Registrar servers and DPEs in a redundant cluster.
- Redundancy and scalability.

See Provisioning Groups, page 3-37, for additional information.

A Kerberos server (KDC) that authenticates PacketCable Multimedia Terminal Adapters (MTAs). See Key Distribution Center, page 3-35, for additional information.

The Cisco Prime Cable Provisioning process watchdog that provides:

- Administrative monitoring of all critical Prime Cable Provisioning processes.
- Automated process-restart capability.
- Ability to start and stop Prime Cable Provisioning component processes.

See Chapter 23, “Prime Cable Provisioning Process Watchdog”, for additional information.

An SNMP agent that provides:

- Third-party management systems.
- SNMP version v2.
- SNMP Notification.

See SNMP Agent, page 22-1, for additional information.

An Admin UI that supports:

- Adding, deleting, modifying, and searching for devices.
- Configuring of global defaults and defining of custom properties.
- Configuring groups.
- Configuring servers and Provisioning Groups.
- Configuring RBAC.

See Administrator User Interface, page 3-36, for additional information.

Regional Distribution Unit

The RDU is the primary server in the Prime Cable Provisioning provisioning system. You must install the RDU on a 64-bit server running either Solaris or Linux operating systems.

The functions of the RDU include:

- Managing device configuration generation
- Generating configurations for devices and distributing them to DPEs for caching
- Synchronizing with DPEs to keep device configurations up to date
- Processing API requests for all Prime Cable Provisioning functions
- Managing the Prime Cable Provisioning system

The RDU supports the addition of new technologies and services through an extensible architecture.

Prime Cable Provisioning supports one RDU per installation. You could configure high availability for the RDU. To provide failover support, we recommend using the clustering software from Symantec or Oracle. We also recommend using RAID (Redundant Array of Independent Disks) shared storage in such a setup. RDU also supports Global Server Load Balancing (GSLB) to enable failover support and continue the RDU service in case the primary RDU service fails.

The following sections describe these RDU concepts:

- High Availability for RDU, page 3-6
- RBAC Management, page 3-14
- Clearing User Sessions, page 3-19
- Service-Level Selection, page 3-20
- Authentication Support, page 3-21
- GSLB Support, page 3-24

High Availability for RDU

RDU as the backbone of Prime Cable Provisioning must be fault tolerant and reliable. An RDU crash can cause severe data losses resulting in discontinuity of the cable provisioning service. RDU can crash for any of the following reasons:

- Electrical outage
- Network outage due to network malfunctioning
- Overheating of server
- Operating System crash
- Hard-disk failure
- RDU process becomes unresponsive
- Database corruption
- Database corruption due to incomplete transaction
- Malfunctioning of infrastructure software
- Race condition between primary and secondary RDU

To avoid this, Prime Cable Provisioning provides RDU redundancy on Linux operating system RHEL 6.3 (64-bit). Redundancy, one of the key aspects of high availability (HA) compliance, is the duplication of critical components or functions of a system with the intention of increasing reliability of the system. HA is assured by making a redundant pair, which will fail over to a peer in case of any outage or service breakdown. The redundant pair is referred as primary and secondary RDU nodes in Prime Cable Provisioning.

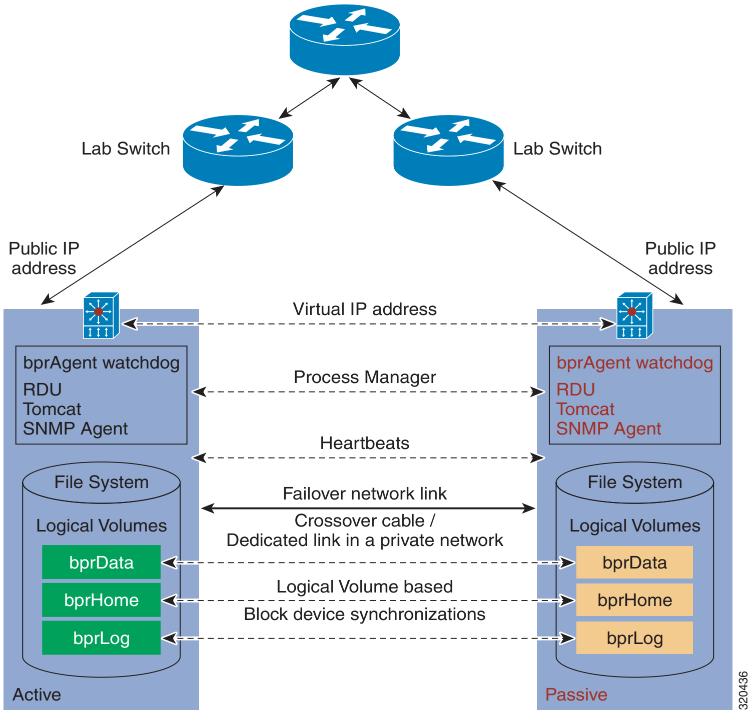
RDU HA clustering is an active-passive setup (1:1 node setup), which means that only one active RDU will function at any given time. RDU HA ensures the following:

- Virtual IP (VIP) based switching between primary and secondary RDU node.
- Failover between primary and secondary node.
- Database replication between primary and secondary by means of block level synchronization.
- Prime Cable Provisioning configuration files replication between primary and secondary RDU nodes. These are RDU configuration files.
- Provisioning of manual and automatic failback.

For more information about configuring, monitoring and troubleshooting RDU redundancy, see Scripts to Manage and Troubleshoot RDU Redundancy, page 28-20.

For installation, configuration and deployment details, see the *Cisco Prime Cable Provisioning 5.0 Quick Start Guide*.

Figure 3-4 RDU Redundancy



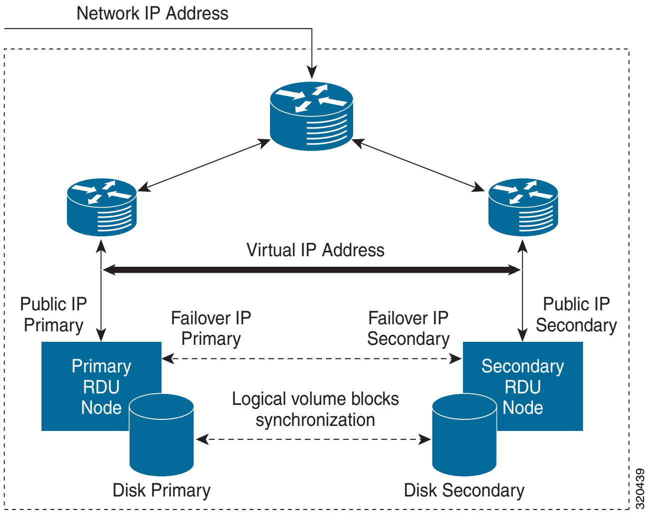
The solid line indicates physical network connectivity between the nodes. The dotted lines indicate logical synchronization between the two nodes.

1:1 Active-Passive Setup

A high performance system dedicates one secondary for each primary, a 1:1 failover relationship, where the secondary is an exact replica of the primary, including configuration information. To ensure RDU redundancy, the following active-passive setup is established:

- By default, Prime Cable Provisioning resources including the VIP resource are active only on the primary node. In case of a failover, they are migrated to the available active node (secondary node).
- Failure of the HA configured resources on active can lead in to a failover to peer (passive) node.
- Automatic failover and manual failback support.
- Minimized race-conditions and split-brain situations.
- Configurable failure stand-by timeout for automatic failback. Automatic failback starts when the CRM resources are cleaned up on the failed node.

Figure 3-5 RDU VIP Setup



Note No Intelligent Platform Management Interface(IPMI) controller is configured through RDU redundancy setup. (STONITH is not configured)

VIP and Interface Redundancy

The purpose of the VIP and interface redundancy feature is to provide IP addresses that can float between the nodes and provide redundancy between the two nodes. These interfaces perform the following:

- Provide network redundancy for interface addresses between switches. The interface address has to be in the range of a subnet common to each switch.
- Provide redundancy for VIP addresses between switches. The VIP address has to be in the range of a subnet common to each of the switches. VIP redundancy can be active or passive, where only one switch services requests for the VIP, or shared, where multiple switches service VIP requests.
- Both the IP addresses and MAC addresses of a redundant interface or VIP are shared. In other words, the MAC address does not change when the backup takes over.
- RDU redundancy setup can be reached via VIP and the VIP can be mapped to a domain name.
- No changes are required at the client layer to communicate to the RDU redundancy setup. During any failover and failback operation, clients may experience a short outage of RDU (maximum of 1 minute). This is the startup time required for RDU to come online on any node.

bprAgent and RDU Process

RDU process is managed by bprAgent, the watchdog process that controls the state of the various Prime Cable Provisioning processes. Admin UI and SNMPAgent are two integral and important processes running along with the RDU. RDU HA provides redundancy for these processes. Failover events for RDU also migrates these processes along with it to the active node.

RDU HA compliance makes the following three set of resources redundant along with the bprAgent watchdog process:

1. bprAgent
    a. RDU
    b. Admin UI (tomcat server)
    c. SNMPAgent
2. VIP
3. File systems. These are RDU redundancy specific file systems mounted on the synchronized logical volume.
    a. /bprData
    b. /bprHome
    c. /bprLog

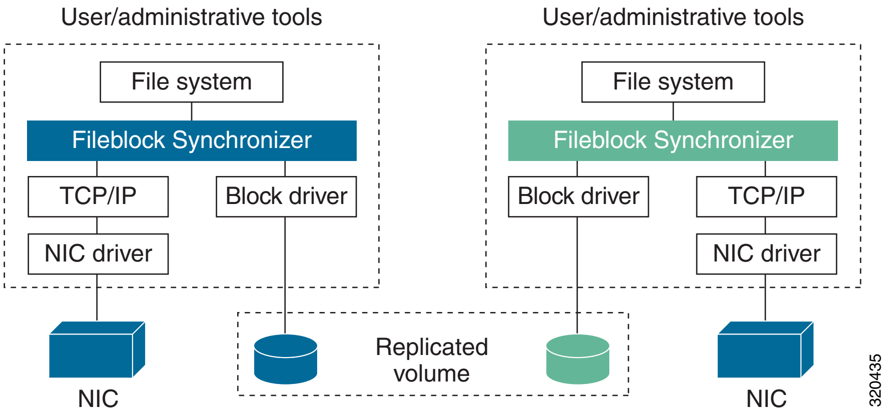
File System Replication

For Prime Cable Provisioning HA compliance, file system replication works on top of file blocks, which are LVM's (Logical Volume Manager) logical volumes (/bprData, /bprHome and /bprLog mounted over respective logical volumes). It mirrors each data block that is written to disk to the peer node.

In Prime Cable Provisioning, asynchronous mirroring is implemented. This means that the entity that issued the write requests is informed about completion as soon as the data is written to the local disk. Asynchronous mirroring is necessary to build mirrors over long distances, i.e., the interconnecting network's round trip time is higher than the write latency you can tolerate for your application.

Note The network latency between primary and secondary node must not be more than 100 milliseconds.

Figure 3-6 Database Mirroring



A consequence of mirroring data on block device level is that you can access your data, using a file system, only on the active node. This is not a shortcoming of Synchronizer but is caused by the nature of most file systems (ext3, XFS, JFS, ext4 ...). These file systems are designed for one computer accessing one disk, so they cannot cope with two computers accessing one (virtually) shared disk.

Prime Cable Provisioning uses the Logical Volume based synchronization and creates three logical volumes, which can be synced over the secondary. You could choose to have even one or two logical volumes.

- Logical volumes on both the nodes should be of same name and capacity. Following are the recommended configurations:
  - lv_bprHome(mounted on /bprHome , capacity 5 GB)
  - lv_bprData(mounted on /bprData , capacity 75 GB)
  - lv_bprLog(mounted on /bprLog , capacity 5 GB)
- Logical volumes must be pre-created with ext4 file system on them. The block size is set to default.

File System Synchronizer

During installation or migration, there is a huge change in data and for synchronization between nodes to begin, you must wait until disks on both sides are not in UpToDate state. The possible disk states are listed in Disk States, page 3-12.

If in case you find the synchronization is stopped, use the following command on both the nodes a few times to start the synchronization

```
#fs_ha_adjust.sh all
```

Heartbeat Configurations

The heartbeat manager manages the heartbeat between the active and passive nodes. There are two physical interfaces connected on both nodes. Heartbeats are configured on both the network links on both nodes.

- Public network interface link (used to access the node by the external clients. For example, API client and PWS client)
- Failover network interface link (failover network interface link/cross over cable used for synchronization data between the nodes)

HA Cluster Management

CRM or cluster resource manager manages HA clustering when:

- RDU process is inactive. If RDU is not able to start due to some reason, after a pre-configured timeout RDU process should failover to secondary node.
- RDU becomes unresponsive. RDU process can become temporarily unresponsive for reasons like; the server is overloaded or the underlying database is corrupted. In situations like this the cluster resource manager is configured to do the following:
  - If RDU process is able to respond before the timeout, do not failover.
  - If RDU process remains unresponsive for longer than timeout, declare the existing primary RDU process unresponsive, and failover to secondary after the restart counts exceeds the threshold value (default 3 minute). Once the primary become responsive again failback to primary after a manual clean up of resources on primary node.

Changing Configuration for RDU Cluster Maintenance

Post installation, if you want to change or add any configuration properties to RDU, you can do so by:

Step 1 Use the VIP to SSH into the RDU server.

Step 2 Provide a valid root username and password to log in. The user must have the administrative privilege.

Step 3 Stop the RDU resource from CRM, using the command:

```
manage_ha_resource.sh stop res_bprAgent1
```

Step 4 Make the property configuration changes and then start the RDU using the command:

```
manage_ha_resource.sh start res_bprAgent1
```

Step 5 Verify that RDU is started:

```
/etc/init.d/bprAgent status
```

Warning **Do not stop bprAgent using the /etc/init.d/bprAgent stop command. Doing so misguides the cluster manager state machine.**

Following are some important configurations, whose details can be viewed using the command, `monitor_ha_cluster.sh`.

- Resources type
- VIP (res_IPaddr2_1)
- Master/slave resource for file system (res_drbd_1, res_drbd_2 and res_drbd_3)
- FileSystem resources for mounting the drbd on filesystem (res_Filesystem_1, res_Filesystem_2 and res_Filesystem_3)
- bprAgent resource (res_bprAgent_1)
- Health of each resource
- Failure timeout values
- Failure threshold
- Co-location of the resources (dependencies)
- Location of all the resources

Cluster Timeout Configuration

Cluster components have a exponential back-off timeout for each of the resources. Here the file system resource is the most fundamental resource, which has to be functional before the bprAgent. Once the bprAgent resource is up and ready to serve only then you can configure the VIP and make it available for the rest of the service infrastructure. Timeout of the individual resources are as follows:

- All resources have a default failure threshold configured to three. After three attempts, resource with all dependencies will failover to peer node. The failed node is troubleshooted and cleaned up using utility scripts; the resource still continues to be alive on the secondary node until the failover timeout expires (default 30 minutes). On the expiry of failover timeout all resources failback to the primary node (preferred location). This happens only if auto-failback is enabled.
- Timeout values for VIP address resource
  - start interval="0" timeout="30 sec"
  - stop interval="0" timeout="20 sec"
  - monitor interval="10" timeout="20 Sec" start-delay="0"
- Timeout for bprAgent resource
  - start interval="0" timeout="180 sec"
  - stop interval="0" timeout="180 sec"
  - monitor interval="30 sec" timeout="120 sec " start-delay="15 sec"
- Timeout for master-slave and file system resource for all three file system resources (/bprHome, /bprData and /bprLog)
  - start interval="0" timeout="30 min"
  - promote interval="0" timeout="30 min"
  - demote interval="0" timeout="30 min"
  - stop interval="0" timeout="30 min"
  - monitor interval="10" timeout="20 Sec" start-delay="0"
  - notify interval="0" timeout="90 Sec"

Disk States

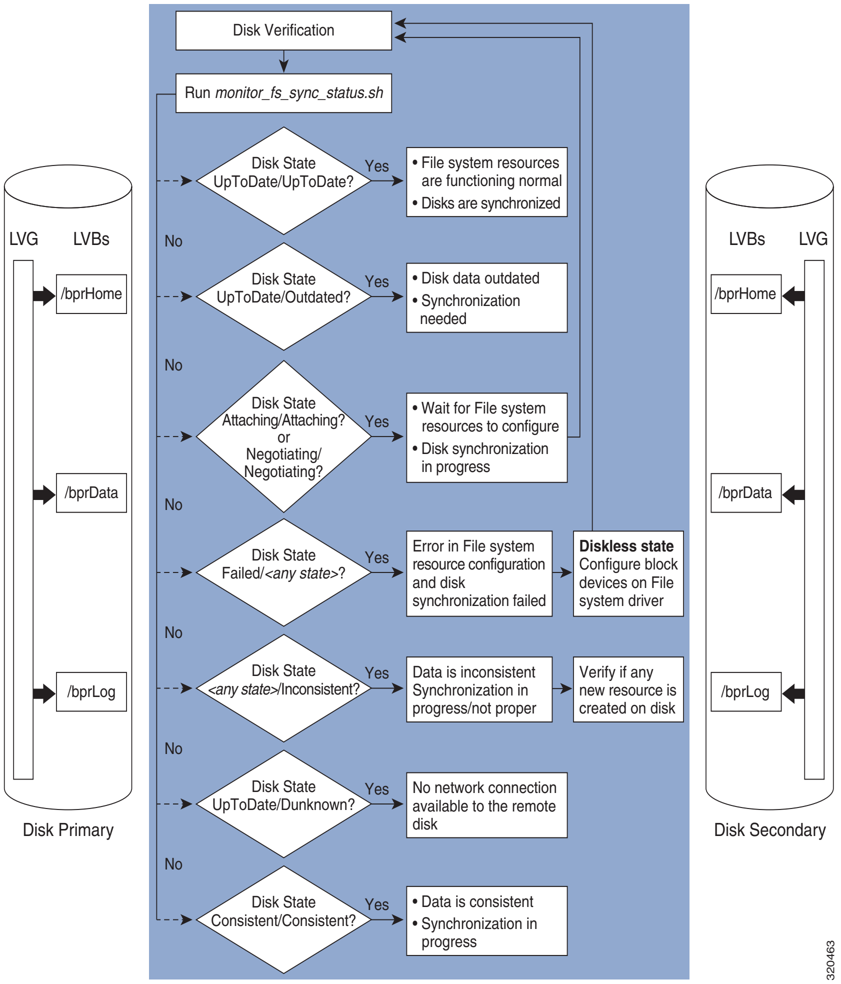
Disk synchronization means synchronizing logical volumes between primary and secondary RDU disks (at file block level). Using Logical volume manager, you can create the logical volume group with three logical volume blocks on each disk, and these block devices would be mounted on home, data, and database log directories. These logical volume blocks are further configured on the File system driver to enable synchronization between RDU nodes.

You can verify the synchronization status using the script monitor_fs_sync_status.sh. This script can be run on either primary or secondary RDU server. The disk status is displayed in the format <local disk status>/<remote disk status> which means the local disk state is displayed first followed by the remote disk state. The disk state helps you to determine the synchronization status of primary and secondary RDU disks.

The local disk state is always displayed first and then the remote disk state. Both the local and remote disk states may be of the following type:

- UpToDate/UpToDate—Indicates that the disks are synchronized and file system resources are functioning normal.
- UpToDate/outdated—Indicates that the local server disk is updated but the remote disk is not synchronized properly. In this case, you must verify whether the File system driver is functioning normal and reinitiate the synchronization process.
- Attaching/Attaching—Indicates that the synchronization is in progress. the primary server is trying to achieve the network connectivity with the secondary server using the failover IP. You must wait until the synchronization completes before running any tasks on the RDU.
- Negotiating/Negotiating—Indicates that the network connectivity between primary RDU server is established, and the data synchronization is in progress. You must wait until the synchronization completes before running any tasks on the RDU.
- Failed /<any state>—Indicates that the synchronization is failed. Verify whether the logical volume blocks are configured on the File system resource driver. If not, set up the file system driver and reinitiate the synchronization process.
- <any state>/Inconsistent—Indicates that either a new resource is added on the local disk or the synchronization is in progress. Verify if any new resource is added before completion of the initial full synchronization. If yes, reinitiate the synchronization process.
- UpToDate/Dunknown—Indicates that the network connectivity is not available between RDU nodes. Verify whether the failover IPs are correct and configured appropriately.
- Consistent/Consistent—Indicates that the data is consistent in both the disks based on the initial synchronization. To get the current disk status, initiate the synchronization process to verify whether the data is up to date in both the disks. You may also receive this disk state if the synchronization is in progress.

Figure 3-7 Disk Space


Disk Verification
Run *monitor_fs_sync_status.sh*
Disk State UpToDate/UpToDate?
Yes
• File system resources are functioning normal
• Disks are synchronized
No
Disk State UpToDate/Outdated?
Yes
• Disk data outdated
• Synchronization needed
No
Disk State Attaching/Attaching? or Negotiating/ Negotiating?
Yes
• Wait for File system resources to configure
• Disk synchronization in progress
No
Disk State Failed/<any state>?
Yes
Error in File system resource configuration and disk synchronization failed
Diskless state Configure block devices on File system driver
No
Disk State <any state>/Inconsistent?
Yes
Data is inconsistent Synchronization in progress/not proper
Verify if any new resource is created on disk
No
Disk State UpToDate/Dunknown?
Yes
No network connection available to the remote disk
No
Disk State Consistent/Consistent?
Yes
• Data is consistent
• Synchronization in progress
LVG
LVBs
/bprHome
/bprData
/bprLog
Disk Primary
Disk Secondary
320463

RBAC Management

For better user management and security, Prime Cable Provisioning introduces Role Based Access Control (RBAC) that provides an approach to restrict access to system functions and resources to authorized users. Roles are composed of fine grain privileges. A privilege is a base unit of enforcement. A role groups a set of privileges into a logical job function that enables the customization of authorization policies.

Prime Cable Provisioning provides default out of the box (OOTB) roles, privileges, users, user groups, and domains that you can leverage from. Apart from these default configurations, you can also define your own setup to meet your organization requirements. The default OOTB configurations cannot be edited or deleted.

Authorization enforcement requires knowing the identity of the users and their granted privileges for any operation or resource to be protected. This information is used while performing access enforcement checks. There are four levels of checks.

- URL access check - Enforcement done by web facing components such as the Admin UI or web services.
- Operation/Method level check - Enforcement done by the components protecting access to operations. This type of access check is primarily performed in the RDU and DPE CLI. It is meant to ensure that the user has the correct privileges to invoke operations.
- Instance level check - Enforcement to ensure that the user has access to a specific object. This enforcement is performed in the RDU and leverage database capabilities.
- Property level check - Enforcement to ensure that the user has write access to a specific property. This enforcement is performed in the RDU.

For more details RBAC configuration, see Configuring RBAC Using Admin UI, page 13-1.

Following topics are explained in this section:

- Authentication, page 3-15
- Authorization, page 3-15
- Role Evaluation, page 3-16
- Operation Level Access Control, page 3-16
- Instance Level Access Control, page 3-17
- Sample RBAC User Role Domain Hierarchy, page 3-18
- Property Filtering and Property Enforcement, page 3-19

Authentication

Authentication is the process of establishing the identity of a user. This process is achieved through the use of username/password credentials against the local RDU database or an external Radius server. Credentials are first checked in the Radius server and if not found, it is sent to the local server for validation. Radius authentication is possible only if it is enabled and configured in RDU Defaults. After validating the user's credentials, either an exception is shown in case of an authentication failure or in case of a valid user, the user is granted the privileges and domains based on the Roles, User-Groups, and Domains associated with the user.

Authorization

Prime Cable Provisioning users are authorized based on the various roles that are assigned to them directly or indirectly via user-groups. These roles consist of finer grain privileges. Following are the major authorization entities of Prime Cable Provisioning.

User

A user represents an identity that can either be a person or a system actor that is granted access to Prime Cable Provisioning. Depending on the role to be performed, a user can either have zero or more roles.

User Group

A user group is a collection of users. Like a user, a user group can also be assigned to zero or more roles. A user who is a member of a user group will inherit all the roles that the user group is assigned to. Those roles are constrained to only be valid on the resources that are also members of the group. A user can be a member of zero or more groups. The set of privileges the user gains is the aggregate of all those from the role.

Privilege

A privilege represents an authority granted for an operation that can be performed. It is the unit of enforcement. Privileges are grouped in roles, which are assigned to users. Privileges can have create, read, update, or delete actions.

Role

A role is a job function that defines a set of capabilities a user or user group can perform. A role binds privileges, users/user groups, and domains together. Prime Cable Provisioning comes with a set of default out-of-the-box roles and it also has the ability to create custom roles. Any set of privileges can be assigned to a custom role.

Domain

A domain represents a collection of objects (e.g. Device, COS, DHCP Criteria, Files, ProvGroup, etc) grouped for the purpose of instance level access control. The following are characteristics of domains:

- They are a set of instances. Domains can partition the overall system. A domain can have various object types. Authenticated users with the appropriate access privileges should be able to view the instances that exist in their domains.
- Domains are hierarchical. A user who has access to a parent domain can access all of the child domains, grandchild domains, and so on, of that parent.
- Domains have unique names across the system.
- A system defined (built-in) RootDomain can be used to give access to all objects for a particular role.
- Resources can be assigned to any domain. Resources can also be moved between domains. In the absence of domain assignment, the resources are assigned to the RootDomain.

Role Evaluation

A user can be granted privileges by directly assigning roles to the user or indirectly through user group membership. The total set of granted capabilities a user has is the union of all privileges derived from all roles directly or indirectly assigned to a user. The order of role evaluation is: user group, then user.

Operation Level Access Control

Every task that you carry out in Prime Cable Provisioning goes through an access control check. During the access control check, the task is validated against the privileges that you are assigned to and only if you have the right privilege, the task gets executed. In case you do not have the privilege, an insufficient privilege message is shown.

Instance Level Access Control

While operational level access control defines what actions a user can perform, instance level access control determines whether or not those actions can be performed on a specific instance. This is additional access control enforcement beyond checking of operation access. Operational access control must first be granted before instance level can be attempted. Prime Cable Provisioning supports a mode where operation level enforcement is enabled, but instance level access control is disabled.

In Prime Cable Provisioning, domains define the subset of instances that a user can access. Both the user and the resource must be members of the same domain. Otherwise, the user is not allowed to perform the operation. User's can be associated with zero or more domains. A resource (e.g. Device, COS, File, etc) can only be associated with a single domain. When a resource is added to Prime Cable Provisioning, it is assigned to a domain. If no domain is assigned, the resource is automatically be added to the RootDomain.

Instance level access is controlled through RDU APIs associated with each resource.

Instance level access control can be enabled or disabled, through the RDU Defaults in the Admin UI using the INSTANCE_LEVEL_AUTH_ENABLE property and this field is enabled only when the /adminui/enableDomainAdministration is set to true in the adminui.properties file.

Prime Cable Provisioning only supports instance level access control for the following resources: Device, COS, File, DPE, NR, Prov Group, and DHCP Criteria.

If instance level access control is disabled:

- When a new resource is added, the resource is internally and automatically assigned to the RootDomain by the respective API without requiring the user to assign the resource to a specific domain.

If instance level checking is enabled:

- When a new resource is being added, it is mandatory to provide a domain name to which the resource must be associated.
- The user must have access (i.e. membership) to the domain that a resource is being assigned to. This is enforced by APIs that assign or change the domain membership. The same API also enforces that a valid domain is being assigned to.
- In the Admin UI, if the user has access to a single domain, then that domain is selected by default while adding or modifying the resources.
- In the Admin UI, if the user has access to multiple domains, no default selection is made. The user needs to explicitly set the domain while adding a new resource.
- The user must have the PRIV_DOMAIN_READ privilege to read the configured domains. This privilege is added to all the out-of-the-box roles.
- The users will only be able to see the domains (includes sub-domains if any) that they are members of.

By default, newly added provisioning groups(PGs) are placed in the RootDomain. You can change a PG's default membership via the Admin UI or through the ChangeDomainProperties API. You can change a DPE's or CNR-EP's domain through the Admin UI or through APIs.

For a newly added DPE or CNR EP, it will take the domain membership of its PG. First primary PG in case of DPE.

While changing a PG's domain through the Admin UI, an option to apply the new domain to all the servers in the PG is provided. This can only be done through the Admin UI; there is no API support for this operation.

Note Configuration generation and regeneration does not support instance level access enforcement. The checks are only enforced at the administration operation and object level. For example, if a user is granted access to change a ClassOfService, the devices whose configuration may be regenerated are permitted as a result of that change. These devices do not undergo instance level access check enforcement during generation and regeneration.

Sample RBAC User Role Domain Hierarchy

Device Admin role contains the following privileges: COS read, DHCP criteria read, device create, read, update, delete.

File Admin role contains only File create, read, update, and delete privileges. Privileges are updated for deviceadmin and fileadmin roles.

Super Admin role can perform create, read, update, and delete on COS, DHCP criteria, device, and provisioning groups operations.

For example:

User W is assigned the Device Admin responsibilities for Domain 1. This allows user W to have:

- Read-only access for all COS in Domain 1.
- Read-only access for all DHCP Criteria in Domain 1.
- Create-Read-Update-Delete access on all Devices in Domain 1.
- Read access that permits the viewing of all details.
- Read access that permits the searching by MAC, DUID, FQDN, Owner ID, COS, DHCP Criteria etc.
- Update access that permits changing MAC, DUID, HOST NAME, Owner ID.
- The privilege to read a COS or DHCP Criteria plus being able to create-update a device. It also allows user W to assign or unassign COS and DHCP Criteria on those devices.
- If user W has read-update on device but no read access on COS, they will not be able to assign a COS, but they can still see the name of the assigned COS.
- Complete access to add, update, or delete any property on those devices in Domain 1.
- Create or update access implicitly enables user W to perform generate and regenerate operations. Having only read access will not be satisfactory.
- The PRIV_DEVICE_OPERATION privilege facilitates user W to invoke performOperation API requests.

If user X is assigned the Super Admin responsibilities for Domain 1, then user X can perform all operations that user W can along with the following operations:

- Create, read, update, and delete any COS, DHCP Criteria, Device in Domain 1.
- Access all provisioning groups associated with Domain 1 and see server details, access logs and perform DPE CLI operations.

If user Y is assigned Super Admin responsibilities for Domain Parent then user Y can perform all operations that user X can and also have access to all of Domain Parent's child domains.

If user Z has similar capabilities as user X, but on Domain 3 then:

- User Z can create (add), read (view or export), update (replace or change properties), and delete files in either Domain Parent or Domain 3.

- Since user Z privileges include COS create, update on Domain 3, user Z can assign and unassign files to COSs that are associated with Domain 3.

DPE CLI Access Enforcement

Using the DPE CLI, you can view the status and configuration of the DPE as well as change properties. You must be authenticated to use the DPE CLI either through the local DPE, TACACS, or via Radius.

The local DPE CLI account has a specific username, **admin** and this requires only the local DPE CLI authentication. By default, the admin user enters into the disable mode upon authentication and then can enter into the enable mode without having to enter the password again. DPE CLI access is controlled by a set of DPE privileges. For details about DPE privileges, see *Cisco Prime Cable Provisioning 5.0 DPE CLI Guide*.

In Prime Cable Provisioning, a DPE audit log file is created to list the authentication details. This file is located at /var/CSCObac/dpe/logs.

Property Filtering and Property Enforcement

To better manage the visibility of device level properties, Prime Cable Provisioning introduces a filtering mechanism in conjunction with write level access enforcement. This is an overly mechanism that is distinct from the fine grain access control capabilities discussed earlier. The Prime Cable Provisioning public APIs are extended to permit the filtering of device properties. It is the responsibility of the client to specify the filter. The Prime Cable Provisioning API call delivers only those properties that satisfy the filter. This mechanism is intended to both reduce the amount of data transmitted and to allow service based applications to enforce access control.

Prime Cable Provisioning provides write level access control for device properties. The list of properties that can be modified must be associated with a role. For a user to modify a given device property, it should be defined as part of the role to which the user is assigned to.

Note The write check for device properties is only at device level. A user can still add or change device properties at the higher property hierarchy like COS.

Clearing User Sessions

Once a RDU local user logs in, a user session is created. A user can have multiple concurrent sessions. The privileges and accessible domains of a user are cached until the last active session of the user is either terminated or timed out. When the admin changes the privileges or accessible domains of an RDU user, the new changes would not take effect until all existing sessions of that user are either terminated or timed out. This is not applicable for Radius users. Prime Cable Provisioning provides a shell script tool, closeSession.sh to clear all the sessions of a given user. This tool is available in the RDU under the location BPR_HOME/rdu/bin.

For example,

To clear the session of user John, run the command:

```
#./closeSession.sh John
```

Service-Level Selection

The extension point for service-level selection determines the DHCP Criteria and the Class of Service that the RDU is to use while generating a configuration for a device. The RDU stores this information for each device in its database.

The DHCP Criteria and the Class of Service that the RDU uses to generate a configuration for a device is based on the type of access granted to the device. Device access is of three types:

- Default—For devices granted default access, Prime Cable Provisioning uses the default Class of Service and DHCP Criteria assigned for the device type.
- Promiscuous—For devices granted promiscuous access, Prime Cable Provisioning obtains the Class of Service and DHCP Criteria from the relay agent that the device is behind.
- Registered—For devices granted registered access, Prime Cable Provisioning uses the Class of Service and the DHCP Criteria registered for the device in the RDU database.

There should always be one default extension per device type.

You can enter service-level selection extension points for specific technologies using the default pages at **Configuration > Defaults** from the Admin UI. For additional information, see Configuring Defaults, page 11-3. By default, these properties are populated with zero or with one of the built-in extensions.

Caution Do not modify these extensions unless you are installing your own custom extensions.

Although a device may have been registered as having to receive one set of DHCP Criteria and Class of Service, a second set may actually be selected. The configuration generation extension looks for the selected DHCP Criteria and Class of Service and uses them.

The service-level selection extension selects a second Class of Service and DHCP Criteria based on certain rules that you specify for a device. For example, you may specify that a device must boot in a particular provisioning group for the device to be assigned a specific Class of Service and DHCP Criteria.

The extension returns information on why a specific set of DHCP Criteria and Class of Service is selected to provision a device. You can view these reasons from the Admin UI on the View Device Details page.

Table 3-1 describes these reasons and the type of access granted in that case.

Table 3-1 Reasons for Device Access as Determined by Service-Level Selection Extension

Reason Code	Description	Type of Device Access Granted		
		Default	Promiscuous	Registered
NOT_BEHIND_ REQUIRED_DEVICE	The device is not behind its required relay agent.	✓		
NOT_IN_REQUIRED_ PROV_GROUP	The device is not in its required provisioning group.	✓		
NOT_REGISTERED	The device is not registered.	✓		
PROMISCUOUS_ ACCESS_ENABLED	Promiscuous access is enabled for the relay agent.		✓	
REGISTERED	The device is registered.			✓

Table 3-1 Reasons for Device Access as Determined by Service-Level Selection Extension

		Type of Device Access Granted		
Reason Code	**Description**	**Default**	**Promiscuous**	**Registered**
RELAY_NOT_IN_ REQUIRED_PROV_GROUP	The relay agent is not in the required provisioning group.	✓		
RELAY_NOT_REGISTERED	The relay agent is not registered.	✓		

Note Most of these reasons indicate violations of requirements for granting registered or promiscuous access, resulting in default access being granted.

Authentication Support

Authentication is the process of establishing the identity of a user to ensure that a user is who they claim to be. A user accessing the RDU can be authenticated locally by the RDU if they have an account created on the RDU. In addition, remote Radius authentication is supported. Prime Cable Provisioning supports the Authentication and Authorization features of AAA.

Local Authentication

This mode authenticates the user in the local RDU database and this mode is always enabled. For the admin user, only local authentication is used. For more details, see Adding a New User, page 13-6 section of User Management, page 13-5 or RDU Defaults, page 11-8.

Remote Authentication

This mode authenticates the user in a remote server. Prime Cable Provisioning uses Radius and TACACS for remote authentication.

Radius Authentication

The RDU supports externalizing authentication to a remote Radius server. The user need not have an account in the RDU database. However, a reliable batch submitted by a Radius-only user cannot be guaranteed to execute across reboots or when the user logs out. This applies to those users that do not have their privileges defined in the RDU account but are only provided by remote Radius.

Radius authentication support can be configured using the Admin UI (Configuration > Defaults > RDU Defaults). To leverage Radius, a primary Radius server must be configured. This following properties must be provided: Primary Host, Primary Shared Secret, Primary Port (default=1812), Timeout (default=1000ms), Retries (default =1). If not specified, authentication will fail.

A secondary Radius server can also be configured. Only in the event the primary server is not available the secondary server is consulted to authenticate the user. The primary and secondary Radius servers support the same set of users.

Radius is a UDP-based protocol that supports centralized authentication, authorization, and accounting for network access. Radius authentication involves authenticating the users accessing the network services via the Radius server, using the Radius standard protocol defined in RFC 2865.

Radius authentication are of two modes and they are as follows:

- **Without Two-Factor:**

In this mode, username and password are required to log on to RDU which must be configured in Radius server.

Note For the users authenticated via Radius, the password can be changed only in the Radius server.

- **Two-Factor:**

In this mode, username and passcode are required to log on to RDU. The username and assigning RSA SecureID Token to user must be configured in RSA Authentication Manager. The RSA SecureID generates the TokenCode which will be updated every 60 seconds in the RSA SecureID Token. The combination of TokenCode and the pin associated with the RSA SecureID Token will be used as the passcode of the user.

For example, if the PIN associated with the RSA SecureID token is 'user' and the Token Code generated from the RSA SecureID token is '12345', then the passcode is 'user12345'.

Note Changing the combination order of the passcode from Token Code and PIN will result in authentication failure. The PIN for the RSA SecureID tokens must be assigned in RSA Authentication Manager through RSA Authentication Agents.

Creating or modifying a PIN for RSA SecureID token can be done via RSA Authentication Manager.

To enable Radius authentication, the authentication mode must be configured in the RDU Defaults page. For more details, see RDU Defaults, page 11-8.

Radius Integration

Prior to Prime Cable Provisioning 5.0, the property /rdu/auth/mode could take the value, LOCAL or Radius indicating which mode of authentication is enabled. With Prime Cable Provisioning, this is changed and the local mode is always enabled. /rdu/auth/mode is only used to enable or disable Radius authentication. If enabled, the external Radius servers are always used for authentication. If the servers fail to respond or reject the user, local authentication is attempted.

For Radius authenticated user, Prime Cable Provisioning supports granting privileges, sessions allowed, and domains to users through the Cisco-AVPair Radius VSA. The Radius Access-Accept can contain zero or more Cisco-AVPair VSA. The supported Cisco-AVPair format is:

Usergroup

Format: cp:groups=<group1>,…<groupN>

- Takes zero or more user group names.
- If no user group is assigned, the user is not granted any privileges.
- First it checks for mapping. If external to internal user group name mapping is found, privileges are given based on internal user group name. If there is no mapping, then the RDU database is searched for a user group with such a name. If that is also not there, then no privileges are assigned.
- The aggregate of all privileges is returned. This depends on the roles the user group possess.
- If no RDU database user group is found, no privileges are assigned.

Example 3-1 cp:groups=Administrators,Regional

This specifies that the authenticated user is a member of two user groups: Administrators and Regional. The user is granted the roles assigned to these two groups. Also for a user to be granted privileges, the user group must be granted one or more roles.

Prime Cable Provisioning also supports mapping of external user group name to an internal user group name. See User Group Mapping, page 13-5 for more details.

Sessions

Format: cp:sessions-allowed=<integer>

- The integer value must be zero or greater.
- If the value cannot be parsed, the user is given the RDU session default and an error is logged.
- If the Access-Accept does not specify the sessions-allowed, the user is given the RDU session default.

Example 3-2 cp:sessions-allowed=3

If the Radius Access-Accept Cisco-AVPair VSA contains cp:sessions-allowed=3, the user is allowed three concurrent sessions.

Domain

Format: cp:domains=<domain1>,…<domainN>

- Domain names must explicitly match those in the RDU.
- Only those domains that match are assigned to the user.
- If no domains are specified, the user is not granted any domain memberships.

Example 3-3 cp:domains=north,south

If the Radius Access-Accept Cisco-AVPair VSA contains cp:domains=north,south, the user is allowed membership into the north and south domains.

Backward Compatibility for Existing Users

Unlike in the earlier releases, there is no need to create duplicate users in both RDU and Radius in Prime Cable Provisioning 5.0. In this release, RDU user configuration overrides Radius user configuration for authorization. This is done to support backward compatibility of existing Radius users. After migrating from an earlier version to Prime Cable Provisioning 5.0, all existing Radius users are created as local users in RDU. So it is advised that you delete all the existing duplicate Radius users once the Radius users are configured with the appropriate Cisco AV Pairs. If there is a duplicate user (same name) present in both RDU and Radius, even though, the user would get authenticated by the Radius server, for authorization RDU configuration takes precedence.

For example

If the user John is present in both Radius and RDU and in Radius, John is configured with COSAdmin privileges and in RDU, John is configured with DeviceAdmin privileges, on login using the Radius password, John is authenticated by Radius but the privileges set in Radius are ignored as the user configuration for John present in RDU takes precedence for authorization.

GSLB Support

Global Server Load Balancing (GSLB) directs DNS requests to the best-performing GSLB website in a distributed internet environment. In Prime Cable Provisioning, GSLB is used to implement failover, which enables the continuation of the RDU service after the failure of the primary RDU. When the primary RDU fails, all the client requests will be routed to the secondary RDU. In Prime Cable Provisioning, if the IP address of FQDN is changed and the primary RDU is down, FQDN is resolved to the new IP address and all the clients are routed to the secondary RDU.

Provisioning Web Service

The Provisioning Web Service(PWS) component of Prime Cable Provisioning provides a SOAP based web interface that supports provisioning operation. The provisioning services include functionalities such as: adding, retrieving, updating, and removing objects necessary to support the provisioning and configuration generation of CPEs. Here objects include devices, classes of service, DHCP criteria, groups, and files.

The web service is hosted on a tomcat container and it is recommended that you install it on a separate server and not on the RDU server.

Note If you are installing both RDU and PWS on the same server, the installation configurations chosen for PWS take precedence over the Admin UI configurations. For example, if you have chosen secured mode of communication for Admin UI and non-secured mode for PWS, non-secured mode is chosen for both Admin UI and PWS.

For complete information about PWS, its APIs, capabilities and user cases see the *Cisco Prime Cable Provisioning 5.0 Integration Developers Guide*.

For details about PWS configuration see, Chapter 6, “Configuring Provisioning Web Services”

The web service manages these activities:

- Exposes a provisioning web service that provides functionality similar to the current API client.
- Supports stateless interactions.
- Supports both synchronous and asynchronous requests.
- Supports singular and plural operations.
- Supports both stop on failure and ignore on failure.
- Service interface supports request that can operate on multiple objects.
- Supports SOAP v1.1 and 1.2.
- Supports WSDL v1.1.
- Support WS-I Basic Profile v1.1.
- Support both HTTP and HTTPS transport.

Note PWS can communicate only with Prime Cable Provisioning 5.0 RDU. It is not compatible with RDUs of earlier releases of Prime Cable Provisioning.

The following sections describe these PWS concepts:

- Provisioning Web Services APIs, page 3-25
- Asynchronous Service, page 3-27
- Session Management, page 3-27
- Transactionality, page 3-28
- Error Handling, page 3-28

Provisioning Web Services APIs

Table 3-2 lists the PWS APIs along with their descriptions. For more details about the APIs, see the *Cisco Prime Cable Provisioning 5.0 Integration Developers Guide*.

Table 3-2 PWS APIs

API name	Description
createSession	Authenticates a client and establishes a session between PWS client and the PWS.
closeSession	Releases the session between the user and the web service including closing the connection with Prime Cable Provisioning components.
addDevice	Submits a request for the addition of a new device.
addDevices	Submits a request to add multiple new devices. Using the execution options, each device can be wrapped in a separate transaction (batch) or a single transaction.
getDevice	Retrieves data associated with a specific device.
getDevices	Retrieves data associated with the specific devices.
getDevicesBehindDevice	Retrieves the list of devices downstream of a specific device. Either the device identifiers or the entire device object is returned.
getDevicesBehindDevices	Retrieves the list of devices downstream of the specified devices. Either the device IDs or the entire device object is returned.
updateDevice	Updates the properties of the specified device. **Note** While updating a device, if FQDN, host and domain details are provided, one of the details gets ignored. It is recommended not to provide all three details.
updateDevices	This operation applies the data contained in the specified device object to all devices specified in the lit of device IDs. This operation will apply the same update to all specified devices. Those properties excluded are: deviceIds, hostName, fqdn, embeddedDevices.
deleteDevice	Deletes the specified device.
deleteDevices	Deletes the specified devices.
unregisterDevice	Unregisters a device.
unregisterDevices	Unregisters multiple devices.

Table 3-2 PWS APIs (continued)

API name	Description
getDHCPLeaseInfo	Retrieves all known DHCP lease information about the specified IP address.
regenConfigs	Submits a request to regenerate configurations for the set of devices that match the specified search criteria.
rebootDevice	Reboots a device.
addDeviceType	Submits a request for the addition of a new device type.
getDeviceTypes	Retrieves all the device types available in the database.
updateDeviceTypes	Updates the specified device type.
deleteDeviceType	Deletes the specified device type.
deviceOperation	A generic operation that sends an opaque command and parameters to all specified devices.
addClassOfService	Submits a request for the addition of a new class of service.
getClassOfService	Retrieves data associated with the specified CoS.
updateClassOfService	Updates the properties of the specified CoS object.
deleteClassOfService	Deletes the specified CoS.
addDHCPCriteria	Submits a request for the addition of a new DHCPCriteria.
getDHCPCriteria	Retrieves the data associated with the specified DHCPCriteria.
updateDHCPCriteria	Updates the properties of the specified DHCPCriteria object.
deleteDHCPCriteria	Deletes the specified DHCPCriteria.
addFile	Submits a request for the addition of a new file.
getFile	Retrieves data associated with the specified file.
updateFile	Updates the properties or data of the specified file object.
deleteFile	Deletes the specified file.
addGroup	Adds a new group.
getGroup	Retrieves data associated with the specified group name.
updateGroup	Updates the properties of the specified group.
deleteGroup	Deletes the specified group.
pollOperationStatus	Queries for the status of RDU asynchronous requests specified by the transaction identifier.
Search	This is a generic operation. Retrieves devices, CoS, files, DHCPCriterion, or groups based on the search criterion defined in search object.

Asynchronous Service

Prime Cable Provisioning web service provides an asynchronous service to support asynchronous operations. Asynchronous service supports operations that are non-blocking for the client, i.e. client execution continues immediately after they submit asynchronous request, and the response is then processed later, likely by a different execution context or thread. The batch identifier can then be used to poll for the status of asynchronous or outstanding operations as well as the operations submitted as reliable batches.

Asynchronous service facilitates the following use cases:

- When response time is expected to be too long to wait for the response (For example, addDevices)
- When response time is not predictable
- When client needs non-blocking batch execution

To achieve effective asynchronous service calls, Prime Cable Provisioning ensures the following:

- Long persistence of the batch results for the asynchronous requests for later retrieval.
- Correlation between the requests and associated responses, at the RDU, PWS, and client side.

Session Management

A web services client must create a session for any communication with the PWS. To create a session with the PWS, the client must provide a valid context object in the request. A valid context object can either include information such as the RDU user's authentication information (username and password), the RDU details (host name and port number) that the request is targeted for or a unique session identifier. It could also have both the information. Upon authentication of these details, a session is created between the client and the PWS and the same session identifier is returned in the response.

PWS uses two types of sessions to communicate with the client:

- To interact with the PWS, a client must provide authentication information and identify the RDU the request is targeted for. Individual requests contain the user's username and password and the RDU information (host name and port details) included in a context object. This context object is sent to the PWS which it uses for authentication for every request that comes from the client.
- Alternatively, in case of multiple requests, a session between the client and PWS can be created and reused across all requests. Upon successful authentication, a session identifier is created and returned to the client. The client should use this identifier for all requests belonging to the same session. A session is per client per RDU and is bound to the capabilities of the roles and privileges assigned to the client. The context object identifies the client, contains client authentication token, and identifies the RDU that the client wants to interact with. All client requests must include the context.

  Upon completing the interaction, the client closes the session. The session also gets closed automatically after a configurable period of idle time. The default idle timeout is 15 minutes and can be changed using a CLI command. Sessions also get closed whenever the PWS is restarted.

Transactionality

The PWS operations can be classified into two categories:

- Single device operations - One operation for a single device. For single device operations, an operation is a single transaction in which all the changes are made or no change is made to the device.
- Multiple device operations - One operation for multiple devices. For multiple device operations, the transaction scope can be set to include all devices, or have each device change its individual transaction.

You can set the execution option, transactionPerItem, to true to retrieve the transaction data for each batch. The OperationStatus returned will contain the status of the entire operation as well as the individual status if multiple transactions are used.

Error Handling

Prime Cable Provisioning uses SOAP faults to relay information regarding issues encountered in validating, processing, or executing client requests. The primary exception raised by operations is the ProvServiceException. The code and message contained in the exception describes the cause of the fault. This exception is used to relay issue raised by the RDU. PWS provides an information level log.

Device Provisioning Engines

The Device Provisioning Engine (DPE) communicates with CPE to perform provisioning and management functions.

The RDU generates DHCP instructions and device configuration files, and distributes them to the relevant DPE servers. The DPE caches these DHCP instructions and device configuration files. The DHCP instructions are then used during interactions with the Network Registrar extensions, and configuration files are delivered to the device via the TFTP service.

Cisco Prime Cable Provisioning supports multiple DPEs. You can use multiple DPEs to ensure redundancy and scalability.

The DPE handles all configuration requests, including providing configuration files for devices. It is integrated with the Network Registrar DHCP server to control the assignment of IP addresses for each device. Multiple DPEs can communicate with a single DHCP server.

In the DPE, the configurations are compressed using Delta Compression technique of RFC 328 to reduce overall DPE cache size for better scalability.

The DPE manages these activities:

- Synchronizes with the RDU to retrieve the latest configurations for caching.
- Generates last-step device configuration (for instance, DOCSIS timestamps).
- Provides the DHCP server with instructions controlling the DHCP message exchange.
- Delivers configuration files via TFTP.
- ToD server
- Integrates with Network Registrar.
- Provisions voice-technology services.

Configure and manage the DPE from the CLI, which you can access locally or remotely via Telnet. For specific information on the CLI commands that a DPE supports, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Guide*.

DPE Licensing

Licensing controls the number of DPEs (nodes) that you can use. If you attempt to install more DPEs than you are licensed to use, those new DPEs will not be able to register with the RDU, and will be rejected. Existing licensed DPEs remain online.

Note For licensing purposes, a registered DPE is considered to be one node.

When you add a license or extend an evaluation license or when an evaluation license has expired, the changes take effect immediately.

When you delete a registered DPE from the RDU database, a license is freed. Because the DPEs automatically register with the RDU, you must take the DPE offline if the intention is to free up the license. Then, delete the DPE from the RDU database via the Admin UI or via the API.

Deleted DPEs are removed from all the provisioning groups that they belong to, and all Network Registrar extensions are notified that the DPE is no longer available. Consequently, when a previously deleted DPE is registered again, it is considered to be licensed again and remains so until it is deleted from the RDU again or its license expires.

DPEs that are not licensed through the RDU do not appear in the Admin UI. You can determine the license state only by examining the DPE and RDU log files (*dpe.log* and *rdu.log*).

Note The functions enabled via a specific license continue to operate even when the corresponding license is deleted from the system.

For detailed information on licensing, see Managing Licenses, page 2-1.

For important information related to DPEs, see:

- DPE CLI Authentication, page 3-29
- DPE-RDU Synchronization, page 3-31
- TFTP Server, page 3-32
- ToD Server, page 3-33

Also, familiarize yourself with the information described in Provisioning Concepts, page 3-37.

DPE CLI Authentication

There are two authentication modes used for DPE CLI:

- TACACS+ Authentication, page 3-30
- Radius Authentication, page 3-30

See the *Cisco Prime Cable Provisioning 5.0 DPE CLI Guide,* for details about how to configure TACACS+ and Radius authentication in DPE CLI.

TACACS+ Authentication

TACACS+ is a TCP-based protocol that supports centralized access for large numbers of network devices and user authentication for the DPE CLI.

Through TACACS+, a DPE CLI can support many users, with each username and password configured at the TACACS+ server. TACACS+ is used to implement the TACACS+ client/server protocol (ASCII login only).

TACACS+ Privilege Levels

The TACACS+ server uses the TACACS+ protocol to authenticate any user logging in to a DPE CLI. The TACACS+ client specifies a certain service level that is configured for the user.

Table 3-3 identifies the two service levels used to authorize DPE CLI user access.

Table 3-3 TACACS+ Service Levels

Mode	Description
Login	User-level commands at *router>* prompt.
Enable	Enable-level commands at *router#* prompt.

TACACS+ Client Settings

A number of properties that are configured from the DPE CLI are used for TCAACS+ authentication. For information on commands related to TACACS+, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.

When TACACS+ is enabled, you must specify either the IP addresses of all TACACS+ servers or their fully qualified domain names (FQDNs) with non-default values.

You can also specify these settings using their default values, if applicable:

- The shared secret key for each TACACS+ server. Using this key, you can encrypt data between the DPE and the TACACS+ server. If you choose to omit the shared secret for any specific TACACS+ server, TACACS+ message encryption is not used.
- The TACACS+ server timeout. Using this value, you can specify the maximum length of time that the TACACS+ client waits for a TACACS+ server to reply to protocol requests.
- The TACACS+ server number of retries. Using this value, you can specify the number of times that the TACACS+ client attempts a valid protocol exchange with a TACACS+ server.

Radius Authentication

DPE CLI supports Radius authentication for authenticating the users logging on to DPE CLI. Radius authentication are of two modes and they are as follows:

Without Two-Factor:

In this mode, username and password are required to log on to DPE CLI.

Two-Factor:

In two-factor authentication mode, the user has to provide the username and, the passcode which is a combination of PIN and Token Code to log on to DPE CLI. The RSA SecureID generates the Token Code which will be updated every 60 seconds in the RSA SecureID device.

Radius Privilege Levels

The Radius server authenticates the user logging on to a DPE CLI. The Radius client settings specifies certain privilege levels that are configured for the user.

Table 3-4 describes the service levels used to authorize the DPE CLI user.

Table 3-4 Radius Service Levels

Mode	Description
Login	User-level commands at *router>* prompt.
Enable	Enable-level commands at *router#* prompt.

DPE-RDU Synchronization

The DPE-RDU synchronization is a process of automatically updating the DPE cache to be consistent with the RDU. The DPE cache comprises the configuration cache, with configurations for devices, and the file cache, with files required for devices.

Under normal conditions, the RDU generates events containing configuration updates and sends them to all relevant DPEs to keep them up to date. Synchronization is needed if the DPE is missing some events due to connection loss. Such loss could be because of a network issue, the DPE server going down for administrative purposes, or a failure.

Synchronization also covers the special case when the RDU database is restored from backup. In this case, the DPE cache database must be returned to an older state to be consistent with the RDU.

The RDU and DPE synchronization process is automatic and requires no administrative intervention. Throughout the synchronization process, the DPE is still fully capable of performing provisioning and management operations on the CPE.

Synchronization Process

The DPE triggers the synchronization process every time it establishes a connection with the RDU.

When the DPE first starts up, it establishes the connection to the RDU and registers with the RDU to receive updates of configuration changes. The DPE and RDU then monitor the connection using heartbeat message exchanges. When the DPE determines that it has lost its connection to the RDU, it automatically attempts to re-establish it. It continues its attempts with a backoff-retry interval until it is successful.

The RDU also detects the lost connection and stops sending events to the DPE. Because the DPE may miss the update events from the RDU when the connection is down, the DPE performs synchronization every time it establishes a connection with the RDU.

During the process of synchronization, the DPE is in the following states:

1. Registering—During the process of establishing a connection and registering with the RDU, the DPE is in the *Registering* state.
2. Synchronizing—The DPE requests groups of configurations that it should have from the RDU. During this process, the DPE determines which configurations in its store are inconsistent (wrong revision number), which ones are missing, and which ones to delete, and, if necessary, updates the configurations in its cache. The DPE also synchronizes deliverable files in its cache for the TFTP server. To ensure that the RDU is not overloaded with configuration requests, the DPE posts only one batch at a time to the central server.

3. Ready— The DPE is up to date and fully synchronized with the RDU. This state is the typical state that the DPE is in.

Table 3-5 describes some other states that the DPE may be in from time to time.

Table 3-5 Related DPE States

State	Description
Initializing	Is starting up
Shutting Down	Is in the process of stopping
Down	Does not respond to queries from Network Registrar extension points
Ready Overloaded	Is similar to *Ready* except that there is a heavy load on the system on which the DPE is running

Note Regardless of the state that the DPE is in, it continues to service device configuration, TFTP, and ToD requests.

You can view the DPE state:

- From the administrator user interface. See Monitoring DPE, page 21-5.
- From the DPE CLI using the **show dpe** command. See the *Cisco Prime Cable Provisioning 5.0 DPE CLI Guide* for more information.

TFTP Server

The integrated TFTP server receives requests for files, including DOCSIS configuration files, from device and nondevice entities. This server then transmits the file to the requesting entity.

Enable the TFTP server in DPE to access the local file-system. The local files are stored in the *<BPR_DATA>/dpe/tftp* directory. All deliverable TFTP files are precached in the DPE; in other words, the DPE is always up to date with all the files in the system.

Note The TFTP service on the DPE features one instance of the service, which you can configure to suit your requirements.

By default, the TFTP server only looks in its cache for a TFTP read. However, if you run the **service tftp** *1..1* **allow-read-access** command from the DPE command line, the TFTP server looks in the local file system before looking in the cache. If the file exists in the local file system, it is read from there. If not, the TFTP server looks in the cache. If the file exists in the cache, the server uses it; otherwise, it returns an error.

When you can enable read access from the local file system, directory structure read requests are allowed only from the local file system.

Note Ensure that you give unique names to all TFTP files instead of differentiating the files by using upper or lowercase. The filename casing is important because the DPE, while looking for a file in its local directory or cache, converts all filenames to lowercase.

You can specify TFTP transfers using IPv4 or IPv6, using the **service tftp** *1..1* **ipv4 | ipv6 enabled true** command from the DPE command line. You can also specify a block size for these transfers using the **service tftp** *1..1* **ipv4 | ipv6 blocksize** command. The blocksize option specifies the number of data octets and allows the client and server to negotiate a block size more applicable to the network medium. When you enable blocksize, the TFTP service uses the requested block size for the transfer if it is within the specified lower and upper limits. For detailed information, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Guide* for more information.

The TFTP service maintains statistics for the number of TFTP packets that are processed for TFTPv4 and TFTPv6. You can view these statistics from the administrator user interface on the device details page. For more information, see Viewing Device Details, page 18-4.

ToD Server

The integrated time of day (ToD) server in Prime Cable Provisioning provides high-performance UDP implementation of RFC 868.

Note The ToD service on the DPE features one instance of the service, which you can configure to suit your requirements.

You can enable the ToD service to support IPv4 or IPv6, from the DPE command line, using the **service tod** *1..1* **enabled true** command. The ToD service is, by default, disabled on the DPE.

While configuring this protocol on the DPE, remember that the ToD service binds only to those interfaces that you have configured for provisioning. For detailed information on configuring the ToD service, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Guide* for more information.

The ToD service maintains statistics for the number of ToD packets that are processed for ToDv4 and ToDv6. You can view these statistics from the administrator user interface on the device details page. For more information, see Viewing Device Details, page 18-4.

Cisco Prime Network Registrar

Cisco Prime Network Registrar provides the DHCP and DNS functionality in Prime Cable Provisioning. The DHCP extension points on Network Registrar integrate Prime Cable Provisioning with Network Registrar. Using these extensions, Prime Cable Provisioning examines the content of DHCP requests to detect device type, manipulates the content according to its configuration, and delivers customized configurations for devices that it provisions.

For additional information on Cisco Prime Network Registrar, see the *User Guide for Cisco Network Registrar 8.1*; *Command Reference Guide for Cisco Network Registrar 8.1;* and *Installation Guide for Cisco Network Registrar, 8.1*.

DHCP

The DHCP server automates the process of configuring IP addresses on IP networks. The protocol performs many of the functions that a system administrator carries out when connecting a device to a network. DHCP automatically manages network-policy decisions and eliminates the need for manual configuration. This feature adds flexibility, mobility, and control to networked device configurations.

This Prime Cable Provisioning release supports DHCP for IPv6, also known as DHCPv6. DHCPv6 enables DHCP servers to deliver configuration parameters, via extensions, to IPv6 hosts. IPv6 hosts by default use stateless auto-configuration, which enables IPv6 hosts to configure their own addresses using a local IPv6 router. DHCPv6 represents the stateful auto-configuration option, a technique in which configuration information is provided to a host by a server.

DHCPv6 provides:

- Expanded addressing capabilities via IPv6 addresses
- Easy network management and administration using the stateful auto-configuration protocol
- Improved support for options and extensions
- DHCPv6 relay agent encapsulates all the incoming packets (from a client or another relay agent) into a Relay-Forward message. Hence, maintaining a clean separation between client and relay agent options which makes DHCPv6 more desirable compared to DHCPv4.
- Assignment of multiple addresses to one interface

DHCPv4 versus DHCPv6

Much like DHCPv4, DHCPv6 uses a client-server model. The DHCP server and the DHCP client converse with a series of messages to request, offer, and lease an IP address. Unlike DHCPv4, DHCPv6 uses a combination of unicast and multicast messages for the bulk of the conversation instead of broadcast messages.

Some other differences between DHCPv4 and DHCPv6 are:

- Unlike DHCPv4, IPv6 address allocation in DHCPv6 is handled using a message option.
- Message types, such as DHCP Discover and DHCP Offer supported by DHCPv4 are removed in DHCPv6. Instead, DHCPv6 servers are located by a client Solicit message followed by a server Advertise message.
- Unlike DHCPv4 clients, DHCPv6 clients can request multiple IPv6 addresses.

DHCPv4 failover allows pairs of DHCP servers to act in such a way that one can take over if the other stops functioning. The server pairs are known as the main and backup server. Under normal circumstances, the main server performs all DHCP functions. If the main server becomes unavailable, the backup server takes over. In this way, DHCP failover prevents loss of access to the DHCP service if the main server fails. Currently DHCPv6 failover is not supported.

DNS

The DNS server contains information on hosts throughout the network, such as IP address host names. DNS uses this information primarily to translate between IP addresses and domain names. The conversion of names such as www.cisco.com to IP addresses simplifies accessing Internet-based applications.

Cisco Prime Network Registrar provide separate DNS servers for authorization and caching. It also supports DNS64, DNSSEC and full IPv6.

DNSv4 versus DNSv6

Much like DNSv4, DNSv6 uses a client-server model. The DNS client requests the DNS server to resolve the DNS name, during conversation DNSv6 uses IPv6, whereas DNSv4 uses IPv4.

Some other differences between DNSv4 and DNSv6 are:

- DNSv6 uses new AAAA RR type for name to address information
- DNS updates are just like DNSv4, except multiple AAAAs under a single client name
- IN DNSv6 PTR RR are under ip6.arpa and in DNSv4 it is under in-addr.arpa

HA DNS allows pairs of DNS servers to act in such a way that one can take over if the other stops functioning. The server pairs are known as the main and backup server. Under normal circumstances, the main server performs all DNS functions. If the main server becomes unavailable, the backup server takes over. In this way, HA DNS prevents loss of Dynamic updates and query translations.

Lease Query

The lease query feature allows you to request current IP address information directly from the Network Registrar DHCP servers in a provisioning group. To find a device's IP address, the RDU sends DHCP lease query messages only to the DHCP servers in the device's provisioning group, which prevents querying all DHCP servers in the network. Among all the responses, the response from the server that last communicated with the devices is taken as the authoritative answer.

In earlier Prime Cable Provisioning versions, the lease query feature relied on the operating system to select the source interface and the source port for sending lease query requests. In this release, you can configure the RDU to use a specific interface and source port.

Note When you install all components in the same Linux server, Prime Cable Provisioningsco Prime Network Registrar will not respond to the lease queries from RDU.

For detailed information on lease query support in this Prime Cable Provisioning release, see Chapter 15, “Lease Query”.

Key Distribution Center

The Key Distribution Center (KDC) authenticates PacketCable MTAs and also grants service tickets to MTAs. As such, it must check the MTA certificate, and provide its own certificates so that the MTA can authenticate the KDC. It also communicates with the DPE (the provisioning server) to validate that the MTA is provisioned on the network.

The certificates used to authenticate the KDC are not shipped with Cisco Prime Cable Provisioning. You must obtain the required certificates from Cable Television Laboratories, Inc. (CableLabs), and the content of these certificates must match those that are installed in the MTA. For additional information, see Using PKCert.sh, page 28-3.

Caution The KDC does not function if the certificates are not installed.

The KDC also requires a license to function. Obtain a KDC license from your Cisco representative and install it in the correct directory. For details on how to install the license, see KDC Licenses, page 6-17.

The KDC has several default properties that are populated during a Cisco Prime Cable Provisioning installation into the BPR_HOME/kdc/<Operating System>/kdc.ini properties file, for Solaris. For Linux, the path is /opt/CSCObac/kdc/linux/kdc.ini. You can edit this file to change values as operational requirements dictate. For detailed information, see Default KDC Properties, page 6-15.

The KDC also supports the management of multiple realms. For details on configuring additional realms, see Multiple Realm Support, page 6-18.

Note KDC is only required for PacketCable Secure mode.

Process Watchdog

The Prime Cable Provisioning process watchdog is an administrative agent that monitors the runtime health of all Prime Cable Provisioning processes. This watchdog process ensures that if a process stops unexpectedly, it is automatically restarted. One instance of the Prime Cable Provisioning process watchdog runs on every system which runs Prime Cable Provisioning components.

You can use the Prime Cable Provisioning process watchdog as a command-line tool to start, stop, restart, and determine the status of any monitored processes.

See Chapter 23, “Prime Cable Provisioning Process Watchdog”, for additional information on how to manage the monitored processes.

SNMP Agent

Prime Cable Provisioning provides basic SNMP v2-based monitoring of the RDU and DPE servers. The Prime Cable Provisioning SNMP agents support SNMP informs and traps, collectively called notifications.

You can configure the SNMP agent:

- On the RDU, using the SNMP configuration command-line tool (see Monitoring Servers Using SNMP, page 22-1) or via the API.
- On the DPE, using the **snmp-server** CLI commands. See the *Cisco Prime Cable Provisioning 5.0 DPE CLI Guide* for more information.

Administrator User Interface

The Prime Cable Provisioning Admin UI is a web-based application for central management of the Prime Cable Provisioning system. You can use this system to:

- Configure global defaults
- Define custom properties
- Add, modify, and delete Class of Service
- Add, modify, and delete DHCP Criteria
- Add, modify, and delete devices
- Group devices

- View server status and server logs
- Manage users
- Manage user groups
- Manage roles
- Manage domain

See these chapters for specific instructions on how to use this interface:

- Chapter 4, "Accessing Admin UI," describes how to access and configure the Prime Cable Provisioning Admin UI.
- Chapter 11, "Configuring Prime Cable Provisioning Using Admin UI," provides instructions for performing administrative activities involving the monitoring of various Prime Cable Provisioning components.
- Chapter 18, "Provisioning Devices Using Admin UI," describes tasks that you perform to configure devices using the Admin UI.
- Chapter 12, "Configuring Groups Using Admin UI," describes tasks that you perform to configure groups using the Admin UI.
- Chapter 21, "Monitoring Servers Using Admin UI," describes tasks that you perform to check the status of Prime Cable Provisioning servers using the Admin UI.
- Chapter 13, "Configuring RBAC Using Admin UI," describes tasks that you perform to add users, groups and map them. It also describes role management and domain management.

Provisioning Concepts

This section describes those concepts that are key to provisioning and include:

- Provisioning Groups, page 3-37
- Static versus Dynamic Provisioning, page 3-38
- Provisioning Group Capabilities, page 3-39

Provisioning Groups

A provisioning group is designed to be a logical (typically geographic) grouping of servers that usually consists of one or more DPEs and a failover pair of DHCP servers. Each DPE in a given provisioning group caches identical sets of configurations from the RDU, thus enabling redundancy and load balancing. As the number of devices grows, you can add additional provisioning groups to the deployment.

Note The servers for a provisioning group are not required to reside at a regional location. They can just as easily be deployed in the central network operations center.

Provisioning groups enhance the scalability of the Cisco Prime Cable Provisioning deployment by making each provisioning group responsible for only a subset of devices. This partitioning of devices can be along regional groupings or any other policy that the service provider defines.

To scale a deployment, the service provider can:

- Upgrade existing DPE server hardware
- Add DPE servers to a provisioning group
- Add provisioning groups

To support redundancy and load sharing, each provisioning group can support any number of DPEs. As the requests come in from the DHCP servers, they are distributed between the DPEs in the provisioning group and an affinity is established between the devices and a specific DPE. This affinity is retained as long as the DPE state within the provisioning group remains stable.

Static versus Dynamic Provisioning

Cisco Prime Cable Provisioning provisions devices in the network using device configurations, which is provisioning data for a specific device based on its technology type. You can provision devices using Cisco Prime Cable Provisioning in two ways: static provisioning and dynamic provisioning.

During static provisioning, you enter static configuration files into the Cisco Prime Cable Provisioning system. These configuration files are then delivered via TFTP to the specific device.
Cisco Prime Cable Provisioning treats static configuration files like any other binary file.

During dynamic provisioning, you use templates or scripts, which are text files containing DOCSIS, PacketCable, or CableHome options and values that, when used with a particular Class of Service, provide dynamic file generation. A dynamic configuration file provides more flexibility and security during the provisioning process.

Table 3-6 describes the impact of static and dynamic provisioning using the corresponding files.

Table 3-6 Static Provisioning versus Dynamic Provisioning

Static Provisioning Using Static Files	Dynamic Provisioning Using Template Files
Used when fewer service offerings are available	Used when many service offerings are available
Offers limited flexibility	Offers more flexibility, especially when devices require unique configurations
Is relatively less secure	Is more secure
Offers higher performance	Offers slower performance, because every time you update a template or a groovy script assigned to a device, configurations for all devices associated with that template are updated.
Is simpler to use	Is more complex

Provisioning Group Capabilities

To provision a subset of devices in a deployment, provisioning groups must be capable of as well as enabled to provision those devices. For example, to provision a DOCSIS 3.0 modem in IPv4 mode, you must enable the IPv4 - DOCSIS 3.0 capability. Following is the list of prominently used capabilities:

* IPv4 - DOCSIS 1.0/1.1

* IPv4 - DOCSIS 2.0

* IPv4 - DOCSIS 3.0

* IPv4 - PacketCable

* IPv4 - CableHome

* IPv6 - DOCSIS 3.0

In previous Cisco Prime Cable Provisioning releases, each DPE in a provisioning group registered what it was capable of supporting with the RDU at startup. After server registration, the provisioning group was automatically enabled to support the device types it was capable of supporting. In Cisco Prime Cable Provisioning 5.0, you must enable device support or capabilities manually.

- From the Admin UI, on the Provisioning Group Details page (see Monitoring Provisioning Groups, page 21-3).
- From the API, using the *ProvGroupCapabilitiesKeys* constants. For details, see the API Javadoc located at the *docs* directory of the build.

Component Based Log Files

Logging of events is performed at the RDU, DPE and PWS, and in some unique situations, DPE events are additionally logged at the RDU to give them higher visibility. Log files are stored in their own log directories and can be examined by using any text processor. You can compress the files for easier e-mailing to the Cisco Technical Assistance Center or system integrators for troubleshooting and fault resolution. You can also access the RDU and the DPE logs from the Admin UI.

You can generate server configuration and other diagnostics information using the diagnostics tools in the *BPR_HOME/*{*rdu* | *dpe*}*/diagnostics/bin* directory. For details see Bundling Server State for Support, page 26-10.

For detailed information on log levels and structures, and how log files are numbered and rotated, see Log Levels and Structures, page 25-1.

CHAPTER 4

Accessing Prime Cable Provisioning Interfaces

This chapter describes how to set up and access the Cisco Prime Cable Provisioning user interfaces. There are two user interfaces to access Prime Cable Provisioning, Admin UI and DPE CLI. You must have a valid Prime Cable Provisioning license for these two interfaces else some configurations might fail.

- Accessing Admin UI, page 4-1
- Accessing DPE CLI, page 4-4

Accessing Admin UI

You can access the Prime Cable Provisioning Admin UI from any computer that can access the URL corresponding to the Prime Cable Provisioning application.

Note To view the new Admin UI pages, you must manually clear the browser cache after upgrading from an earlier version (4.0.1.x, 4.1.0.x or 4.2.x) to Prime Cable Provisioning 5.0.

Logging In

You can log in to the Prime Cable Provisioning Admin UI if you have a user account either in the RDU or in a remote server such as Radius or TACACS used for authentication. For details about user access and privileges see Configuring Security, page 13-1.

Complete this procedure to access the Prime Cable Provisioning Admin UI:

Step 1 Launch your web browser. Prime Cable Provisioning can be launched using the following browsers:

- Firefox 10 or higher
- Google Chrome 17 and higher
- Internet Explorer 9

Step 2 Enter the administrator's location using this syntax:

```
http://machine_name:port_number/
```

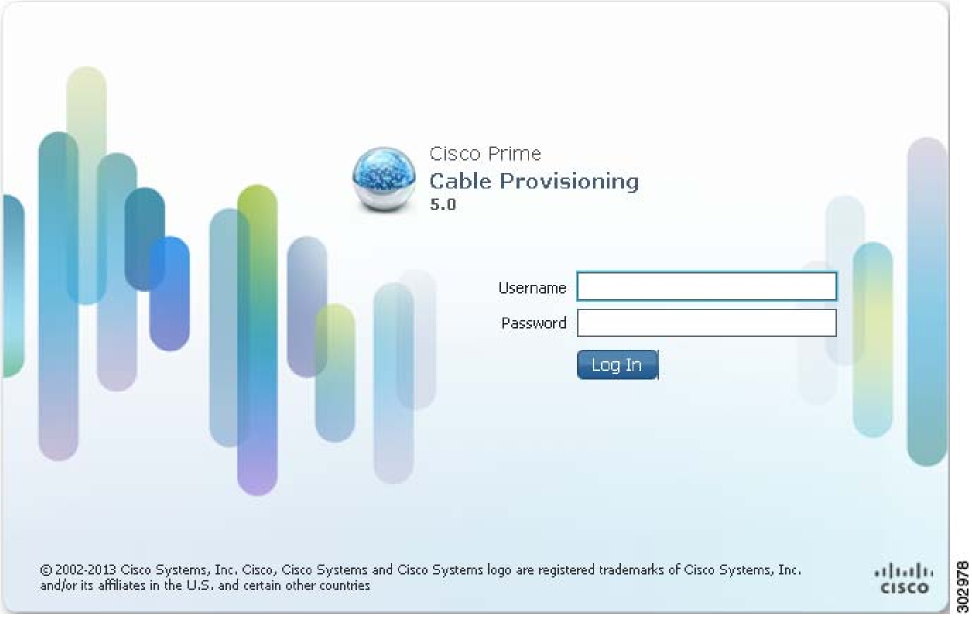
- *machine_name*—Identifies the computer on which the RDU is running.

- *port_number*—Identifies the computer port on which the server side of the administrator application runs. The default port number is:
  - 8100 for HTTP over TCP
  - 8443 for HTTP over SSL

Note By default, Prime Cable Provisioning redirects all HTTP communications over HTTPS. If you want to bypass the HTTPS redirection, set the property adminui/redirectToHttps to false in the admin.properties file. For more details about this property file, see Using admin.properties.

The main login page, as shown in Figure 4-1, appears.

Figure 4-1 Login Page


Cisco Prime
Cable Provisioning
5.0
Username
Password
Log In
© 2002-2013 Cisco Systems, Inc. Cisco, Cisco Systems and Cisco Systems logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and certain other countries
cisco
302978

Step 3 Enter the default username (**admin**) and default password (**changeme**).

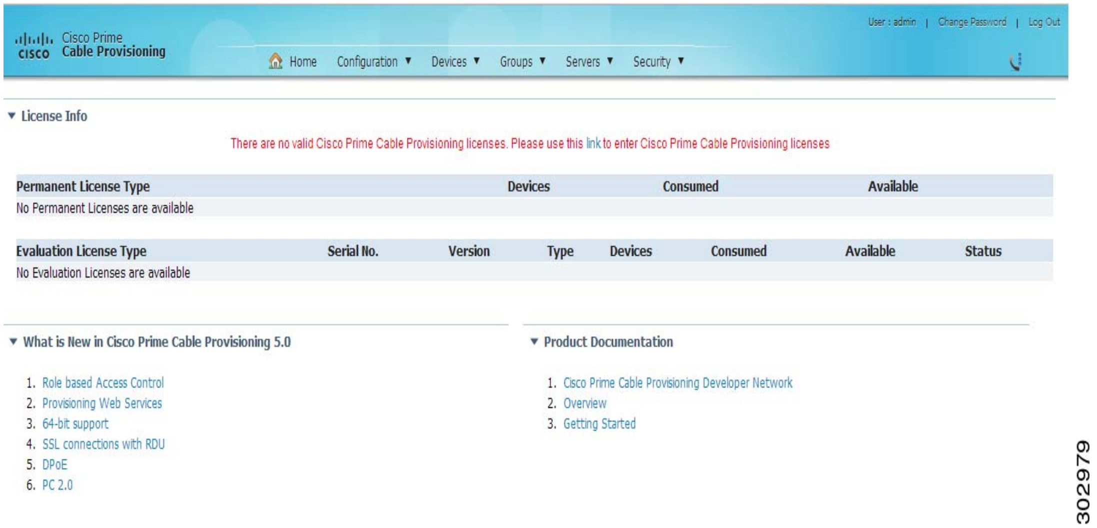
If you are logging in for the first time, the Change Password screen appears.

Enter a new password and confirm it. Ensure that the password that you enter has at least 8 characters.

Step 4 Click **Login**.

The Main Menu page, as shown in Figure 4-2, appears.

Figure 4-2 Main Menu Page



▼ License Info

There are no valid Cisco Prime Cable Provisioning licenses. Please use this link to enter Cisco Prime Cable Provisioning licenses

Permanent License Type	Devices	Consumed	Available
No Permanent Licenses are available			

Evaluation License Type	Serial No.	Version	Type	Devices	Consumed	Available	Status
No Evaluation Licenses are available							

▼ What is New in Cisco Prime Cable Provisioning 5.0

1. Role based Access Control
2. Provisioning Web Services
3. 64-bit support
4. SSL connections with RDU
5. DPoE
6. PC 2.0

▼ Product Documentation

1. Cisco Prime Cable Provisioning Developer Network
2. Overview
3. Getting Started

Note You can use the link at the top of the page to add the license file. For details, see Chapter 2, “Managing Licenses”.

Admin UI Operations

When you log into Prime Cable Provisioning, the portlets that you see on the home page depend on your user privileges. The operations that can be performed using the Admin UI are explained in the following chapters:

- Chapter 11, “Configuring Prime Cable Provisioning Using Admin UI”
- Chapter 18, “Provisioning Devices Using Admin UI”
- Chapter 13, “Configuring RBAC Using Admin UI”
- Chapter 12, “Configuring Groups Using Admin UI”
- Chapter 21, “Monitoring Servers Using Admin UI”

Changing Password

All users of Prime Cable Provisioning can change their password. For better security this operation is performed in the HTTPS mode.

To change your password:

Step 1 Click the **Change Password** tab at the top-right corner of any page.

A Change Password dialog box appears.

Step 2 Enter a new password and confirm it. Ensure that the password that you enter has at least 8 characters

Step 3 Click **Submit**.

The application returns you to the Home page.

Logging Out

Complete this procedure to log out of Prime Cable Provisioning:

Step 1 Click the **Logout** tab at the top-right corner of any page.

A confirmation dialog box appears.

Step 2 Click **OK**.

The application returns you to the Login page. See Figure 4-1.

Accessing DPE CLI

The Prime Cable Provisioning CLI is an IOS-like command-line interface that you use to configure and view the status of the DPE by using Telnet or SSH. The CLI supports built-in command help and command autocompletion.

You can enable authentication of the CLI through a locally configured login and privileged passwords, or through a remote username and password for a TACACS+ service.

To access the DPE CLI, open a Telnet session to port 2323 from a local or remote host.

Accessing DPE CLI from a Local Host

To access the CLI from a local host, you can use:

# **telnet** *local_hostname* **2323**

or

# **telnet 0 2323**

Accessing DPE CLI from a Remote Host

To access the CLI from a remote host, enter:

```
# telnet remote-hostname 2323
```

Note If you cannot establish a Telnet connection to the CLI, the CLI server might not be running. You may need to start the server; enter:

```
# /etc/init.d/bprAgent start cli
```

After you access the CLI, you must enter the DPE password to continue. The default login and privileged passwords are **changeme**.

See the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide* for specific information on the CLI commands that a DPE supports.

Accessing SNMP Agent

Prime Cable Provisioning supports management of servers via SNMP. Specifically, an SNMP-based management system can be used to monitor Prime Cable Provisioning server state, license utilization information, server connections, and server-specific statistics

You can configure the SNMP agent:

- On the RDU, using the SNMP configuration command-line tool (see Monitoring Servers Using SNMP, page 22-1) or via the API.
- On the DPE, using the snmp-server CLI commands. See the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.

CHAPTER 5

Managing Database

This chapter contains information on RDU database management and maintenance. The Prime Cable Provisioning RDU requires virtually no maintenance other than to ensure availability of sufficient disk space. As the administrator, you must understand and be familiar with database backup and recovery procedures.

Understanding Failure Resiliency

The RDU database uses a technique known as *write-ahead logging* to protect against database corruption that could result from unforeseen problems, such as an application crash, system failure, or power outage.

Write-ahead logging involves writing a description of any database change to a database log file prior to writing the change into the database files. This mechanism allows the repair of any incomplete database writes that can result from system failures.

The RDU server performs an automatic recovery each time it is started. During this recovery process, the database log files are used to synchronize the data with the database files. Database changes that were written into the database log, but not into the database, are written into the database during this automatic recovery.

In this way, write-ahead logging virtually guarantees that the database does not become corrupted when the RDU server crashes because the database is automatically repaired when the RDU server is restarted.

Write-ahead logging requires these conditions to work properly:

- You must set up the file system and physical storage so that they guarantee that the data is flushed to physical storage when requested. For example, a storage system with volatile memory-only write cache, which loses data during system failure, is not appropriate. However, a disk array with battery-backed write cache which guarantees that the data gets persisted, even in the event of a system failure, is acceptable. A system without battery-backed write cache should flush the data disk when requested instead of performing in-memory data caching.
- You must set up the file system with an 8192-byte block size to match the RDU database block size. This setting is usually the default on Solaris unless explicitly adjusted.

Database Files

The RDU database stores data in binary files using the file system you have mounted on the partition containing the files. It is essential to choose and configure a file system in a way that it is not susceptible to long recovery times after system failures.

Database files are vital to the operation of the RDU. Therefore, take extra precautions to safeguard them against accidental removal or other manual manipulation. Follow standard system administration practices to safeguard these important files. For example, these files should always have permissions that allow only root user access. Additionally, it is a good practice to never log in to your production system as a root user. Instead, log in as a less privileged user and use the **sudo** command to execute tasks requiring root privileges.

Database Storage File

The RDU server stores its database in a file called bpr.db, which resides in the database directory. This directory resides in the BPR_DATA/rdu/db directory; you can configure this location by specifying the BPR_DATA parameter during a component installation. See Changing Database Location, page 5-7, for additional information on moving the database.

Note The database file is normally accessed in a random fashion. You should, therefore, select a disk with the fastest seek time and rotational access latency to obtain the best database performance.

Database Transaction Log Files

The RDU server stores database transaction logs in 25-MB files that are stored in the database log directory. You configure this directory during installation by specifying the BPR_DBLOG parameter. The log directory resides in the BPR_DBLOG/rdu/dblog directory. See Changing Database Location, page 5-7, for additional information on moving the transaction logs to a new directory.

Database log files are named in numeric sequence, starting at log.00000001, log.00000002, and so on.

Note The disk on which transaction logs are stored is usually accessed in a sequential manner, with data being appended to the log files. Select a disk that can efficiently handle this access pattern to achieve the best database performance. We recommend that you locate the database transaction logs directory on the fastest disk on the system. Also, ensure that 1 GB of disk space is available.

Automatic Log Management

Database transaction logs files are used to store transaction data until that data is completely written into the database. After that, the transaction log data becomes redundant and the files are then automatically removed from the system.

Under normal circumstances there should be only a few log files in the database transaction log directory. Over time, you will notice that older transaction logs disappear and newer ones are created.

Note Database transaction logs are an integral part of the database. Manual deletion or modification of transaction log files will result in database corruption.

Miscellaneous Database Files

The database directory contains additional files that are essential to database operation. These files, in addition to the rdu.db file, are found in the BPR_DATA/rdu/db directory and are copied as part of the database backup:

- DB_VERSION—Identifies the physical and logical version of the database and is used internally by the RDU.
- history.log—Used to log activity about essential database management tasks, such as automatic log file deletion, backup, recovery, and restore operations. In addition to providing useful historical information for the administrator, this log file is essential to RDU database operation.

Disk Space Requirements

The size of a fully populated database depends on:

- Device objects that are managed by the RDU
- Custom properties stored on each object

The approximate estimates for disk space required for each partition are:

- BPR_DATA, approximately 2 to 5 KB per device object
- BPR_DBLOG, at least 500 MB

Caution These requirements are provided as a guideline only and do not eliminate the need for normal system monitoring.

You can use the disk_monitor.sh tool to monitor available disk space and alert the administrator. See Using disk_monitor.sh, page 28-13, for additional information.

Handling Out of Disk Space Conditions

When the RDU server runs out of disk space, it generates an alert through the syslog facility and the RDU log. The RDU server then tries to restart automatically. When attempting to restart, the RDU server may again encounter the out of disk space error and attempt to restart again.

The RDU server continues trying to restart until free disk space becomes available. Once you free up some disk space on the disk that is operating near a limit, the next time the RDU restarts it will succeed.

If the size of your database grows beyond the capacity of the current disk partition, move the database to a new disk or partition. For more information, see Changing Database Location, page 5-7.

Note It is a good practice to monitor disk space utilization to prevent failure. See Using disk_monitor.sh, page 28-13, for additional information.

Backup and Recovery

The RDU server supports a highly efficient backup process that can be performed without stopping the server or suspending any of its activities. Database backup and recovery involves these stages:

- Backup—Takes a snapshot of the RDU database from a live server.
- Recovery—Prepares the database snapshot for reuse.
- Restore—Copies the recovered database snapshot to the RDU server.

Note Once migration is complete, you can optionally check for database consistency.

Automated tools are provided for each of these steps. You can use these tools in either interactive mode or silent mode, but you must have root privileges to use the tools.

Database Backup

Backup is the process of copying the database files into a backup directory. The files can then be compressed and placed on tape or other archive.

RDU database backup is highly efficient because it involves just copying files without interrupting server activity. However, because it involves accessing the RDU database disk, backup may adversely affect RDU performance. The opposite is also true. RDU activity happening during backup will adversely affect backup performance. Therefore, you should perform backups during off-peak hours.

Other than concurrent system activity, backup performance also depends on the underlying disk and file system performance. Essentially, backup will perform as fast as database files can be copied from source to target.

Use the backupDb.sh tool, in the BPR_HOME/rdu/bin directory, to perform database backups:

- To use this tool, you must provide the target directory where the backup files will be placed. This directory should be on a disk or partition that has available disk space equivalent to 120% of the current database file size.

- As illustrated in the following example, this tool automatically creates a timestamped subdirectory, under the directory you specify, and places the backups there. You can also use the optional **–nosubdir** flag to disable, if necessary, the automatic creation of the subdirectory.

Note To avoid database crash, Prime Cable Provisioning backup tool must be used if the backup is done while the RDU is running. The database backup made by other file backup tools may not be recoverable if the RDU modifies the database file during the backup.

The backupDb.sh command also reports progress to the screen and logs its activity in history.log.

When using the backupDb.sh tool, you can use a –help option to obtain usage information.

Examples

In this example, */var/backup* identifies the target location for database backup files.

```
# /backupDb.sh -nosubdir -throttle 500 /var/backup.

Database backup started
Back up to: /var/backup

Copying DB_VERSION. Size: 396 bytes.
DB_VERSION: 100% completed.

Copying bpr.db. Size: 434176 bytes.
bpr.db: 100% completed.

Copying log.0000000001. Size: 469268 bytes.
log.0000000001: 100% completed.

Copying history.log. Size: 574 bytes.
history.log: 100% completed.

Database backup completed
```

The timestamped subdirectory format is rdu-backup-yyyyMMdd-HHmmss. In this example, the subdirectory would be rdu-backup-20070316-031028, meaning that the directory contains a backup that was started at 3:10:28 a.m. on March 16, 2007.

Note Once migration is complete, you can run the verifyDb.sh tool to check the integrity of the database. Verification is an optional task. It is a resource-intensive operation and should be performed on the system using the backup of the database. Run the verifyDb.sh script, which resides in the BPR_HOME/rdu/internal/db/bin directory. In case of a large database, the heap size must be set to a higher value. For example, -Xms1024M -Xmx2048M.

Database Recovery

Database recovery is the process of restoring the database to a consistent state. Because backup is performed on a live RDU, the database can be changing while it is being copied. The database log files, however, ensure that the database can be recovered to a consistent state.

Recovery is performed on a snapshot of a database. In other words, this task does not involve touching the database on the live RDU server. The recovery task can be performed either immediately following a backup or prior to restoring the database to the RDU server.

Note We recommend that you perform database recovery immediately after each backup. This way, the backed-up database can be more quickly restored in case of emergency.

The duration of database recovery depends on the number of database log files that were copied as part of the backup, which in turn depends on the level of RDU activity at the time of the backup. The more concurrent activity RDU experiences during the backup, the more transaction log files have to be copied as part of the backup and the longer the recovery takes. Generally, recovering a database takes from 10 to 60 seconds per transaction log file.

Use the recoverDb.sh tool, located in the BPR_HOME/rdu/bin directory, to perform recovery of the snapshot of a database. When you use this tool, you must provide the location of the backup. This is also the directory in which the recovery will be performed.

When using the recoverDb.sh tool, you can use the –help option to obtain usage information on the tool.

Examples

```
# recoverDb.sh /var/backup/rdu-backup-20070316-031028

**********************************************************
*                                                        *
*  Recovery process modifies the backup snapshot of      *
*  the database.  You should never do recovery without   *
*  making a copy of the database and log files you       *
*  are using for recovery.                               *
*                                                        *
**********************************************************

To start recovery please type "yes" and enter: yes


Database recovery started
Recovering in: /var/backup/rdu-backup-20070316-031028
This process may take a few minutes.
Database recovery completed
```

In this example, the snapshot located in the /var/backup/rdu-backup-20070316-031028 directory is recovered to a consistent state. The progress of the recovery operation appears on screen and the activity is recorded in the history.log file in the snapshot directory.

Database Restore

Restoring the database is the process of copying the previously recovered database snapshot to the database location used by the RDU server. Only a database that has been previously recovered can be restored.

Because restoring the database means replacing the current RDU database, it is very important that you first properly remove and archive the old database.

Caution Do not delete the database you are replacing. You might need a copy of an old database to simplify future system diagnostics.

Use the restoreDb.sh tool, which resides in the BPR_HOME/rdu/bin directory, to replace the current RDU database with another database. When using this tool, you must specify an input directory. This directory must contain the recovered backup snapshot of the database to be restored to the RDU server.

Note Before running the restoreDb.sh tool, you must stop the RDU server by running the /etc/init.d/bprAgent stop rdu command. Also, remember to back up the database, then remove all files from the rdu/db and the rdu/dblog directories.

When using the restoreDb.sh tool, you can use the –help option to obtain usage information.

Examples

```
# restoreDb.sh /var/backup/rdu-backup-20070316-031028

Restoring RDU database...
Restoring from: /var/backup/rdu-backup-20070316-031028

Copying bpr.db. Size: 434176 bytes.
bpr.db: 100% completed.

Copying log.0000000001. Size: 471261 bytes.
log.0000000001: 100% completed.

Copying history.log. Size: 1260 bytes.
history.log: 100% completed.

Copying DB_VERSION. Size: 396 bytes.
DB_VERSION: 100% completed.

Database was successfully restored
You can now start RDU server.
```

In this example, the database found in the /var/backup/rdu-backup-20070316-031028 directory is restored to the RDU server.

You must restart the RDU after the restore operation is completed. The RDU log file will contain successful startup messages.

This tool reports progress to the user and logs activity in the history.log file.

Changing Database Location

You can move the database from one partition or disk to another on the same system. You might sometimes want to do this for administrative reasons. This process requires stopping the RDU server and the Prime Cable Provisioning process watchdog.

The process of changing the database location involves changing system parameters and copying the appropriate files to the new location.

You can adjust one or both of the following parameters:

- BPR_DATA—This parameter is initially set during installation and points to a directory that is used to store the database, and many other important files, such as logs, configuration files, and so on.

  This directory also stores log data for the Prime Cable Provisioning process watchdog, the DPE (if installed on the same system), the RDU, and SNMP agent, among others.

- BPR_DBLOG—This parameter is initially set during installation and points to the directory that stores database transaction log files.

The values for the above parameters are recorded in a file called BPR_HOME/ bpr_definitions.sh. Any change to this file requires that you restart all Prime Cable Provisioning components running on the system.

To change the location of the database and transaction logs:

Step 1 Run the **/etc/init.d/bprAgent stop** command to stop the Prime Cable Provisioning process watchdog and all Prime Cable Provisioning components.

Step 2 Make a backup copy of the BPR_HOME/bpr_definitions.sh file.

Step 3 Edit the file and change either or both the BPR_DATA and BPR_DBLOG parameters to new directories.

Step 4 Save the file.

Step 5 Copy or move the directory structure and contents of the original BPR_DATA, BPR_DBLOG, or both, directories to the new locations. If you make a copy, make sure that all file and directory permissions are preserved.

Step 6 Run the **/etc/init.d/bprAgent start** command to start the Prime Cable Provisioning process watchdog and all Prime Cable Provisioning components.

Step 7 Monitor the appropriate log files to ensure that all components have successfully started.

RDU Database Migration

For information on migrating the RDU database, see the *Cisco Prime Cable Provisioning 5.0 Quick Start Guide*.

PART 2

Configuring Prime Cable Provisioning

This part contains the following sections:

- Configuring Prime Cable Provisioning Components
- Configuring Technologies in Prime Cable Provisioning
- Configuring Secure Communication
- Configuring IPv6
- Configuring Syslog Utility to Receive Alerts
- Configuring Prime Cable Provisioning Using Admin UI
- Configuring Groups Using Admin UI
- Configuring RBAC Using Admin UI

CHAPTER 6

Configuring Prime Cable Provisioning Components

This chapter describes the workflows that you must follow to configure each Prime Cable Provisioning component for the technologies that Prime Cable Provisioning supports. You must perform these configuration tasks before configuring Prime Cable Provisioning to support specific technologies.

You must configure the Prime Cable Provisioning components in the order specified below.

1. Configuring Regional Distribution Unit, page 6-1
2. Configuring Provisioning Web Services, page 6-6 (required only in case of SOAP based communication)
3. Configuring Device Provisioning Engines, page 6-7
4. Configuring Cisco Prime Network Registrar, page 6-9
5. Configuring Key Distribution Center, page 6-15 (required only in case of Secure eMTA)

Configuring Regional Distribution Unit

This section describes how to configure the Regional Distribution Unit (RDU) component. You must perform these configuration tasks before configuring the other components and technologies.

Table 6-1 identifies the workflow to follow when configuring the RDU.

Table 6-1 RDU Configuration Workflow

	Task	See...
Step 1	Configure the system syslog service for use with Prime Cable Provisioning.	Configuring Syslog Utility to Receive Alerts, page 10-1
Step 2	Access the Prime Cable Provisioning Admin UI.	Administrator User Interface, page 3-36
Step 3	Change the login password.	Accessing Admin UI, page 4-1
Step 4	Add the license file.	Managing Licenses, page 2-1
Step 5	Back up the RDU database.	Backup and Recovery, page 5-4
Step 6	Configure the RDU SNMP agent.	Using snmpAgentCfgUtil.sh Tool, page 22-3

Table 6-1 RDU Configuration Workflow (continued)

	Task	See...
Step 7	Configure the default severity log level, which is the Notification level.	Using the RDU Log Level Tool, page 25-5
Step 8	Enable provisioning-group capabilities for IPv4 or IPv6.	Monitoring Provisioning Groups, page 21-3

Configuring RDU Extensions

Creating a custom extension point is a programming activity that can, when used with the Prime Cable Provisioning Admin UI, allow you to augment Prime Cable Provisioning behavior or add support for new device technologies.

Before familiarizing yourself with managing extensions, you should know the RDU extension points that Prime Cable Provisioning requires. At least one disruption extension must be attached to the associated technology's disruption extension point when disrupting devices on behalf of a batch.

Table 6-2 lists the RDU extension points that Prime Cable Provisioning requires to execute extensions.

Table 6-2 Required RDU Extension Points

Extension Point	Description	Use	Specific to Technology?
Common Configuration Generation	Executed to generate a configuration for a device. Extensions attached to this extension point are executed after the technology-specific service-level selection extension and before the technology-specific configuration generation extensions. The default extensions built into this release do not use this extension point.	Optional	No
Configuration Generation	Executed to generate a configuration for a device.	Required	Yes
Device Detection	Executed to determine a device technology based on information in the DHCP Discover request packet of the device.	Required	No
Disruption	Executed to disrupt a device.	Optional	Yes
Publishing	Executed to publish provisioning data to an external datastore. The default extensions built into Prime Cable Provisioning do not include any publishing plug-ins.	Optional	No

Table 6-2 Required RDU Extension Points (continued)

Extension Point	Description	Use	Specific to Technology?
Service-Level Selection	Executed to select the service level to grant to a device. Extensions attached to this extension point are executed before any common configuration generation extensions and the technology-specific configuration generation extensions.	Optional	Yes
Authentication	Executed to authenticate the user via remote authentication servers. Extensions will be attached to the extension points based on the authentication mode listed in RDU Defaults Page. Radius extensions are default built in authentication extensions in Cisco Prime Cable Provisioning.	Required	Yes

Managing extensions includes:

- Writing a New Class, page 6-3
- Installing RDU Custom Extension Points, page 6-4
- Viewing RDU Extensions, page 6-5
- RDU Extension Dependencies on IPDeviceKeys Properties, page 6-5

Note You can specify multiple extension points by specifying the extension points in a comma-separated list.

Writing a New Class

This procedure is included to better illustrate the entire custom extension creation process. You can create many different types of extensions; for the purposes of this procedure, a new Publishing Extension Point is used.

To write a new class:

Step 1 Create a Java source file for the custom publishing extension, and compile it.

Step 2 Create a manifest file for the JAR file that will contain the extension class.

Note For detailed information on creating a manifest file and using the command-line JAR tool, see Java documentation.

For example:

```
Name: com/cisco/support/extensions/configgeneration
Specification-Title: "DOCSIS TOD synchronization"
Specification-Version: "1.0"
Specification-Vendor: "General Cable, Inc."
Implementation-Title: "Remove the time-servers DHCP option"
Implementation-Version: "1.0"
Implementation-Vendor: "Cisco Systems, Inc."
```

Note Java JAR file manifests contain attributes that are formatted as name-value pairs and support a group of attributes that provide package versioning information. While Prime Cable Provisioning accepts extension JAR files that do not contain this information, we recommend that you include a manifest with versioning information in the files to track custom RDU extensions.

You can view manifest information from the administrator user interface via the **Servers > Regional Distribution Unit > View Regional Distribution Unit Details** page. Detailed information on the installed extension JAR files and the loaded extension class files appears after the Device Statistics section. You can view manifest information from the RDU logs also.

Step 3 Create a JAR file for the custom extension point.

For example:

```
C:\>jar cm0vf manifest.txt removetimeservers.jar com
added manifest
adding: com/(in = 0) (out= 0)(stored 0%)
adding: com/cisco/(in = 0) (out= 0)(stored 0%)
adding: com/cisco/support/(in = 0) (out= 0)(stored 0%)
adding: com/cisco/support/extensions/(in = 0) (out= 0)(stored 0%)
adding: com/cisco/support/extensions/configgeneration/(in = 0) (out= 0)(stored 0%)
adding: com/cisco/support/extensions/configgeneration/
RemoveTimeServersExtension.class(in = 4038) (out= 4038)(stored 0%)
C:\>
```

Note You can give the JAR file any name. The name can be descriptive, but do not duplicate another existing JAR filename.

Installing RDU Custom Extension Points

After a JAR file is created, use the Admin UI to install it:

Step 1 To add the new JAR file, see Adding Files, page 11-15.

Note Select the JAR file type. Use the Browse function to locate the JAR file created in the procedure described in Writing a New Class, page 6-3, and select this file as the Source File. Leaving the File Name blank assigns the same filename for both the source and the file. The filename is what you will see on the administrator user interface.

Step 2 Click **Submit**.

Step 3 Return to the RDU Defaults page and note if the newly added JAR file appears in the Extension Point JAR File Search Order field.

Step 4 Enter the extension class name in the Publishing Extension Point field.

Note The RDU returns an error if the class name does not exist within the JAR file. This error occurs mostly when replacing a JAR file, if, for example, the class you set up is not found in the replacement JAR file.

Step 5 Click **Submit** to commit the changes to the RDU database.

Step 6 View the RDU extensions to ensure that the correct extensions are loaded.

Viewing RDU Extensions

You can view the attributes of all RDU extensions directly from the View Regional Distribution Unit Details page. This page displays details on the installed extension JAR files and the loaded extension class files. See Monitoring RDU, page 21-2.

RDU Extension Dependencies on IPDeviceKeys Properties

The behavior of the RDU built-in extensions varies based on IPDeviceKeys properties that are defined in the Cisco Prime Cable Provisioning properties hierarchy. For example, PacketCable BASIC vs. SECURE mode provisioning is based on the DHCPv4 option-60 capability for supported provisioning flow modes.

Table 6-3 RDU Extension Dependencies on IPDeviceKeys Properties

Property	Scope	RDU Built-in Extension
EXTENDED_FILENAME_SCRIPT	DHCP	ConfigGeneration
DROP_IF_MAX_IA_ADDRESSES_EXCEEDED_ENABLE MAX_IA_ADDRESSES	DHCP	ConfigGeneration
MUST_BE_BEHIND_DEVICE MUST_BE_IN_PROV_GROUP MUST_BE_BEHIND_DEVICE_AUTO_ENABLE	DHCP	ServiceLevelSelection
USE_BOOT_FILE_OPTION_FOR_CONFIG_FILE USE_FILE_OPTION_FOR_CONFIG_FILE	DHCP	ConfigGeneration

Table 6-3 RDU Extension Dependencies on IPDeviceKeys Properties

USE_VALIDATE_CONTINUE_FOR_CABLELABS_SOFTWARE_VERSION_DHCPV4 USE_VALIDATE_CONTINUE_FOR_CABLELABS_SOFTWARE_VERSION_DHCPV6 USE_VALIDATE_CONTINUE_FOR_CLIENT_ID USE_VALIDATE_CONTINUE_FOR_CLIENT_ID_CREATED_FROM_MAC_ADDRESS USE_VALIDATE_CONTINUE_FOR_DHCP_CLASS_IDENTIFIER USE_VALIDATE_CONTINUE_FOR_DHCP_PARAMETER_REQUEST_LIST USE_VALIDATE_CONTINUE_FOR_VENDOR_CLASS USE_VALIDATE_PREFIX_FOR_DHCP_CLASS_IDENTIFIER	DHCP	ConfigGeneration
VALIDATE_CABLELABS_SOFTWARE_VERSION_DHCPV4 VALIDATE_CABLELABS_SOFTWARE_VERSION_DHCPV6 VALIDATE_CLIENT_ID VALIDATE_CLIENT_ID_CREATED_FROM_MAC_ADDRESS VALIDATE_DHCP_CLASS_IDENTIFIER VALIDATE_DHCP_PARAMTER_REQUEST_LIST VALIDATE_VENDOR_CLASS	DHCP	ConfigGeneration

Configuring Provisioning Web Services

The Provisioning Web Services(PWS) component exposes a SOAP based web service interface as an external integration interface. The web service is a layer above the RDU and can be deployed in the same server as the RDU or as a remote server though the recommendation is to deploy it on a remote server.

For each of the PWS service, an internal API request is created and sent to the RDU. The service is capable of interacting with one or more RDUs.

The provisioning service is described using a Web Service Description Language (WSDL) v1.1. The WSDL is a contract between the web service client and the RDU and it describes the PWS operations. You can use the PWS WSDL to generate client language bindings for any language which supports SOAP messages.

Note Shared secret is not supported for the PWS component.

Certain PWS provisioning functions can be carried out using the **ws-cli.sh** tool. For details abut the tool see, Chapter 28, “Using ws-cli.sh”.

Table 6-4 identifies the workflow to follow when configuring the RDU.

Table 6-4 PWS Configuration Workflow

	Task	See...
Step 1	Configure the PWS to connect to the RDU.	Configuring RDU and User Details in PWS, page 6-7
Step 2	To communicate with the client, you can either create a session or use the API.	Configuring Device Provisioning Engines, page 6-7

Configuring RDU and User Details in PWS

To facilitate any provisioning services being requested by a client, the PWS needs to fetch information from the RDU. To communicate with the RDU, the PWS must have a user account in the RDU and to do so, you must configure the RDU details as well as the user credentials in the PWS. These details can be configured in the PWS either while installing PWS or by executing CLI commands that are listed below.

Note The user added to the RDU must have the out of box Admin role.

To configure RDU details in PWS:

Step 1 At the CLI, execute the `-ardu <host> <port> <username> <Password>` command:

For example:

```
./ws-cli.sh -ardu  test1-host  49187  admin  changeme
```

Run the same command to add additional RDUs.

Step 2 Save the entered properties into ws.xml by executing the `-sap, --saveproperty` command:

For example:

```
./ws-cli.sh -sap
```

Step 3 After the account is created, restart the PWS to ensure that the changes take effect.

For example:

```
/etc/init.d/bprAgent restart pws
```

Configuring Device Provisioning Engines

You perform the tasks described in this workflow only after configuring the RDU which is described in Table 6-1. You can configure the DPE to support:

- IPv4. See Table 6-5.
- IPv6. See Table 6-6.

Note Tasks marked with an asterisk (*) are mandatory.

Table 6-5 identifies the workflow to follow when configuring the DPE for IPv4.

Table 6-5 ***DPE Configuration Workflow for IPv4***

	Task	See...
Step 1	Configure the system syslog service for use with Prime Cable Provisioning.	Configuring Syslog Utility to Receive Alerts, page 10-1
Step 2	Change the passwords.	The **password** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
Step 3	Configure the provisioning interface.*	The **interface ip** *ip_address* **provisioning** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
Step 4	Configure the provisioning FQDN.	The **interface ip** *ip_address* **provisioning fqdn** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
Step 5	Configure the interface that communicates with Cisco Prime Network Registrar extensions.	The **interface ip** *ip_address* **pg-communication** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
Step 6	Configure the Prime Cable Provisioning shared secret.*	The **dpe shared-secret** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
Step 7	Configure the DPE to connect to the RDU.*	The **dpe rdu-server port** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
Step 8	Configure the Network Time Protocol (NTP).	Solaris and Linux documentation for configuration information
Step 9	Configure the primary provisioning group.*	The **dpe provisioning-group primary** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
Step 10	Configure the DPE SNMP agent.	The SNMP agent commands in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
	Note You can configure the SNMP agent using either the DPE command-line interface or the **snmpAgentCfgUtil.sh** tool (see Using snmpAgentCfgUtil.sh Tool, page 22-3).	
Step 11	Verify that you are connected to RDU.	Monitoring Servers Using Admin UI, page 21-1
Step 12	Enable provisioning-group capabilities for IPv4.	Monitoring Provisioning Groups, page 21-3

Table 6-6 identifies the workflow to follow when configuring the DPE for IPv6. The tasks that are described here relate to IPv6 alone. To perform basic configuration of the DPE, complete the tasks described in Table 6-5, then additionally complete the steps described in this table.

Table 6-6 DPE Configuration Workflow for IPv6

	Task	See...
Step 1	Configure the provisioning interface.*	The **interface ip** *ipv6_address* **provisioning** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
Step 2	Configure the provisioning FQDN.	The **interface ip** *ipv6_address* **provisioning FQDNs** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
Step 3	Enable TFTP.	The **service tftp** *1..1* **ipv6 enabled true** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
Step 4	Enable ToD.	The **service tod** *1..1* **ipv6 enabled true** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
Step 5	Reload the DPE.	The **dpe reload** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
Step 6	Enable provisioning-group capabilities for v6.	Monitoring Provisioning Groups, page 21-3

Configuring Cisco Prime Network Registrar

You perform the activities described in this workflow only after configuring the RDU and DPE.

Caution The Prime Cable Provisioning DHCP option settings always replace any DHCP option values set within Cisco Prime Network Registrar.

To configure Network Registrar for:

- DHCPv4, see Table 6-7.
- DHCPv6, see Table 6-8.

Note Tasks marked with an asterisk (*) are mandatory.

Table 6-7 identifies the workflow to follow when configuring Network Registrar for DHCPv4.

Table 6-7 Network Registrar Workflow for DHCPv4

	Task	See...
Step 1	Validate the Network Registrar extensions.	*Cisco Prime Cable Provisioning 5.0 Quick Start Guide*
Step 2	Configure the system syslog service for use with Prime Cable Provisioning.	Configuring Syslog Utility to Receive Alerts, page 10-1
Step 3	Configure client classes/selection tags that match those defined in the RDU. *	*Cisco Prime Network Registrar 8.1 User Guide*
Step 4	Configure policies.*	*Cisco Prime Network Registrar 8.1 User Guide*
Step 5	Configure scopes.*	*Cisco Prime Network Registrar 8.1 User Guide*
Step 6	Back up the Network Registrar database.	*Cisco Prime Network Registrar 8.1 User Guide*
Step 7	Verify that you are connected to the correct RDU.	Monitoring Servers Using Admin UI, page 21-1
Step 8	Reload the DHCP server.	*Cisco Prime Network Registrar 8.1 User Guide*

Table 6-8 identifies the workflow to follow when configuring Network Registrar for DHCPv6. Follow this task list for each category of provisioned and unprovisioned devices, including DOCSIS cable modems, computers, and PacketCable MTAs.

Table 6-8 Network Registrar Workflow for DHCPv6

	Task	Refer to...
Step 1	Validate the Network Registrar extensions.	*Cisco Prime Cable Provisioning 5.0 Quick Start Guide*
Step 2	Configure the system syslog service for use with Prime Cable Provisioning.	Configuring Syslog Utility to Receive Alerts, page 10-1
Step 3	Configure client classes/selection tags that match those defined in the RDU. *	*Cisco Prime Network Registrar 8.1 User Guide*
Step 4	Configure policies.*	*Cisco Prime Network Registrar 8.1 User Guide*
Step 5	Configure links.*	*Cisco Prime Network Registrar 8.1 User Guide*
Step 6	Configure prefixes. For each prefix, ensure that you configure the appropriate policy, link, and selection tag.* **Note** Some DHCP clients, such as cable modems, reject Offers that contain multiple IPv6 addresses. While defining prefixes, configure Network Registrar such that it does not assign more than one IPv6 address to a client. Ensure that you do not add the same selection tag to two prefixes, because doing so makes Network Registrar pick one IP address from each prefix, thus assigning two IP addresses to the client.	*Cisco Prime Network Registrar 8.1 User Guide*
Step 7	Back up the Network Registrar database.	*Cisco Prime Network Registrar 8.1 User Guide*

Table 6-8 Network Registrar Workflow for DHCPv6 (continued)

	Task	Refer to...
Step 8	Verify that you are connected to the correct RDU.	Monitoring Servers Using Admin UI, page 21-1
Step 9	Reload the DHCP server.	*Cisco Prime Network Registrar 8.1 User Guide*

Configuring Cisco Prime Network Registrar Extension

This section describes attributes and options as used by Prime Cable Provisioning when communicating with Cisco Prime Network Registrar.

Prime Cable Provisioning uses DHCP extensions installed on Cisco Prime Network Registrar to manipulate DHCP messages based on the configuration in its database. Using these extensions, Prime Cable Provisioning gets information from DHCP Requests and sets the values in the DHCP Responses. In this way, it provides customized configurations for the devices that it provisions.

To facilitate this interaction, Cisco Prime Network Registrar exposes a set of dictionaries to Prime Cable Provisioning extensions. The Prime Cable Provisioning extensions use these dictionaries to interact with Cisco Prime Network Registrar.

There are four types of dictionaries:

- Environment Dictionary—Represents attributes contained in the dictionary that the DHCP server uses to communicate with extensions.
- Request Dictionary—Represents the DHCP options and attributes for a request packet.
- Response Dictionary—Represents the DHCP options and attributes of a response packet.
- Inform Dictionary—Represents information that is communicated between the Prime Cable Provisioning extension and the RDU.

The dictionaries represent various DHCP options and settings as configured on Prime Cable Provisioning and Network Registrar. Options are DHCP configuration parameter and other control information that are stored in the options field of a DHCP message. DHCP clients determine what options are requested and sent in a DHCP packet.

Attributes are name-value pairs and can be:

- DHCPv4 options; for example, **relay-agent-info**.
- A subset of information that is derived from DHCPv4 options; for example, the **relay-agent-remote-id** represents DHCPv4 Option 82 suboption 2.
- Fields from DHCPv4 options; for example, “file” is a DHCPv4 header field.

Attributes can also contain settings, such as:

- Those that control Network Registrar behavior. For example, “drop” to indicate that the packet is to be dropped.
- Those that provide information.

Prime Cable Provisioning, along with Network Registrar 8.x, supports two API versions, each of which Prime Cable Provisioning extensions use to enable DHCPv4 or DHCPv6:

- DEX API version 1—This API allows Network Registrar extensions to query for DHCPv4 packet details via attributes.
- DEX API version 2—This API allows Network Registrar extensions to query for DHCPv4 and DHCPv6 options and suboptions directly.

  If the Prime Cable Provisioning extension discovers the API version of the Network Registrar extension to be DEX API version 2, it enables support for DHCPv6.

Properties that Control Data Discovered for DHCPv6

There are three sets of properties that control the data that Prime Cable Provisioning extensions discover for DHCPv6:

Note From the Admin UI, you can view the settings for these properties on the **Configuration > Defaults > NR Defaults** page.

- Properties that control the behavior of Cisco Prime Network Registrar extensions in versions earlier than Prime Cable Provisioning 5.0, see Table 6-9.
- Properties that control the behavior of Cisco Prime Network Registrar extensions for DHCPv4 in Prime Cable Provisioning 5.0, see Table 6-10.
- Properties that control the behavior of Network Registrar extensions for DHCPv6 in Prime Cable Provisioning for the client (cable modem) and the relay agent (CMTS). This distinction occurs because the DHCPv4 standard combines the client and relay message into one message, while the DHCPv6 standard splits them. See Table 6-10.

Table 6-9 describes the properties that influence the behavior of Cisco Prime Network Registrar extensions in Prime Cable Provisioning versions.

Table 6-9 Properties for Cisco Prime Network Registrar Extensions Before Prime Cable Provisioning 5.0

Property Name	Description
/cnrExtension/attributesRequiredInRequest	Identifies a list of attributes that the Network Registrar request dictionary must contain for extensions to submit a request for configuration generation to the RDU
	API Constant CNR_ATTRIBUTES_REQUIRED_IN_REQUEST_DICTIONARY
/cnrExtension/attributesToPullFromRequestAsBytes	Identifies a list of attributes that must be pulled from the Network Request request dictionary in binary format
	API Constant CNR_ATTRIBUTES_TO_READ_FROM_REQUEST_DICTIONARY_AS_BYTES
/cnrExtension/attributesToPullFromRequestAsStrings	Identifies a list of attributes that must be pulled from the Network Registrar request dictionary in string format
	API Constant CNR_ATTRIBUTES_TO_READ_FROM_REQUEST_DICTIONARY_AS_STRINGS

Table 6-9 Properties for Cisco Prime Network Registrar Extensions Before Prime Cable Provisioning 5.0 (continued)

Property Name	Description
/cnrExtension/attributesToReadFromEnvironmentDictionary	Identifies a list of attributes that must be pulled from the Network Registrar environment dictionary
	API Constant
	`CNR_ATTRIBUTES_TO_READ_FROM_ENVIRONMENT_DICTIONARY`
/cnrExtension/attributesRequiredInV4Request	Identifies a list of attributes that the Network Registrar request dictionary must contain for extensions to submit a request for configuration generation to the RDU
	API Constant
	`CNR_ATTRIBUTES_REQUIRED_IN_V4_REQUEST_DICTIONARY`
/cnrExtension/attributesToPullFromV4RequestAsBytes	Identifies a list of attributes to be pulled from the Network Registrar request dictionary in binary format
	API Constant
	`CNR_ATTRIBUTES_TO_READ_FROM_V4_REQUEST_DICTIONARY_AS_BYTES`
/cnrExtension/attributesToPullFromV4RequestAsStrings	Identifies a list of attributes to be pulled from the Network Registrar request dictionary in string format
	API Constant
	`CNR_ATTRIBUTES_TO_READ_FROM_V4_REQUEST_DICTIONARY_AS_STRINGS`

Table 6-10 describes the properties that control the behavior of Prime Network Registrar extensions for DHCPv6 in Prime Cable Provisioning.

Table 6-10 Properties for DHCPv6 Cisco Prime Network Registrar Extensions in Prime Cable Provisioning 5.0

Property Name	Description
Client Message	
/cnrExtension/attributesRequiredInV6Request	Identifies a list of attributes that the Network Registrar DHCPv6 request dictionary must contain for extensions to submit a request for configuration generation to the RDU
	API Constant
	`CNR_ATTRIBUTES_REQUIRED_IN_V6_REQUEST_DICTIONARY`
/cnrExtension/attributesToPullFromV6RequestAsBytes	Identifies a list of attributes to be pulled from the Network Registrar DHCPv6 request dictionary in binary format
	API Constant
	`CNR_ATTRIBUTES_TO_READ_FROM_V6_REQUEST_DICTIONARY_AS_BYTES`

Table 6-10 Properties for DHCPv6 Cisco Prime Network Registrar Extensions in Prime Cable Provisioning 5.0 (continued)

Property Name	Description
/cnrExtension/optionsRequiredIn V6Request	Identifies a list of DHCP options that the Network Registrar DHCPv6 request dictionary must contain for extensions to submit a request for configuration generation to the RDU
	API Constant `CNR_OPTIONS_REQUIRED_IN_V6_REQUEST_DICTIONARY`
/cnrExtension/optionsToPullFrom V6Request AsBytes	Identifies a list of DHCP options to be pulled from the Network Registrar DHCPv6 request dictionary in binary format
	API Constant `CNR_OPTIONS_TO_READ_FROM_V6_REQUEST_DICTIONARY_AS_ BYTES`
Relay Message	
/cnrExtension/attributesRequiredIn V6Relay	Identifies a list of attributes that the Network Registrar DHCPv6 Relay-Forward request dictionary must contain for extensions to submit a request for configuration generation to the RDU
	API Constant `CNR_ATTRIBUTES_REQUIRED_IN_V6_RELAY_DICTIONARY`
/cnrExtension/attributesToPullFrom V6RelayAsBytes	Identifies a list of attributes to be pulled from the Network Registrar DHCPv6 Relay-Forward request relay dictionary in binary format
	API Constant `CNR_ATTRIBUTES_TO_READ_FROM_V6_RELAY_DICTIONARY_AS_ BYTES`
/cnrExtension/optionsRequiredIn V6Relay	Identifies a list of DHCP options that the Network Registrar DHCPv6 Relay-Forward request dictionary must contain for extensions to submit a request for configuration generation to the RDU
	API Constant `CNR_OPTIONS_REQUIRED_IN_V6_RELAY_DICTIONARY`
/cnrExtension/optionsToPullFrom V6RelayAsBytes	Identifies a list of DHCP options to be pulled from the Network Registrar DHCPv6 Relay-Forward request Relay dictionary in binary format
	API Constant `CNR_OPTIONS_TO_READ_FROM_V6_RELAY_DICTIONARY_AS_BYTES`

Configuring Key Distribution Center

PacketCable Secure depends on the Kerberos infrastructure to mutually authenticate the MTA and the provisioning system; in Prime Cable Provisioning, the KDC functions as the Kerberos server. For an overview of the KDC component, see Key Distribution Center, page 3-35.

Default KDC Properties

The KDC has several default properties that are populated during a Prime Cable Provisioning installation into the BPR_HOME/kdc/<Linux/Solaris>/kdc.ini properties file. You can edit this file to change values as operational requirements dictate.

Note Be careful in editing the kdc.ini file if operational requirements dictate. Incorrect values can render the KDC inoperative. If you do make changes, restart the KDC.

The default properties are:

- interface address—Specifies the IP address of the local Ethernet interface that you want the KDC to monitor for incoming Kerberos messages.

  For example:

  ```
  interface address = 10.10.10.1
  ```

- FQDN—Identifies the fully qualified domain name (FQDN) on which the KDC is installed.

  For example:

  ```
  FQDN = kdc.example.com
  ```

  Note You must enter the interface address and FQDN values through the KDC Realm Name screen during installation. For specific information, see the *Cisco Prime Cable Provisioning 5.0 Quick Start Guide*.

- maximum log file size—Specifies the maximum size, in kilobytes, that the log file that is generated by the KDC can reach. The KDC creates a new log file only when the current file reaches this maximum size.

  For example:

  ```
  maximum log file size = 1000
  ```

- *n* saved log files—Defines the number of old log files that the KDC saves. The default value is 7. You can specify as many as required.

  For example:

  ```
  n saved log files = 10
  ```

- log debug level—Specifies the logging level for the log file.

  ```
  log debug level = 5
  ```

Table 6-11 describes the available logging levels for the KDC log file.

Table 6-11 KDC Logging Levels

Log Level	Description
0	Error conditions exist. Sets the logging function to save all error messages and those of a more severe nature.
1	Warning conditions exist. Sets the logging function to save all warning messages and those of a more severe nature.
2	Informational messages. Sets the logging function to save all logging messages available.
{*3-7*}	Debugging messages. Sets the logging function to save all debugging messages at various levels, from level 3 to level 7.

- minimum (maximum) ps backoff—Specifies the minimum (or maximum) time, in tenths of a second, that the KDC waits for Prime Cable Provisioning to respond to the FQDN-Request.

  For example:

  ```
  minimum ps backoff = 150
  ```

Using the sample values shown above, a sample INI file might contain data similar to that shown in Example 6-1.

Example 6-1 Sample kdc.ini Configuration File

```
interface address =  10.10.10.1
FQDN = kdc.example.com
maximum log file size = 1000
n saved log files = 10
log debug level = 5
minimum ps backoff = 150
maximum ps backoff = 300
```

You can set the times for both minimum and maximum ticket duration to effectively smooth out excessive numbers of ticket requests that could occur during deployment. This setting is beneficial given that most deployments occur during traditional working hours and excessive loading might, from time to time, adversely affect performance.

Note Shortening the ticket duration forces the MTA to authenticate to the KDC much more frequently. While this results in greater control over the authorization of telephony endpoints, it also causes heavier message loads on the KDC and increased network traffic. In most situations, the default setting is appropriate and should not be changed.

- maximum ticket duration—Defines the maximum duration for tickets generated by the KDC. The default unit is hours; however, by appending an **m** or **d**, you can change the units to minutes or days, respectively.

  The default value is 168, or seven days. We recommend that you not change this value because this value is the length of time required to conform to the PacketCable security specification.

  For example:

  ```
  maximum ticket duration = 168
  ```

- minimum ticket duration—Defines the minimum duration for tickets generated by the KDC. The default unit is hours; however, by appending an **m** or **d**, you can change the units to minutes or days, respectively.

  The default value is 144, or six days. We recommend that you not change this value.

  For example:

  ```
  minimum ticket duration = 144
  ```

KDC Certificates

The certificates used to authenticate the KDC are not shipped with Prime Cable Provisioning. You must obtain the required certificates from Cable Television Laboratories, Inc. (CableLabs), and the content of these certificates must match the content in the certificates installed in the MTA.

Note Certificates are required for the KDC to function.

You can use the PKCert tool to install, and manage, the certificates that the KDC requires for its operation. The PKCert tool installs the CableLabs service provider certificates as certificate files. For information on running this tool, see Using PKCert.sh, page 28-3.

The PKCert tool is available only if you have installed the KDC component.

KDC Licenses

Obtain a KDC license from your Cisco representative and then install it in the correct directory.

To install a KDC license file:

Step 1 Obtain your license file from your Cisco representative.

Step 2 Log into the Prime Cable Provisioning host as root.

Step 3 Copy the license file to the BPR_HOME/kdc directory.

Caution Be careful not to copy the file as an ASCII file. The file contains binary data susceptible to unwanted modification during an ASCII transfer.

Do not copy KDC license files between operating systems because the transfer process may damage the file.

Step 4 To restart the KDC server and make the changes take effect, run the **bprAgent restart kdc** command from the /etc/init.d directory.

Multiple Realm Support

The Prime Cable Provisioning KDC supports the management of multiple realms, for which a complete set of valid PacketCable X.509 certificates and a KDC private key must be present. These certificates must reside in the BPR_HOME/kdc/<Operating System>/packetcable/certificates directory.

Prime Cable Provisioning supports additional realms by installing subdirectories under the BPR_HOME/kdc/<Operating System>/packetcable/certificates directory; each subdirectory is named after a specific realm.

Table 6-12 lists the different certificates, with their corresponding filenames, that must be available in the BPR_HOME/kdc/<Operating System>/packetcable/certificates directory.

Table 6-12 PacketCable Certificates

Certificate	Certificate Filename
MTA Root	MTA_Root.cer
Service Provider Root	CableLabs_Service_Provider_Root.cer
Service Provider CA	Service_Provider.cer
Local System Operator CA	Local_System.cer
KDC	KDC.cer

The primary realm is set up during installation of the KDC component. For the primary realm, the KDC certificate (KDC.cer) resides in the BPR_HOME/kdc/<Operating System>/packetcable/certificates directory. Its private key (KDC_private_key.pkcs8) resides in the BPR_HOME/kdc/<Operating System>/ directory.

To configure additional realms, follow this procedure, which is described in detail subsequently.

Step 1 Locate the directory containing your KDC certificates.

Step 2 Create a subdirectory under the directory that stores the KDC certificates.

Note Match the name of the subdirectory with the name of the specific realm. Use only uppercase characters while naming the subdirectory.

Step 3 Place the KDC certificate and the private key for the realm in the subdirectory you created.

Step 4 If the new realm is not chained to the same service provider as the KDC certificate, include all additional higher-level certificates that differ from those in the certificates directory.

Note Because all realms must be rooted in the same certificate chain, a KDC installation supports only one locale (North American PacketCable or Euro PacketCable) at any given point.

Table 6-13 describes the directory structure and files for a primary realm (for example, CISCO.COM) with two secondary realms (for example, CISCO1.COM and CISCO2.COM). The structure assumes that the higher-level certificates are similar for the primary realm and its secondary realms.

Table 6-13 Directory Structure for Multiple Realms

Directory	File Content in Directory
BPR_HOME/kdc/solaris	For primary realm CISCO.COM: KDC private key
BPR_HOME/kdc/<Operating System>/packetcable/certificates	For primary realm CISCO.COM: • MTA_Root.cer • CableLabs_Service_Provider_Root.cer • Service_Provider.cer • Local_System.cer • KDC.cer Directory /CISCO1.COM Directory /CISCO2.COM
BPR_HOME/kdc/<Operating System>/packetcable/certificates/CISCO1.COM	For secondary realm CISCO1.COM: • KDC.cer • KDC private key
BPR_HOME/kdc/<Operating System>/packetcable/certificates/CISCO2.COM	For secondary realm CISCO2.COM: • KDC.cer • KDC private key

Configuring the KDC for Multiple Realms

This section describes the workflow to configure the KDC for multiple realms. Before proceeding, complete the installation of the RDU, the DPE, and the Network Registrar extensions. For installation instructions, see the *Cisco Prime Cable Provisioning 5.0 Quick Start Guide*.

The following workflow uses sample realms and directories to describe how to configure the KDC for multiple realms. The primary realm used here is CISCO.COM and its secondary realms are CISCO1.COM and CISCO2.COM.

The setup featured in the following workflow provisions three MTAs: a Motorola SBV 5120 MTA, a Linksys CM2P2 MTA, and an SA WebStar DPX 2203 MTA. Each MTA is to be provisioned in one realm: the Motorola in the CISCO.COM realm, the Linksys MTA in the CISCO1.COM realm, and the SA MTA in the CISCO2.COM realm.

Note The sample output shown in the following procedure has been trimmed for demonstration purposes.

To configure the KDC for multiple realms:

Step 1 Verify the following configuration settings on the DPE:

a. Ensure that PacketCable services are enabled, by using the **show run** command.

To enable the PacketCable service, use the **service packetcable** *1..1* **enable** command.

For example:

```
dpe# show run
  aaa authentication local
  dpe port 49186
  dpe provisioning-group primary default
  service packetcable 1 enable
  snmp-server location equipmentrack5D
  snmp-server udp-port 8001
  tacacs-server retries 2
  tacacs-server timeout 5
```

For details on the commands, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.

b. Ensure that the security used for communication between the KDC and a DPE is set, by using the **show run** command.

To generate and set the security key, use the **service packetcable** *1..1* **registration kdc-service-key** command.

For example:

```
dpe# show run
  aaa authentication local
  debug dpe events
  dpe port 49186
  service packetcable 1 enable
  service packetcable 1 registration kdc-service-key <value is set>
  snmp-server contact AceDuffy-ext1234
```

For details on the commands, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.

c. Ensure that the security key that permits secure communication between the DPE and the RDU for PacketCable SNMPv3 cloning is set. Again, use the **show run** command. To generate and set the security key, use the **service packetcable** *1..1* **snmp key-material** command.

For example:

```
dpe# show run
  aaa authentication local
  debug dpe events
  dpe port 49186
  service packetcable 1 enable
  service packetcable 1 registration kdc-service-key <value is set>
  service packetcable 1 snmp key-material <value is set>
```

For details on the commands, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.

Note When you configure PacketCable settings on the DPE, ensure that you run the **dpe reload** command so that the changes take effect.

Step 2 In the configuration file for Network Registrar extension points (cnr_ep.properties), verify if the /ccc/kerb/realm parameter is set to the primary realm; in this case, CISCO.COM. To do this, run the more cnr_ep.properties command from the BPR_HOME/cnr_ep/conf directory.

For example:

```
/opt/CSCObac/cnr_ep/conf# more cnr_ep.properties
#DO NOT MODIFY THIS FILE.
#This file was created on Wed, March 4 06:34:34 EDT 2007
/rdu/port=49187
/rdu/fqdn=dpe4.cisco.com
/cache/provGroupList=Default
/cnr/sharedSecret=fggTaLg0XwKRs
/pktcbl/enable=enabled
/ccc/tgt=01
/ccc/kerb/realm=CISCO.COM
/ccc/dhcp/primary=10.10.0.1
/ccc/dns/primary=10.10.0.1
```

Step 3 Enable static routes appropriately to ensure Prime Cable Provisioning connectivity with devices behind the CMTS.

Step 4 Create DNS realm zones for the DNS server that is listed in the cnr_ep.properties file. You can add zones using the Network Registrar administrator user interface via the DNS > Forward Zones > List/Add Zones pages.

Note Ensure that the zones you add contain the SRV record and the DNS ‘A’ record for the KDC server, and that the SRV record for each zone (in this example, CISCO.COM, CISCO1.COM, and CISCO2.COM) point to one KDC.

For information on configuring zones from the administrator user interface, see the *Cisco Prime Network Registrar 8.1 User Guide*.

Step 5 Configure certificates using the PKCert.sh tool.

a. Create directories for the secondary realms (for example, CISCO1.COM and CISCO2.COM) under BPR_HOME/kdc/<Operating System>/packetcable/certificates.

For example:

```
/opt/CSCObac/kdc/<Operating System>/packetcable/certificates# mkdir CISCO1.COM
/opt/CSCObac/kdc/<Operating System>/packetcable/certificates# mkdir CISCO2.COM
```

For more information on creating directories, see Solaris documentation.

b. Create a directory in which you can copy the following certificates:

- CableLabs_Service_Provider_Root.cer
- Service_Provider.cer
- Local_System.cer
- MTA_Root.cer
- Local_System.der

For example:

```
# cd /var
# mkdir certsInput
```

Note The /certsInput directory created under the /var directory is only an example. You can choose to create any directory under any other directory. For more information on creating directories, see the specific Operating System documentation.

c. Copy the certificates mentioned in the previous step into the directory that you created.

d. Copy the following certificates to the BPR_HOME/kdc/<Operating System>/packetcable/certificates directory:

- CableLabs_Service_Provider_Root.cer
- Service_Provider.cer
- Local_System.cer
- MTA_Root.cer

For information on copying files, see Solaris documentation on the **cp** command.

e. Create the KDC certificate and its associated private key for the primary realm.

For example:

```
# ./opt/CSCObac/kdc/PKCert.sh -c "-s /var/certsInput -d /var/certsOutput
-k /var/certsInput/Local_System.der -c /var/certsInput/Local_System.cer
-r CISCO.COM -n 100 -a bactest.cisco.com -o"
Pkcert Version 1.0
Logging to pkcert.log
Source Directory: /var/certsInput
Destination Directory: /var/certsOutput
Private Key File: /var/certsInput/Local_System.der
Certificate File: /var/certsInput/Local_System.cer
Realm: CISCO.COM
Serial Number: 100
DNS Name of KDC: bactest.cisco.com
WARNING - Certificate File will be overwritten
SP Cert subject name: C=US,O=CableLabs\, Inc.,OU=ABC Cable Company,CN=Shared-01
CableLabs Local System CA
File written: /var/certsOutput/KDC_private_key.pkcs8
File written: /var/certsOutput/KDC_private_key_proprietary.
File written: /var/certsOutput/KDC_PublicKey.der
File written: /var/certsOutput/KDC.cer
KDC Certificate Successfully Created at /var/certsOutput/KDC.cer

Copy KDC.cer to the KDC certificate directory (i.e. /opt/CSCObac/kdc/<Operating
System>/
packetcable/certificates)
Copy KDC_private_key.pkcs8 to the KDC platform directory (i.e. /opt/CSCObac/
kdc/solaris)
Copy KDC_private_key_proprietary. to the KDC platform directory (i.e. /opt/CSCObac/
kdc/solaris)
```

For more information on the tool, see Using PKCert.sh, page 28-3.

f. Copy the KDC.cer file to the KDC certificate directory (BPR_HOME/kdc/<Operating System>/packetcable/certificates). For information on copying files, see Solaris documentation on the cp command.

g. Copy the private key KDC_private_key.pkcs8 to the KDC platform directory (BPR_HOME/kdc/solaris). For information on copying files, see Solaris documentation on the cp command.

h. Copy the private key KDC_private_key_proprietary. to the KDC platform directory (BPR_HOME/kdc/solaris). For information on copying files, see Solaris documentation on the **cp** command.

i. Create the KDC certificate and its associated private key for the secondary realm; in this case, CISCO1.COM.

For example:

```
# ./opt/CSCObac/kdc/PKCert.sh -c "-s /var/certsInput -d /var/certsOutput
-k /var/certsInput/Local_System.der -c /var/certsInput/Local_System.cer
-r CISCO1.COM -n 100 -a bactest.cisco.com -o"
Pkcert Version 1.0
Logging to pkcert.log
Source Directory: /var/certsInput
Destination Directory: /var/certsOutput
Private Key File: /var/certsInput/Local_System.der
Certificate File: /var/certsInput/Local_System.cer
Realm: CISCO.COM
Serial Number: 100
DNS Name of KDC: bactest.cisco.com
WARNING - Certificate File will be overwritten
SP Cert subject name: C=US,O=CableLabs\, Inc.,OU=ABC Cable Company,CN=Shared-01
CableLabs Local System CA
File written: /var/certsOutput/KDC_private_key.pkcs8
File written: /var/certsOutput/KDC_private_key_proprietary.
File written: /var/certsOutput/KDC_PublicKey.der
File written: /var/certsOutput/KDC.cer
KDC Certificate Successfully Created at /var/certsOutput/KDC.cer

Copy KDC.cer to the KDC certificate directory (i.e. /opt/CSCObac/kdc/<Operating
System>/
packetcable/certificates)
Copy KDC_private_key.pkcs8 to the KDC platform directory (i.e. /opt/CSCObac/
kdc/solaris)
Copy KDC_private_key_proprietary. to the KDC platform directory (i.e. /opt/CSCObac/
kdc/solaris)
```

For more information on the tool, see Using PKCert.sh, page 28-3.

j. Copy *KDC.cer* to the secondary realm directory; for example, the /CISCO1.COM directory under BPR_HOME/kdc/<Operating System>/packetcable/certificates. For information on copying files, see Solaris documentation on the cp command.

k. Copy the private key KDC_private_key.pkcs8 to the secondary realm directory; for example, the /CISCO1.COM directory under BPR_HOME/kdc/<Operating System>/packetcable/certificates. For information on copying files, see Solaris documentation on the cp command.

l. Copy the private key KDC_private_key_proprietary. to the secondary realm directory; for example, the /CISCO1.COM directory under BPR_HOME/kdc/<Operating System>/packetcable/certificates. For information on copying files, see Solaris documentation on the **cp** command.

m. Create the KDC certificate and its associated private key for the secondary CISCO2.COM realm.

For example:

```
# ./opt/CSCObac/kdc/PKCert.sh -c "-s /var/certsInput -d /var/certsOutput
-k /var/certsInput/Local_System.der -c /var/certsInput/Local_System.cer
-r CISCO2.COM -n 100 -a bactest.cisco.com -o"
Pkcert Version 1.0
Logging to pkcert.log
Source Directory: /var/certsInput
Destination Directory: /var/certsOutput
Private Key File: /var/certsInput/Local_System.der
Certificate File: /var/certsInput/Local_System.cer
Realm: CISCO.COM
Serial Number: 100
DNS Name of KDC: bactest.cisco.com
WARNING - Certificate File will be overwritten
SP Cert subject name: C=US,O=CableLabs\, Inc.,OU=ABC Cable Company,CN=Shared-01
CableLabs Local System CA
File written: /var/certsOutput/KDC_private_key.pkcs8
File written: /var/certsOutput/KDC_private_key_proprietary.
File written: /var/certsOutput/KDC_PublicKey.der
File written: /var/certsOutput/KDC.cer
KDC Certificate Successfully Created at /var/certsOutput/KDC.cer

Copy KDC.cer to the KDC certificate directory (i.e. /opt/CSCObac/kdc/<Operating
System>/
packetcable/certificates)
Copy KDC_private_key.pkcs8 to the KDC platform directory (i.e. /opt/CSCObac/
kdc/solaris)
Copy KDC_private_key_proprietary. to the KDC platform directory (i.e. /opt/CSCObac/
kdc/solaris)
```

For information on the tool, see Using PKCert.sh, page 28-3.

n. Copy KDC.cer to the secondary realm directory; for example, the /CISCO2.COM directory under BPR_HOME/kdc/<Operating System>/packetcable/certificates. For information on copying files, see Solaris documentation on the cp command.

o. Copy the private key KDC_private_key.pkcs8 to the secondary realm directory; for example, the /CISCO2.COM directory under BPR_HOME/kdc/<Operating System>/packetcable/certificates. For information on copying files, see Solaris documentation on the cp command.

p. Copy the private key KDC_private_key_proprietary. to the secondary realm directory; for example, the /CISCO2.COM directory under BPR_HOME/kdc/<Operating System>/packetcable/certificates. For information on copying files, see Solaris or Linux documentation on the **cp** command.

Step 6 Generate PacketCable service keys by using the KeyGen tool.

Note Ensure that the password that you use to generate a service key matches the password that you set on the DPE by using the **packetcable registration kdc service-key** command.

For example:

```
# /opt/CSCObac/kdc/keygen bactest.cisco.com CISCO.COM changeme
# /opt/CSCObac/kdc/keygen bactest.cisco.com CISCO1.COM changeme
# /opt/CSCObac/kdc/keygen bactest.cisco.com CISCO2.COM changeme
```

For details, see Using PKCert.sh, page 28-3.

Step 7 Ensure that the service keys you generated in Step 6, exist in the BPR_HOME/kdc/<Operating System>/keys directory.

For example:

```
/opt/CSCObac/kdc/<Operating System>/keys# ls -l
total 18
-rw-r--r--   1 root     other 2 Nov  4 09:44 krbtgt,CISCO1.COM@CISCO1.COM
-rw-r--r--   1 root     other 2 Nov  4 09:44 krbtgt,CISCO2.COM@CISCO2.COM
-rw-r--r--   1 root     other 2 Nov  4 09:44 krbtgt,CISCO.COM@CISCO.COM
-rw-r--r--   1 root     other 2 Nov  4 09:44 mtafqdnmap,bactest.cisco.com@CISCO1.COM
-rw-r--r--   1 root     other 2 Nov  4 09:44 mtafqdnmap,bactest.cisco.com@CISCO2.COM
-rw-r--r--   1 root     other 2 Nov  4 09:44 mtafqdnmap,bactest.cisco.com@CISCO.COM
-rw-r--r--   1 root     other 2 Nov  4 09:44 mtaprovsrvr,bactest.cisco.com@CISCO1.COM
-rw-r--r--   1 root     other 2 Nov  4 09:44 mtaprovsrvr,bactest.cisco.com@CISCO2.COM
-rw-r--r--   1 root     other 2 Nov  4 09:44 mtaprovsrvr,bactest.cisco.com@CISCO.COM
```

For more information, see Solaris documentation.

Step 8 Ensure that the various certificates and service keys exist in the BPR_HOME/kdc directory.

For example:

```
/opt/CSCObac/kdc# ls
PKCert.sh   internal keygen lib pkcert.log  solaris bacckdc.license

/opt/CSCObac/kdc# cd /internal/bin
/internal/bin# ls
kdc runKDC.sh shutdownKDC.sh

# cd /opt/CSCObac/kdc/lib
# ls
libgcc_s.so.1      libstdc++.so.5     libstlport_gcc.so

# cd /opt/CSCObac/<Operating System>/logs
# ls
kdc.log    kdc.log.1

# cd /opt/CSCObac/solaris
# ls
logs kdc.ini packetcable KDC_private_key_proprietary.

# cd keys
# ls
krbtgt,CISCO1.COM@CISCO1.COM
krbtgt,CISCO2.COM@CISCO2.COM
krbtgt,CISCO.COM@CISCO.COM
mtafqdnmap,bactest.cisco.com@CISCO1.COM
mtafqdnmap,bactest.cisco.com@CISCO2.COM
mtafqdnmap,bactest.cisco.com@CISCO.COM
mtaprovsrvr,bactest.cisco.com@CISCO1.COM
mtaprovsrvr,bactest.cisco.com@CISCO2.COM
mtaprovsrvr,bactest.cisco.com@CISCO.COM

# cd ./<Operating System>/packetcable/certificates
# ls
KDC.cer
Local_System.cer
CableLabs_Service_Provider_Root.cer  MTA_Root.cer
CISCO1.COM                         Service_Provider.cer
CISCO2.COM

# cd ./<Operating System>/packetcable/certificates/CISCO1.COM
# ls
KDC.cer
```

```
KDC_private_key_proprietary.

# cd ./<Operating System>/packetcable/certificates/CISCO2.COM:
# ls
KDC.cer
KDC_private_key_proprietary.
```

For more information, see Solaris/Linux documentation.

Step 9 Restart the KDC.

For example:

```
# /etc/init.d/bprAgent restart kdc
```

For more information, see Using Prime Cable Provisioning Process Watchdog from CLI, page 23-2.

Step 10 Configure the Prime Cable Provisioning administrator user interface for multiple realms.

a. Add DHCP Criteria for the secondary realm; in this case, CISCO1.COM.

For example:

1. From **Configuration > DHCP Criteria > Manage DHCP Criteria**, click the **Add** button.
2. The Add DHCP Criteria page appears.
3. Enter **cisco1** in the DHCP Name field.
4. Click **Submit**.
5. Return to the Manage DHCP Criteria page, and click the cisco1 DHCP criteria. The Modify DHCP Criteria page appears.
6. Under Property Name, select /ccc/kerb/realm and enter CISCO1.COM in the Property Value field.
7. Click **Add** and **Submit**.

For more information, see Configuring DHCP Criteria, page 11-12.

b. Add DHCP Criteria for the secondary realm; in this case, CISCO2.COM.

For example:

1. From **Configuration > DHCP Criteria > Manage DHCP Criteria**, click the **Add** button.
2. The Add DHCP Criteria page appears.
3. Enter **cisco2** in the DHCP Name field.
4. Click **Submit**.
5. Return to the Manage DHCP Criteria page, and click the cisco2 DHCP criteria. The Modify DHCP Criteria page appears.
6. Under Property Name, select /ccc/kerb/realm and enter cisco2.COM in the Property Value field.
7. Click **Add** and **Submit**.

For more information, see Configuring DHCP Criteria, page 11-12.

c. Add templates as files to Prime Cable Provisioning for each of the devices being provisioned; in this step, for the Motorola MTA.

For example:

1. Choose **Configuration > Files**. The Manage Files page appears.
2. Click **Add**, and the Add Files page appears.
3. Select the CableLabs Configuration Template option from the File Type drop-down list.

4. Add the mot-mta.tmpl file. This file is the template used to provision a Motorola MTA. For template syntax, see Example 6-2 on page 6-32.
5. Click **Submit**.

For more information, see Managing Files, page 11-14.

d. Add templates as files to Prime Cable Provisioning for each of the devices being provisioned; in this step, for the Linksys MTA.

For example:

1. Choose **Configuration > Files**. The Manage Files page appears.
2. Click **Add**, and the Add Files page appears.
3. Select the CableLabs Configuration Template option from the File Type drop-down list.
4. Add the linksys-mta.tmpl file. This file is the template used to provision a Linksys MTA. For template syntax, see Example 6-3 on page 6-33.
5. Click **Submit**.

For more information, see Managing Files, page 11-14.

e. Add templates as files to Prime Cable Provisioning for each of the devices being provisioned; in this step, for the SA MTA.

For example:

1. Choose **Configuration > Files**. The Manage Files page appears.
2. Click **Add**, and the Add Files page appears.
3. Select the CableLabs Configuration Template option from the File Type drop-down list.
4. Add the sa-mta.tmpl file. This file is the template used to provision an SA MTA. For template syntax, see Example 6-4 on page 6-34.
5. Click **Submit**.

For more information, see Managing Files, page 11-14.

f. Add a Class of Service for the primary realm; in this case, CISCO.COM.

For example:

1. Choose **Configuration > Class of Service**.
2. Click **Add**. The Add Class of Service page appears.
3. Enter mot-mta as the name of the new Class of Service for the CISCO.COM realm.
4. Choose the Class of Service Type as PacketCableMTA.
5. Select /cos/packetCableMTA/file from the Property Name drop-down list and associate it to the mot-mta.tmpl template file (which is used to provision the Motorola MTA in the primary CISCO.COM realm).
6. Click **Add** and **Submit**.

For more information, see Configuring Class of Service, page 11-1.

g. Add a Class of Service for the secondary realm; in this case, CISCO1.COM.

For example:

1. Choose **Configuration > Class of Service**.
2. Click **Add**. The Add Class of Service page appears.
3. Enter linksys-mta as the name of the new Class of Service for the CISCO1.COM realm.

4. Choose the Class of Service Type as PacketCableMTA.
5. Select /cos/packetCableMTA/file from the Property Name drop-down list and associate it to the linksys-mta.tmpl template file (which is used to provision the Linksys MTA in the secondary CISCO1.COM realm).
6. Click **Add** and **Submit**.

For more information, see Configuring Class of Service, page 11-1.

h. Add a Class of Service for the secondary realm; in this case, CISCO2.COM.

For example:

1. Choose **Configuration > Class of Service**.
2. Click **Add**. The Add Class of Service page appears.
3. Enter sa-mta as the name of the new Class of Service for the CISCO1.COM realm.
4. Choose the Class of Service Type as PacketCableMTA.
5. Select /cos/packetCableMTA/file from the Property Name drop-down list and associate it to the sa-mta.tmpl template file (which is used to provision the SA MTA in the secondary CISCO2.COM realm).
6. Click **Add** and **Submit**.

For more information, see Configuring Class of Service, page 11-1.

Step 11 Bring the devices online and provision them. See the following examples that describe the provisioning process.

Example 1

The following example describes how you can provision the Motorola SBV5120.

a. Provision the cable modem part of the device by setting it to use the **sample-bronze-docsis** Class of Service.

b. To provision the MTA part, go to the **Devices > Manage Devices** page. Search and select the PacketCable device you want to provision. The Modify Device page appears.

c. Set the domain name. This example uses bacclab.cisco.com.

d. From the drop-down list corresponding to Registered Class of Service, select **mot-mta**. This is the Class of Service that you added in Step 10-f.

e. From the drop-down list corresponding to Registered DHCP Criteria, select the **default** option.

f. Click **Submit**.

Example 2

The following example illustrates how you can provision the Linksys CM2P2.

a. Provision the cable modem part of the device by setting it to use the **sample-bronze-docsis** Class of Service.

b. To provision the MTA part, go to the **Devices > Manage Devices** page. Search and select the PacketCable device you want to provision. The Modify Device page appears.

c. Set the domain name. This example uses bacclab.cisco.com.

d. From the drop-down list corresponding to Registered Class of Service, select **linksys-mta**. This is the Class of Service that you added in Step 10-g.

e. From the drop-down list corresponding to Registered DHCP Criteria, select the **cisco1** option. This is the DHCP Criteria that you added for the secondary CISCO1.COM realm in Step 10-a.

f. Click **Submit**.

Example 3

The following example illustrates how you can provision the SA WebStar DPX 2203.

a. Provision the cable modem part of the device by setting it to use the **sample-bronze-docsis** Class of Service.

b. To provision the MTA part, go to the **Devices > Manage Devices** page. Search and select the PacketCable device you want to provision. The Modify Device page appears.

c. Set the domain name. This example uses bacclab.cisco.com.

d. From the drop-down list corresponding to Registered Class of Service, select **sa-mta**. This is the Class of Service that you added in Step 10-h.

e. From the drop-down list corresponding to Registered DHCP Criteria, select the **cisco2** option. This is the DHCP Criteria that you added for the secondary CISCO2.COM realm in Step 10-b.

f. Click **Submit**.

Step 12 Verify if multiple realm support is operational by using an ethereal trace. See the sample output from the KDC and DPE log files shown here from the sample setup used in this procedure.

Example 1

The following example features excerpts from the KDC and DPE log files for the Motorola SBV 5120 MTA provisioned in the primary CISCO.COM realm:

KDC Log Sample Output–Motorola MTA

```
INFO [Thread-4] 2007-02-07 07:56:21,133 (DHHelper.java:114) - Time to create DH key
pair(ms): 48
 INFO [Thread-4] 2007-02-07 07:56:21,229 (DHHelper.java:114) - Time to create DH key
pair(ms): 49
 INFO [Thread-4] 2007-02-07 07:56:21,287 (DHHelper.java:150) - Time to create shared
secret: 57 ms.
 INFO [Thread-4] 2007-02-07 07:56:21,289 (PKAsReqMsg.java:104) - ##MTA-9a Unconfirmed AS
Request: 1133717956 Received from /10.10.1.2
 INFO [Thread-4] 2007-02-07 07:56:21,298 (KRBProperties.java:612) - Replacing property:
'minimum ps backoff' Old Value:'150' New Value: '150'
 INFO [Thread-4] 2007-02-07 07:56:21,324 (KDCMessageHandler.java:257) - AS-REQ contains
PKINIT - QA Tag.
 INFO [Thread-4] 2007-02-07 07:56:21,325 (KDCMessageHandler.java:279) - PK Request from
MTA received. Client is MTA - QA Tag
 INFO [Thread-4] 2007-02-07 07:56:21,365 (KDCMessageHandler.java:208) - ##MTA-9b KDC Reply
AS-REP Sent to /10.10.1.2:1039 Time(ms): 290
 WARN [main] 2005-11-07 07:56:23,193 (KDC.java:113) - Statistics Report   ASREP's: 1
 INFO [main] 2005-11-07 07:56:23,195 (KDC.java:121) - /pktcbl/mtaAsRepSent: 10
 INFO [main] 2005-11-07 07:56:23,195 (KDC.java:121) - /pktcbl/DHKeygenTotalTime: 1043
 INFO [main] 2005-11-07 07:56:23,196 (KDC.java:121) - /pktcbl/mtaAsReqRecvd: 10
 INFO [main] 2005-11-07 07:56:23,197 (KDC.java:121) - /pktcbl/DHKeygenNumOps: 20
 INFO [main] 2005-11-07 07:56:23,197 (KDC.java:121) - /pktcbl/total: 60
```

DPE Log Sample Output–Motorola MTA

```
dpe.cisco.com: 2007 02 07 07:56:24 EST: %BAC-DPE-6-4178: Adding Replay Packet:  []
dpe.cisco.com: 2007 02 07 07:56:24 EST: %BAC-PKTSNMP-6-0764: [System Description for MTA:
<<HW_REV: 1.0, VENDOR: Motorola Corporation, BOOTR: 8.1, SW_REV:
SBV5120-2.9.0.1-SCM21-SHPC, MODEL: SBV5120>>]
dpe.cisco.com: 2007 02 07 07:56:24 EST: %BAC-PKTSNMP-6-0764: [##MTA-15 SNMPv3 INFORM
Received From 10.10.1.2.]
dpe.cisco.com: 2007 02 07 07:56:24 EST: %BAC-DPE-6-0688: Received key material update for
device [1,6,01:11:82:61:5e:30]
dpe.cisco.com: 2007 02 07 07:56:24 EST: %BAC-PKTSNMP-6-0764: [##MTA-19 SNMPv3 SET Sent to
10.10.1.2]
dpe.cisco.com: 2007 02 07 07:56:24 EST: %BAC-TFTP-6-0310: Finished handling [read] request
from [10.10.1.2:1190] for [bpr0106001182615e300001]
dpe.cisco.com: 2007 02 07 07:56:25 EST: %BAC-PKTSNMP-6-0764: [##MTA-25 SNMP Provisioning
State INFORM Received from 10.10.1.2.  Value: 1]
```

Example 2

The following example features excerpts from the KDC and DPE log files for the Linksys CM2P2 MTA provisioned in the secondary CISCO1.COM realm:

KDC Log Sample Output–Linksys MTA

```
 INFO [Thread-8] 2007-02-07 08:00:10,664 (DHHelper.java:114) - Time to create DH key
pair(ms): 49
 INFO [Thread-8] 2007-02-07 08:00:10,759 (DHHelper.java:114) - Time to create DH key
pair(ms): 49
 INFO [Thread-8] 2007-02-07 08:00:10,817 (DHHelper.java:150) - Time to create shared
secret: 57 ms.
 INFO [Thread-8] 2007-02-07 08:00:10,819 (PKAsReqMsg.java:104) - ##MTA-9a Unconfirmed AS
Request: 1391094112 Received from /10.10.1.5
 INFO [Thread-8] 2007-02-07 08:00:10,828 (KRBProperties.java:612) - Replacing property:
'minimum ps backoff' Old Value:'150' New Value: '150'
 INFO [Thread-8] 2007-02-07 08:00:10,860 (KDCMessageHandler.java:257) - AS-REQ contains
PKINIT - QA Tag.
 INFO [Thread-8] 2007-02-07 08:00:10,862 (KDCMessageHandler.java:279) - PK Request from
MTA received. Client is MTA - QA Tag
 INFO [Thread-8] 2007-02-07 08:00:10,901 (KDCMessageHandler.java:208) - ##MTA-9b KDC Reply
AS-REP Sent to /10.10.1.5:3679 Time(ms): 296
 WARN [main] 2007-02-07 08:00:13,383 (KDC.java:113) - Statistics Report   ASREP's: 1
 INFO [main] 2007-02-07 08:00:13,384 (KDC.java:121) - /pktcbl/mtaAsRepSent: 11
 INFO [main] 2007-02-07 08:00:13,384 (KDC.java:121) - /pktcbl/DHKeygenTotalTime: 1141
```

DPE Log Sample Output–Linksys MTA

```
dpe.cisco.com: 2007 02 07 08:00:10 EST: %BAC-DPE-6-4112: Adding Replay Packet:  []
dpe.cisco.com: 2007 02 07 08:00:12 EST: %BAC-DPE-6-4178: Adding Replay Packet:  []
dpe.cisco.com: 2007 02 07 08:00:12 EST: %BAC-PKTSNMP-6-0764: [System Description for MTA:
Linksys Cable Modem with 2 Phone Ports (CM2P2) <<HW_REV: 2.0, VENDOR: Linksys, BOOTR:
2.1.6V, SW_REV: 2.0.3.3.11-1102, MODEL: CM2P2>>]
dpe.cisco.com: 2007 02 07 08:00:12 EST: %BAC-PKTSNMP-6-0764: [##MTA-15 SNMPv3 INFORM
Received From 10.10.1.5.]
dpe.cisco.com: 2007 02 07 08:00:12 EST: %BAC-DPE-6-0688: Received key material update for
device [1,6,00:0f:68:f9:42:f6]
dpe.cisco.com: 2007 02 07 08:00:12 EST: %BAC-PKTSNMP-6-0764: [##MTA-19 SNMPv3 SET Sent to
10.10.1.5]
dpe.cisco.com: 2007 02 07 08:00:18 EST: %BAC-TFTP-6-0310: Finished handling [read] request
from [10.10.1.5:1032] for [bpr0106000f68f942f60001]
dpe.cisco.com: 2007 02 07 08:00:18 EST: %BAC-PKTSNMP-6-0764: [##MTA-25 SNMP Provisioning
State INFORM Received from 10.10.1.5.  Value: 1]
```

Example 3

The following example features excerpts from the KDC and DPE log files for the SA WebStar DPX 2203 MTA provisioned in the secondary CISCO2.COM realm:

KDC Log Sample Output–SA MTA

```
 INFO [Thread-6] 2007-02-07 08:01:31,556 (DHHelper.java:114) - Time to create DH key
pair(ms): 49
 INFO [Thread-6] 2007-02-07 08:01:31,652 (DHHelper.java:114) - Time to create DH key
pair(ms): 50
 INFO [Thread-6] 2007-02-07 08:01:31,711 (DHHelper.java:150) - Time to create shared
secret: 57 ms.
 INFO [Thread-6] 2007-02-07 08:01:31,715 (PKAsReqMsg.java:104) - ##MTA-9a Unconfirmed AS
Request: 575634000 Received from /10.10.1.50
 INFO [Thread-6] 2007-02-07 08:01:31,727 (KRBProperties.java:612) - Replacing property:
'minimum ps backoff' Old Value:'150' New Value: '150'
 INFO [Thread-6] 2007-02-07 08:01:31,752 (KDCMessageHandler.java:257) - AS-REQ contains
PKINIT - QA Tag.
 INFO [Thread-6] 2007-02-07 08:01:31,753 (KDCMessageHandler.java:279) - PK Request from
MTA received. Client is MTA - QA Tag
 INFO [Thread-6] 2007-02-07 08:01:31,792 (KDCMessageHandler.java:208) - ##MTA-9b KDC Reply
AS-REP Sent to /10.10.1.50:3679 Time(ms): 292
 WARN [main] 2007-02-07 08:01:33,423 (KDC.java:113) - Statistics Report   ASREP's: 1
 INFO [main] 2007-02-07 08:01:33,424 (KDC.java:121) - /pktcbl/mtaAsRepSent: 12
 INFO [main] 2007-02-07 08:01:33,425 (KDC.java:121) - /pktcbl/DHKeygenTotalTime: 1240
 INFO [main] 2007-02-07 08:01:33,425 (KDC.java:121) - /pktcbl/mtaAsReqRecvd: 12
 INFO [main] 2007-02-07 08:01:33,426 (KDC.java:121) - /pktcbl/DHKeygenNumOps: 24
 INFO [main] 2007-02-07 08:01:33,426 (KDC.java:121) - /pktcbl/total: 72
```

DPE Log Sample Output–SA MTA

```
dpe.cisco.com: 2007 02 07 08:01:31 EST: %BAC-DPE-6-4112: Adding Replay Packet:  []
dpe.cisco.com: 2007 02 07 08:01:33 EST: %BAC-DPE-6-4178: Adding Replay Packet:  []
dpe.cisco.com: 2007 02 07 08:01:33 EST: %BAC-PKTSNMP-6-0764: [System Description for MTA:
S-A WebSTAR DPX2200 Series DOCSIS E-MTA Ethernet+USB (2)Lines VOIP <<HW_REV: 2.0, VENDOR:
S-A, BOOTR: 2.1.6b, SW_REV: v1.0.1r1133-0324, MODEL: DPX2203>>]
dpe.cisco.com: 2007 02 07 08:01:33 EST: %BAC-PKTSNMP-6-0764: [##MTA-15 SNMPv3 INFORM
Received From 10.10.1.50.]
dpe.cisco.com: 2007 02 07 08:01:33 EST: %BAC-DPE-6-0688: Received key material update for
device [1,6,00:0f:24:d8:6e:f5]
dpe.cisco.com: 2007 02 07 08:01:33 EST: %BAC-PKTSNMP-6-0764: [##MTA-19 SNMPv3 SET Sent to
10.10.1.50]
dpe.cisco.com: 2007 02 07 08:01:38 EST: %BAC-TFTP-6-0310: Finished handling [read] request
from [10.10.1.50:1037] for [bpr0106000f24d86ef50001]
dpe.cisco.com: 2007 02 07 08:01:39 EST: %BAC-PKTSNMP-6-0764: [##MTA-25 SNMP Provisioning
State INFORM Received from 10.10.1.50.  Value: 1]
```

Authoring Template for Provisioning Devices in Multiple Realms

You can use the template syntax described here to provision devices in a particular realm. The examples shown here are specific to the Motorola SBV5120 MTA (Example 6-2), the Linksys CM2P2 MTA (Example 6-3), and the SA WebStar DPX2203 MTA (Example 6-4).

Note You must modify these templates to suit the specifics of the MTA in your network.

Example 6-2 Template Used to Provision a Motorola MTA

```
#
# Example PacketCable MTA template: mot-mta.tmpl
#
# Note that this template is specific to the TI 401 MTA.
# This template must be modified to the specifics of your MTA.
#
# First, the start marker.
#
option 254 1
#
# Enable MTA
#
option 11 .pktcMtaDevEnabled.0,INTEGER,true
#
# Set CMS FQDN for each endpoint on the MTA.
# NOTE: the indexes (9 and 10 here) will differ per manufacturer.
#
option 11
.pktcNcsEndPntConfigTable.pktcNcsEndPntConfigEntry.pktcNcsEndPntConfigCallAgentId.9,STRING
,CMS.CISCO.COM
option 11
.pktcNcsEndPntConfigTable.pktcNcsEndPntConfigEntry.pktcNcsEndPntConfigCallAgentId.10,STRIN
G,CMS.CISCO.COM
#
# Set the realm org name.  This MUST match that contained in the cert chain used by the
device.
#
# "CableLabs, Inc."
option 11
.pktcMtaDevRealmTable.pktcMtaDevRealmEntry.pktcMtaDevRealmOrgName.'CISCO.COM',STRING,"Cabl
eLabs, Inc."
#
# Set the realm name and IPSec control for the CMS.
#
option 11
.pktcMtaDevCmsTable.pktcMtaDevCmsEntry.pktcMtaDevCmsIpsecCtrl.'CMS.CISCO.COM',INTEGER,true
option 11
pktcMtaDevCmsTable.pktcMtaDevCmsEntry.pktcMtaDevCmsKerbRealmName.'CMS.CISCO.COM',STRING,CI
SCO.COM
#
# Finally, the end marker.
#
option 254 255
```

Example 6-3 Template Used to Provision a Linksys MTA

Note that, in this template, the realm has been set to CISCO1.COM.

```
#
# Example PacketCable MTA template: linksys-mta.tmpl
#
# Note that this template is specific to the TI 401 MTA.
# This template must be modified to the specifics of your MTA.
#
# First, the start marker.
#
option 254 1
#
# Enable MTA
#
option 11 .pktcMtaDevEnabled.0,INTEGER,true
#
# Set CMS FQDN for each endpoint on the MTA.
# NOTE: the indexes (9 and 10 here) will differ per manufacturer.
#
option 11
.pktcNcsEndPntConfigTable.pktcNcsEndPntConfigEntry.pktcNcsEndPntConfigCallAgentId.9,STRING
,CMS.CISCO.COM
option 11
.pktcNcsEndPntConfigTable.pktcNcsEndPntConfigEntry.pktcNcsEndPntConfigCallAgentId.10,STRIN
G,CMS.CISCO.COM
#
# Set the realm org name.  This MUST match that contained in the cert chain used by the
device.
#
# "CableLabs, Inc."
option 11
.pktcMtaDevRealmTable.pktcMtaDevRealmEntry.pktcMtaDevRealmOrgName.'CISCO1.COM',STRING,"Cab
leLabs, Inc."
#
# Set the realm name and IPSec control for the CMS.
#
option 11
.pktcMtaDevCmsTable.pktcMtaDevCmsEntry.pktcMtaDevCmsIpsecCtrl.'CMS.CISCO.COM',INTEGER,true
option 11
pktcMtaDevCmsTable.pktcMtaDevCmsEntry.pktcMtaDevCmsKerbRealmName.'CMS.CISCO.COM',STRING,CI
SCO1.COM
#
# Finally, the end marker.
#
option 254 255
```

Example 6-4 Template Used to Provision an SA MTA

Note that, in the template, the realm has been set to CISCO2.COM.

```
#
# Example PacketCable MTA template: sa-mta.tmpl
#
# Note that this template is specific to the TI 401 MTA.
# This template must be modified to the specifics of your MTA.
#
# First, the start marker.
#
option 254 1
#
# Enable MTA
#
option 11 .pktcMtaDevEnabled.0,INTEGER,true
#
# Set CMS FQDN for each endpoint on the MTA.
# NOTE: the indexes (9 and 10 here) will differ per manufacturer.
#
option 11
.pktcNcsEndPntConfigTable.pktcNcsEndPntConfigEntry.pktcNcsEndPntConfigCallAgentId.9,STRING
,CMS.CISCO.COM
option 11
.pktcNcsEndPntConfigTable.pktcNcsEndPntConfigEntry.pktcNcsEndPntConfigCallAgentId.10,STRIN
G,CMS.CISCO.COM
#
# Set the realm org name.  This MUST match that contained in the cert chain used by the
device.
#
# "CableLabs, Inc."
option 11
.pktcMtaDevRealmTable.pktcMtaDevRealmEntry.pktcMtaDevRealmOrgName.'CISCO2.COM',STRING,"Cab
leLabs, Inc."
#
# Set the realm name and IPSec control for the CMS.
#
option 11
.pktcMtaDevCmsTable.pktcMtaDevCmsEntry.pktcMtaDevCmsIpsecCtrl.'CMS.CISCO.COM',INTEGER,true
option 11
pktcMtaDevCmsTable.pktcMtaDevCmsEntry.pktcMtaDevCmsKerbRealmName.'CMS.CISCO.COM',STRING,CI
SCO2.COM
#
# Finally, the end marker.
#
option 254 255
```

CHAPTER 7

Configuring Technologies in Prime Cable Provisioning

This chapter describes the tasks that you must perform when configuring Prime Cable Provisioning to support specific technologies:

- Configuring DOCSIS, page 7-1
- Configuring PacketCable, page 7-4
- Configuring DPoE, page 7-19
- Configuring CableHome, page 7-21

Configuring DOCSIS

This section describes the tasks that you must perform when configuring Prime Cable Provisioning to support the DOCSIS technologies.

Note See DOCSIS Option Support, page A-1, for information on DOCSIS options supported by this Prime Cable Provisioning release.

DOCSIS Workflow

Prime Cable Provisioning supports these versions of the DOCSIS specifications: 1.0, 1.1, 2.0, and 3.0.

To successfully configure Prime Cable Provisioning for DOCSIS operations, you must configure the components as described in Chapter 6, “Configuring Prime Cable Provisioning Components”, in addition to those described in this section.

Table 7-1 identifies the workflow to follow when configuring Prime Cable Provisioning to support DOCSIS.

Table 7-1 DOCSIS Workflow

Task	Refer to...
Step 1 Configure the RDU	
a. Configure all provisioned DHCP Criteria.	Configuring DHCP Criteria, page 11-12
b. Configure provisioned Class of Service.	Configuring Class of Service, page 11-1
c. Configure the promiscuous mode of operation.	System Defaults, page 11-10
Step 2 Configure the DPE	
a. Enable the TFTP service.	The **service tftp** *1..1* **ipv4 | ipv6 enabled true** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.
b. Optionally, enable the ToD service.	The **service tod** *1..1* **ipv4 | ipv6 enabled true** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.
Step 3 Configure Cisco Prime Network Registrar	
Configure client classes/selection tags to match those added for the provisioned DOCSIS modem DHCP Criteria.	*Cisco Prime Network Registrar 8.1 User Guide*

DOCSIS Shared Secret

Prime Cable Provisioning lets you define a different DOCSIS shared secret (DSS) for each cable modem termination system (CMTS). In this way, a compromised shared secret affects only a limited number of CMTS, instead of every CMTS in the deployment.

Although the DSS can be set for each DPE, you should set it on a provisioning-group basis. Also, ensure that it matches what has been configured for the CMTS in that provisioning group.

Caution Configuring multiple DSS within one provisioning group could, under some conditions, result in degraded CMTS performance. However, this factor has virtually no effect on Prime Cable Provisioning.

You can enter the shared secret as a clear text string or as an IOS-encrypted string. When entered in clear text, the DSS is encrypted to suit IOS version 12.2BC.

You can also set the DSS from the RDU using the administrator user interface or the API. In this case, the DSS is entered, stored at the RDU, and passed to all DPEs in clear text. Consequently, before a DSS entered this way is stored on the DPE, it is encrypted.

If you set the DSS directly at the DPE using the **dpe docsis shared-secret** command from the CLI, this DSS takes precedence over the one set from the RDU.

Resetting the DOCSIS Shared Secret

You can reset the DSS if the security of the DSS is compromised or to simply change the shared secret for administrative purposes.

To reset the DSS, run the **show running-config** command from the CMTS CLI, then copy and paste the DOCSIS shared secret from the configuration that appears into the DPE configuration. In this way, you can copy the configuration that you enter in a Cisco CMTS into the DPE CLI.

Note To change the shared secret as described, the CMTS must be running a software version later than version 12.2BC.

To change the DSS:

Step 1 Identify the provisioning group on which you need to reset the DOCSIS shared secret.

Step 2 Examine the list of DPEs and CMTS associated with the provisioning group.

Step 3 Change the primary DSS on the CMTS.

Step 4 Change the compromised DSS on the CMTS to the secondary DSS. This change is required to allow cable modems to continue to register until all the DOCSIS configuration files are successfully changed to use the new DSS.

Step 5 Determine which DPEs were affected and change the DSS on each accordingly.

Step 6 Confirm that the DOCSIS configuration files are using the new DSS and then remove the compromised secondary shared secret from the CMTS configuration.

Extended CMTS MIC Shared Secret

Prime Cable Provisioning lets you define a different Extended CMTS MIC (EMIC) shared secret for each cable modem termination system (CMTS) for EMIC calculation.

The CMTS must support a configuration for the shared secret for EMIC calculation to differ from the shared secret for pre-3.0 DOCSIS CMTS MIC calculation. In the absence of such configuration, the CMTS MUST use the same shared secret for Extended CMTS MIC Digest calculation as for pre-3.0 DOCSIS CMTS MIC digest calculation.

In this way, a compromised shared secret affects only a limited number of CMTS, instead of every CMTS in the deployment.

Similar to DSS, EMIC DOCSIS shared secret can be set for each DPE, you should set it on a provisioning-group basis. Also, ensure that it matches what has been configured for the CMTS in that provisioning group.

Caution Configuring multiple EMIC DOCSIS Shared Secret within one provisioning group could, under some conditions, result in degraded CMTS performance. However, this factor has virtually no effect on Prime Cable Provisioning.

You can enter the shared secret as a clear text string or as an IOS-encrypted string. When entered in clear text, the EMIC shared secret is encrypted to suit IOS version 12.2BC.

You can also set the EMIC Shared Secret from the RDU using the administrator user interface or the API. In this case, the DOCSIS shared secret is entered, stored at the RDU, and passed to all DPEs in clear text. Consequently, before an Extended MIC shared secret entered this way is stored on the DPE, it is encrypted.

If you set the Extended MIC shared secret directly at the DPE using the **dpe docsis emic shared-secret** command from the CLI, this Extended MIC shared secret takes precedence over the one set from the RDU.

Resetting the Extended EMIC Shared Secret

You can reset the Extended MIC shared secret if the security of the EMIC shared secret is compromised or to simply change the shared secret for administrative purposes.

To reset the DSS, run the **show running-config** command from the CLI, then copy and paste the EMIC shared secret from the configuration that appears into the DPE configuration. In this way, you can copy the configuration that you enter in a Cisco CMTS into the DPE CLI.

Note To change the shared secret as described, the CMTS must be running a software version later than version 12.2(11)CX.

To change the Extended MIC shared secret:

Step 1 Identify the provisioning group on which you need to reset the EMIC shared secret.

Step 2 Examine the list of DPEs and CMTS associated with the provisioning group.

Step 3 Change the primary EMIC shared secret on the CMTS.

Step 4 Change the compromised EMIC shared secret on the CMTS to the secondary EMIC shared secret. This change is required to allow cable modems to continue to register until all the DOCSIS configuration files are successfully changed to use the new DSS.

Step 5 Determine which DPEs were affected and change the EMIC shared secret on each accordingly.

Step 6 Confirm that the DOCSIS configuration files are using the new EMIC shared secret and then remove the compromised secondary shared secret from the CMTS configuration.

Configuring PacketCable

This section describes the tasks that you must perform when configuring Prime Cable Provisioning to support the Packetcable technologies and bring a PacketCable voice deployment into service.

This chapter contains information on these variants of PacketCable:

- Configuring PacketCable Basic, page 7-8
- Configuring PacketCable Secure, page 7-10
- Configuring PacketCable 2.0, page 7-15

For information that will help you solve issues in a PacketCable voice technology deployment, see Troubleshooting PacketCable eMTA Provisioning, page 26-11.

This chapter assumes that you are familiar with the contents of the PacketCable Multimedia Terminal Adapter (MTA) Device Provisioning Specification, PKT-SP-PROV1.5-I03-070412. For details, see the PacketCable website.

PacketCable Workflows

Prime Cable Provisioning supports these versions of the PacketCable specifications: 1.0, 1.5 and 2.0.

Prime Cable Provisioning also supports two variants of PacketCable voice services: the default Secure mode and the non-secure Basic mode. PacketCable Basic is much the same as the standard PacketCable, except for the lack of security found in the non-secure variant.

This section identifies the tasks that you must perform for each variant.

- PacketCable Basic, page 7-5
- PacketCable Secure, page 7-6

Note The workflows in this section assume that you have loaded an appropriate PacketCable configuration file and the correct MIBs.

PacketCable Basic

You perform the PacketCable-related tasks described in this section only after completing those described in Chapter 6, "Configuring Prime Cable Provisioning Components".

Table 7-2 identifies the workflow to follow when configuring PacketCable Basic on Prime Cable Provisioning.

Note Tasks marked with an asterisk (*) are mandatory.

Table 7-2 PacketCable Basic Workflow

	Task	Refer to...
Step 1	Configure the DPE	
	a. Enable PacketCable.*	The **service packetcable** *1..1* **enable** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
Step 2	Configure DHCP	
	a. Configure dynamic DNS for the MTA scopes.	*Cisco Prime Network Registrar 8.1 User Guide*
	b. Configure client classes/scope-selection tags that match those added for provisioned PacketCable MTA DHCP criteria.*	*Cisco Prime Network Registrar 8.1 User Guide*
Step 3	Configure DNS	
	Configure dynamic DNS for each DHCP server.	*Cisco Prime Network Registrar 8.1 User Guide*

Table 7-2 PacketCable Basic Workflow (continued)

	Task	Refer to...
Step 4	Configure a Class of Service, which must contain the following properties:	
	a. */pktcbl/prov/flow/mode* This property commands the specific flow that an MTA uses. Set this property to either: – BASIC.1—Executes the BASIC.1 flow. – BASIC.2—Executes the BASIC.2 flow. **Note** You can configure this property anywhere on the device-property hierarchy.	Configuring Class of Service, page 11-1
	b. */cos/packetCableMTA/file* This property contains the name of the configuration file that is to be presented to the MTA. The configuration file is stored as a file in Prime Cable Provisioning. The configuration file presented to a Basic MTA must contain the Basic integrity hash. If you are using a dynamic configuration template, the hash is inserted transparently during template processing. You can use the dynamic template for provisioning in both Secure and Basic modes. However, if the file is a Secure static configuration file, you must convert this file to a Basic static configuration file because Secure and Basic static configuration files are not interoperable. For details on how to perform this conversion, see Activating PacketCable Basic Flow, page 19-12.	Configuring Class of Service, page 11-1

PacketCable Secure

Prime Cable Provisioning supports two variants of PacketCable Secure:

- North American PacketCable
- European PacketCable

Euro-PacketCable services are the European equivalent of the North American PacketCable standard. The only significant difference between the two is that Euro PacketCable uses different MIBs. For details, see Euro-PacketCable MIBs, page 7-18.

You perform the PacketCable-related tasks described in this section only after configuring the components as explained in Chapter 6, “Configuring Prime Cable Provisioning Components”.

Note For PacketCable-compliant operations, the maximum allowable clock skew between the MTA, KDC, and DPE is 300 seconds (5 minutes). This value is the default setting.

Table 7-3 identifies the workflow to follow when configuring Prime Cable Provisioning to support PacketCable Secure.

Note Tasks marked with an asterisk (*) are mandatory.

Table 7-3 PacketCable Secure Workflow

	Task	Refer to...
Step 1	Configure the RDU	
	a. Enable the autogeneration of Multimedia Terminal Adapter (MTA) FQDNs.	Automatic FQDN Generation, page 16-1
	b. Configure all provisioned DHCP Criteria.	Configuring DHCP Criteria, page 11-12
	c. Configure all provisioned Class of Service.	Configuring Class of Service, page 11-1
	d. Configure an SNMPv3 cloning key.*	Configuring SNMPv3 Cloning on RDU and DPE for Secure Communication with PacketCable MTAs, page 7-17
	e. If you are using Euro PacketCable, configure the RDU to use Euro-PacketCable MIBs.	Configuring Euro-PacketCable MIBs, page 7-19
Step 2	Configure the DPE	
	a. Configure a KDC service key.*	The **service packetcable** *1..1* **registration kdc-service-key** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
	b. Configure a privacy policy.*	The **service packetcable** *1..1* **registration policy-privacy** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
	c. Configure an SNMPv3 cloning key.*	The **service packetcable** *1..1* **snmp key-material** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
	d. Enable PacketCable.*	The **service packetcable** *1..1* **enable** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
	e. Optionally, configure MTA file encryption.	The **service packetcable** *1..1* **registration encryption enable** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*
Step 3	Configure the KDC	
	a. Obtain a KDC license from your Cisco representative.	KDC Certificate, page 26-26
	b. Configure a certificate chain using the PKCert.sh tool. For Euro PacketCable, use the **-e** option.	Using PKCert.sh, page 28-3

Table 7-3 PacketCable Secure Workflow (continued)

Task	Refer to...
c. Configure a service key pair for each DPE's provisioning FQDN.	Using KeyGen Tool, page 28-9
d. Configure service keys for the ticket-granting-ticket (TGT).	Using KeyGen Tool, page 28-9
e. Configure NTP Synchronization.	Solaris and Linux documentation for information on configuring NTP
Step 4 Configure DHCP	
a. Configure all necessary PacketCable properties.	Using changeNRProperties.sh, page 28-11
b. Configure dynamic DNS for the MTA scopes.	*Cisco Prime Network Registrar 8.1 User Guide*
c. Configure client classes/scope-selection tags to match those added for provisioned PacketCable MTA DHCP criteria.*	*Cisco Prime Network Registrar 8.1 User Guide*
Step 5 Configure DNS	
a. Configure dynamic DNS for each DHCP server.	*Cisco Prime Network Registrar 8.1 User Guide*
b. Configure a zone for the KDC realm.	*Cisco Prime Network Registrar 8.1 User Guide*

Configuring PacketCable Basic

Prime Cable Provisioning also supports PacketCable Basic, which offers a simpler, DOCSIS-like, non-secure provisioning flow. Table 7-4 describes the BASIC.1 flow using the provisioning workflow in Figure 7-1 on page 7-10.

Table 7-4 PacketCable Basic eMTA Provisioning

Step	Workflow	Description
MTA-1	DHCP Broadcast Discover	Executes as for the Secure flow.
MTA-2	DHCP Offer	If the provisioning system is configured to provision the MTA in BASIC.1 mode, the provisioning system returns a DHCP Offer containing Option 122 suboption 6, which contains the special reserved realm name "BASIC.1". This reserved realm name commands the MTA to use the BASIC.1 provisioning flow. This Offer also contains the provisioning system IP address in Option 122.3, and the file and siaddr fields are populated with the configuration file location of the MTA.
MTA-3	DHCP Request	The remainder of the MTA DHCP exchange is executed (Request and Ack exchanged).
MTA-4	DHCP Ack	

Table 7-4 PacketCable Basic eMTA Provisioning (continued)

Step	Workflow	Description
MTA-22	Telephony Config File Request	The MTA skips directly to step MTA-22. Using the file and siaddr information, the MTA copies its configuration file from the provisioning system via TFTP. Note that Prime Cable Provisioning integrates the TFTP server into the DPE component. **Note** No authentication of MTA/provisioning server or encryption occurs.
MTA-23	Telephony Config File	

The BASIC.2 flow is identical to BASIC.1, with the following exceptions:

- "BASIC.2" is populated into the MTA's DHCP Option 122 suboption 6.
- The MTA issues a provisioning status SNMPv2c INFORM at the very end of the flow, MTA-25 (DHCP Option 122 suboption 3 specifies the Inform target).

The PacketCable Basic flow is similar to the DOCSIS flow with the following differences:

- There is no ToD exchange between MTA and the provisioning system.
- The MTA configuration file contains an integrity hash. Specifically, the SHA1 hash of the entire content of the configuration file is populated into a pktcMtadevConfigFileHash SNMP VarBind and placed within a TLV 11 just before the end of file TLV.
- BASIC.2 flow issues a provisioning status SNMPv2c Inform after the MTA receives and processes its configuration file. This Inform notifies Prime Cable Provisioning if MTA provisioning completed successfully. If there is a problem, an error is generated and an event sent from the DPE to the RDU, then on to a Prime Cable Provisioning client. This Inform is useful while debugging configuration file issues.

For additional information about the DOCSIS flow, see Chapter 7, "Configuring DOCSIS."

Note Before using the PacketCable Basic provisioning flow, ensure that you are using a PacketCable Basic-capable eMTA. The eMTA must report that it is Basic-capable with its DHCP Discover Option 60, TLV 5.18 (supported flows).

PacketCable TLV 38 and MIB Support

Prime Cable Provisioning supports the complete set of PacketCable 1.5 MIBs.

Prime Cable Provisioning supports TLV 38 in PacketCable configuration templates. This TLV lets you configure multiple SNMP notification targets. Configuration of this TLV means that all notifications are also issued to the targets configured through TLV 38.

SNMP v2C Notifications

Prime Cable Provisioning supports both SNMP v2C TRAP and INFORM notifications from the PacketCable MTA.

Configuring PacketCable Secure

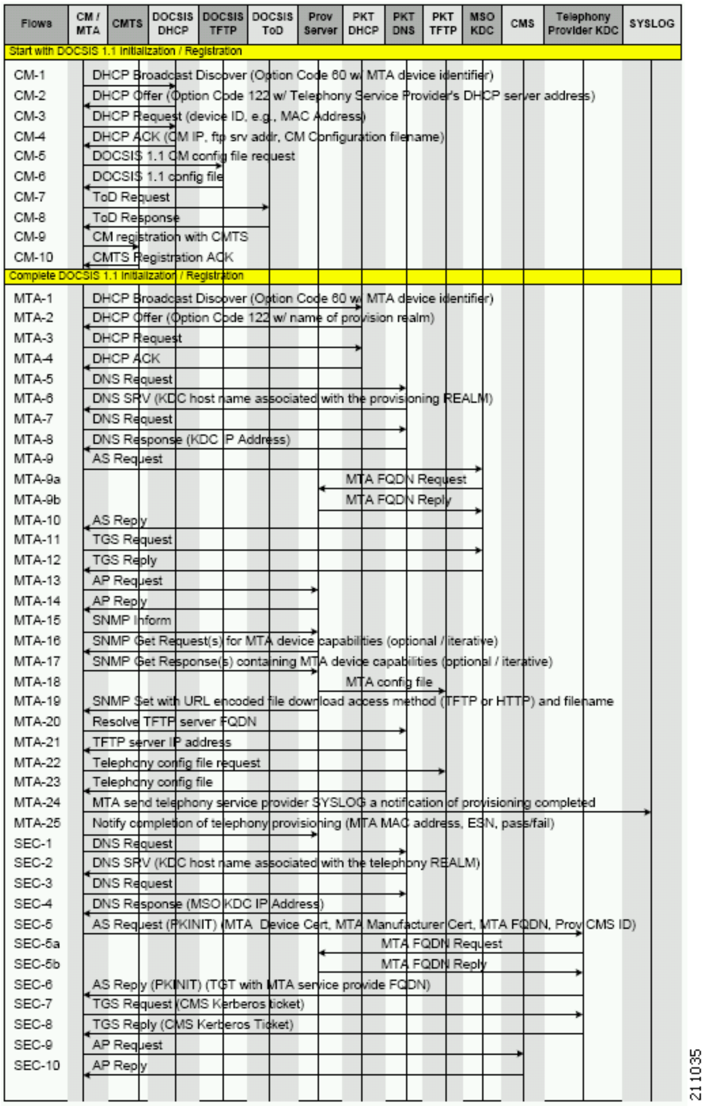
This section deals exclusively with Secure PacketCable voice provisioning. PacketCable Secure is designed to minimize the possibility of theft of telephony service, malicious disruption of service, and so on. PacketCable Secure depends on the Kerberos infrastructure to mutually authenticate the MTA and the provisioning system; in Prime Cable Provisioning, the Key Distribution Center (KDC) functions as the Kerberos server. SNMPv3 is also used to secure the conversation between the MTA and the provisioning system.

Cisco Prime Cable Provisioning PacketCable Secure Provisioning Flow

All PacketCable provisioning flows are defined as a sequence of steps.

Figure 7-1 illustrates the Secure provisioning flow for PacketCable eMTAs.

Figure 7-1 Embedded-MTA Secure Power-On Provisioning Flow


Flows
CM / MTA
CMTS
DOCSIS DHCP
DOCSIS TFTP
DOCSIS ToD
Prov Server
PKT DHCP
PKT DNS
PKT TFTP
MSO KDC
CMS
Telephony Provider KDC
SYSLOG
Start with DOCSIS 1.1 Initialization / Registration
CM-1 DHCP Broadcast Discover (Option Code 60 w/ MTA device identifier)
CM-2 DHCP Offer (Option Code 122 w/ Telephony Service Provider's DHCP server address)
CM-3 DHCP Request (device ID, e.g., MAC Address)
CM-4 DHCP ACK (CM IP, ftp srv addr, CM Configuration filename)
CM-5 DOCSIS 1.1 CM config file request
CM-6 DOCSIS 1.1 config file
CM-7 ToD Request
CM-8 ToD Response
CM-9 CM registration with CMTS
CM-10 CMTS Registration ACK
Complete DOCSIS 1.1 Initialization / Registration
MTA-1 DHCP Broadcast Discover (Option Code 60 w/ MTA device identifier)
MTA-2 DHCP Offer (Option Code 122 w/ name of provision realm)
MTA-3 DHCP Request
MTA-4 DHCP ACK
MTA-5 DNS Request
MTA-6 DNS SRV (KDC host name associated with the provisioning REALM)
MTA-7 DNS Request
MTA-8 DNS Response (KDC IP Address)
MTA-9 AS Request
MTA-9a MTA FQDN Request
MTA-9b MTA FQDN Reply
MTA-10 AS Reply
MTA-11 TGS Request
MTA-12 TGS Reply
MTA-13 AP Request
MTA-14 AP Reply
MTA-15 SNMP Inform
MTA-16 SNMP Get Request(s) for MTA device capabilities (optional / iterative)
MTA-17 SNMP Get Response(s) containing MTA device capabilities (optional / iterative)
MTA-18 MTA config file
MTA-19 SNMP Set with URL encoded file download access method (TFTP or HTTP) and filename
MTA-20 Resolve TFTP server FQDN
MTA-21 TFTP server IP address
MTA-22 Telephony config file request
MTA-23 Telephony config file
MTA-24 MTA send telephony service provider SYSLOG a notification of provisioning completed
MTA-25 Notify completion of telephony provisioning (MTA MAC address, ESN, pass/fail)
SEC-1 DNS Request
SEC-2 DNS SRV (KDC host name associated with the telephony REALM)
SEC-3 DNS Request
SEC-4 DNS Response (MSO KDC IP Address)
SEC-5 AS Request (PKINIT) (MTA Device Cert, MTA Manufacturer Cert, MTA FQDN, Prov CMS ID)
SEC-5a MTA FQDN Request
SEC-5b MTA FQDN Reply
SEC-6 AS Reply (PKINIT) (TGT with MTA service provide FQDN)
SEC-7 TGS Request (CMS Kerberos ticket)
SEC-8 TGS Reply (CMS Kerberos Ticket)
SEC-9 AP Request
SEC-10 AP Reply
211035

Note It is strongly recommended that you use a protocol analyzer (protocol sniffer) with the ability to capture data packets to understand exactly which step is failing.

In addition, the content of the KDC log file is critical to understanding the root cause of any KDC failure.

When diagnosing problems in provisioning an embedded Multimedia Terminal Adapters (eMTA), the flow description in Table 7-5 helps identify which step in the PacketCable provisioning flow is failing.

Table 7-5 PacketCable Secure eMTA Provisioning

Step	Workflow	Description
CM-1	DHCP Broadcast Discover	This is similar to the DOCSIS cable modem (CM) boot flow for DHCPv4 or DHCPv6 with DHCP options added to provide the MTA with a list of PacketCable DHCP servers from which the MTA is allowed to accept DHCP offers.
CM-2	DHCP Offer	
CM-3	DHCP Request	
CM-4	DHCP Ack	
CM-5	DOCSIS 1.1 CM Config File Request	
CM-6	DOCSIS 1.1 Config File	
CM-7	ToD Request	
CM-8	ToD Response	
CM-9	CM Registration with CMTS (cable modem termination system)	
CM-10	CMTS Registration Ack	

Table 7-5 PacketCable Secure eMTA Provisioning (continued)

Step	Workflow	Description
MTA-1	DHCP Broadcast Discover	Using DHCP, the MTA announces itself as a PacketCable MTA and provides information on the capabilities and provisioning flows it supports (Secure, Basic, and so on.). The MTA also obtains addressing information and DHCP Option 122. DHCP Option 122 contains the PacketCable provisioning server address and the security realm name. This information is used to allow the MTA to contact the KDC and provisioning server. Some key troubleshooting hints are: • Check the DHCP relay agent on the CMTS for the correct configuration; ensure that your CMTS points to the correct DHCP server. • Verify that you have the correct routing between the MTA, CMTS, DHCP server, and the DPE. • Verify that secondary subnets are configured correctly on the CMTS. • Check the Cisco Prime Network Registrar DHCP configuration. Verify if the scopes are configured, if IP addresses are available, and if all secondary subnets are configured. • Check the Prime Cable Provisioning configuration. Check the *cnr_ep.properties* file and ensure that the required PacketCable Network Registrar extension properties are configured. For more information, see Mapping PacketCable DHCP Options to Prime Cable Provisioning Properties, page B-1. If a packet trace reveals that the MTA is cycling between steps MTA-1 and MTA-2, there could be a problem with the configuration of DHCP Option 122 (realm name or provisioning server FQDN suboptions), DHCP Option 12 (hostname), or DHCP Option 15 (domain name).
MTA-2	DHCP Offer	
MTA-3	DHCP Request	
MTA-4	DHCP Ack	
MTA-5	DNS Request	MTA uses the security realm name (delivered within DHCP Option 122) to perform a DNS SRV lookup on the KDC service and then resolves the KDC IP address. Some key troubleshooting hints are: • Use a packet sniffer to watch for misdirected or malformed DNS packets sent to the Network Registrar DNS. • Set the Network Registrar DNS log level to detailed packet tracing and verify what arrives there. • Check the DNS configuration—The DNS server specified in *cnr_ep.properties* must contain the realm zone, the SRV record, and the DNS ‘A’ record for the KDC.
MTA-6	DNS Srv	
MTA-7	DNS Request	
MTA-8	DNS Response	

Table 7-5 PacketCable Secure eMTA Provisioning (continued)

Step	Workflow	Description
MTA-9	AS Request	The AS-REQ request message is used by the KDC to authenticate the MTA. Some key troubleshooting hints are: • Check the KDC log file to determine if the AS-REQ arrives and to observe any errors or warnings. • Check that the KDC is configured with the correct MTA_Root certificate. The Manufacturer and Device certificates sent by the MTA within the AS-REQ message must chain with the MTA_Root certificate installed at the KDC.
MTA-9a	MTA FQDN Request	The KDC extracts the MTA MAC address from the MTA certificate and sends it to the provisioning server for validation. If the provisioning server has the FQDN for that MAC address, it is returned to the KDC. The KDC then compares the FQDN received from the MTA to the FQDN received in the FQDN-REP reply message. Some key troubleshooting hints are: • Use a packet sniffer to watch for misdirected or malformed DNS packets. The MTA passes the provisioning server FQDN (which the MTA received in DHCP Option 122) within the AS-REP message to the KDC. The KDC then uses this FQDN to resolve the IP address of the provisioning server. • Check the filenames and content of the KDC key file; the KDC service key in the DPE must match the service key at the KDC. The names of the service key files at the KDC are critical.
MTA-9b	MTA FQDN Reply	
MTA-10	AS Reply (AS-REP)	The KDC grants a provisioning service ticket to the MTA and also sends the Service Provider, Local System Provider (optional), and KDC certificate to the MTA. The MTA then verifies if the certificates sent by the KDC chain to the Service Provider Root certificate stored in the MTA. If these certificates do not chain, the MTA loops back to step MTA-1 of the provisioning flow. See Using PKCert.sh, page 28-3, for additional information on the *KDC.cer* file. A key troubleshooting hint: Verify if the KDC log files show that the AS-REP message was sent to the device. If a packet trace reveals the MTA is cycling between steps MTA-1 and MTA-10, there is a problem with the service provider certificate chain.
MTA-11	TGS Request	The MTA receives either a service ticket or a ticket-granting-ticket (TGT) following step MTA-10. If the MTA had obtained a TGT instead of a service ticket in step MTA-10, it contacts the ticket-granting-server (KDC) to obtain a service ticket.
MTA-12	TGS Reply	The KDC sends a service ticket in the TGS Reply to the MTA.
MTA-13	AP Request (AP-REQ)	The MTA presents the ticket (received at step MTA-10) to the provisioning server specified by DHCP Option 122.

Table 7-5 PacketCable Secure eMTA Provisioning (continued)

Step	Workflow	Description
MTA-14	AP Reply (AP-REP)	The provisioning server uses the KDC shared secret to decrypt the AP-REQ, validates the provisioning server ticket presented by the MTA, and sends AP-REP with SNMPv3 keys. SNMPv3 is now authenticated and (optionally) encrypted.
MTA-15	SNMP Inform	The MTA signals to the provisioning server that it is ready to receive provisioning information.
MTA-16	SNMP Get Request	SNMPv3—If the provisioning server (DPE) requires additional device capabilities, it sends the MTA one or more SNMPv3 Get requests to obtain the required information on MTA capability. The provisioning server (DPE) may use a GetBulk request to request a bulk of information in a single message.
MTA-17	SNMP Get Response	SNMPv3—The MTA sends to the provisioning server (DPE) a response for each GetRequest that contains information on MTA capabilities requested in step MTA-16.
MTA-18	MTA Config file	Using information made available in steps MTA-16 and MTA-17, the provisioning server (DPE) determines the contents of the MTA configuration data file.
MTA-19	SNMP Set	SNMPv3—The provisioning server performs an SNMPv3 Set to the MTA containing the URL for the MTA configuration file, encryption key for the file, and the file hash value.
MTA-20	Resolve TFTP Server FQDN	DNS Request—If the URL-encoded access method contains an FQDN instead of an IPv4 address, the MTA uses the DNS server of the service provider network to resolve the FQDN into an IPv4 address of the TFTP server or the HTTP server.
MTA-21	TFTP Server IP Address	DNS Response—The DNS server returns the IPv4 IP address of the service provider network as requested in step MTA-20.
MTA-22	Telephony Config File Request	The MTA proceeds to download the VoIP configuration file from the specified TFTP server. Note that Prime Cable Provisioning integrates the TFTP server into the DPE component.
MTA-23	Telephony Config File	
MTA-24	MTA Send	The MTA optionally sends a syslog notification to the service provider that provisioning is complete.
MTA-25	Notify completion of telephony provisioning	The MTA signals to the provisioning server if the new configuration is acceptable.
SEC-1 to SEC-10	These steps are the post-MTA provisioning security flow and are not applicable to Prime Cable Provisioning provisioning. This flow involves getting Kerberos tickets associated with each CMS with which the MTA communicates. For details, see the PacketCable Security Specifications.	

Configuring SRV Records in the Network Registrar DNS Server

You must configure the Network Registrar DNS server to operate with the KDC. To set up this configuration, see your Network Registrar documentation and these instructions.

Note We recommend that you create a zone name that matches the desired realm name, and that the only DNS record in this special zone (other than the records required by the DNS server to maintain the zone) should be the SRV record for the realm. This example assumes that the desired Kerberos realm is voice.example.com, and that all other KDC, Network Registrar, and DPE configurations have been performed. The FQDN of the KDC is assumed to be kdc.example.com.

Step 1 Start the **nrcmd** command-line tool (which resides, by default, in the */opt/nwreg2/local/usrbin* directory).

Step 2 Enter your username and password.

Step 3 To create a zone for the Kerberos realm, enter:

```
nrcmd> zone voice.example.com create primary address_of_nameserver hostmaster
```

where *address_of_nameserver* specifies the IP address of the name server.

Step 4 To add the SRV record to the new zone, enter:

```
nrcmd> zone voice.example.com. addRR _kerberos._udp. srv 0 0 88 KDC_FQDN
```

where *KDC_FQDN* specifies the FQDN of the KDC.

Step 5 To save and reload the DNS server, enter:

```
nrcmd> save
```

```
nrcmd> dns reload
```

Configuring PacketCable 2.0

PacketCable 2.0 supports the convergence of voice, video, data, and mobility technologies. It is based on Session Initiation Protocol (SIP) and IP multimedia system (IMS) and supports configuration and management of Non-Embedded User Equipment (UE) as well as Embedded User Equipment (E-UE).

Prime Cable Provisioning supports only the UEs that are embedded with a DOCSIS Cable Modem and are called as E-UE or Embedded Digital Voice Adapter (E-DVA). E-DVA supports RST (Residential SIP Telephony.

Note Prime Cable Provisioning supports E-DVA provisioning only in IPv4 but both in Basic and Secure modes.

Adding a Dial Plan for PacketCable 2.0 Groovy

A dial plan is provisioned on the UE to inform the UE about how dialed digits should be interpreted. A dial plan is an ordered set of regular expressions combined with some special tokens that represent actions to be carried out by the UE when a regular expression is matched.

The dial plan is organized into a list of rules. The UE must apply the dial plan rules sequentially and upon matching a pattern, including timers, the UE must perform the specified action or actions.

To create a dial plan you must be familiar with the notation and content of Augmented Backus-Naur Form (ABNF) defined in RFC 4234. Below is a sample dial plan that you can use as a reference.

Note If any issues are encountered while determining the device capabilities, Prime Cable Provisioning defaults to the Secure mode. While adding a dial plan, you can either use a groovy script or a binary file but not a template.

A sample groovy file to create a dial plan

```
def dialPlan = '''
            TIMER S=4.000000
            TIMER Z=2.000000

            domain = "@ims.packetcable.com"
            dialString = ";user=dialstring"
            dialPhone = ";user=phone"

            homeEmergencyNumber = "911"
            localEmergencyNumber = "911"

            MAP MainTable =
            "0S" : MAKE-CALL
            "0#" : MAKE-CALL
            "00" : MAKE-CALL
            "(=Emergency)" : EMERGENCY-CALL("sip:" "911" =domain =dialPhone)
            "(=N11)" : MAKE-CALL("sip:" #1v =domain =dialString)
            "(=SpeedDial)" : MAKE-CALL("sip:" #1v =domain =dialString)
            "(=PhoneNumber)" : MAKE-CALL("sip:" #1v =domain =dialPhone)
            "(=ImmediateVSCs)" : RETURN
            "(=DelayedVSCs)" : RETURN
            "(x{1-20})S" : MAKE-CALL("sip:" #1 =domain =dialPhone)
            "(x{1-20})#" : MAKE-CALL("sip:" #1 =domain =dialPhone)
'''
* PKTC-IETF-MTA-MIB pktcMtaDevEnabled (1.3.6.1.2.1.140.1.1.6.0)
*/
configFile.add(TLV_SNMP("1.3.6.1.2.1.140.1.1.6.0", "Integer", "1"))
/*
* Device Level Configuration (Secure flow only):
* Include required Secure-flow realm TLVs
*/
if (isSecureProvFlowMode)
{
   // PKTC-IETF-MTA-MIB pktcMtaDevRealmName.1 (1.3.6.1.2.1.140.1.3.6.1.2.1)
   configFile.add(
       TLV_SNMP("1.3.6.1.2.1.140.1.3.6.1.2.1", "STRING", realmName))
   // PKTC-IETF-MTA-MIB pktcMtaDevRealmOrgName.1 (1.3.6.1.2.1.140.1.3.6.1.5.1)
       configFile.add(
       TLV_SNMP("1.3.6.1.2.1.140.1.3.6.1.5.1", "STRING", realmOrgName))
}

configFile.add(option.createOptionValue(OptionSyntax.SNMP,"64",[".pktcEUERSTDMValue.1","ST
RING",dialPlan]));
```

Note The sample PacketCable 2.0 groovy script (example_edva.groovy) uses numeric OIDs for the PacketCable Secure-mode SNMP TLVs (pktcMtaDevRealmName, pktcMtaDevRealmOrgName).

If the TLV length of Option 64 exceeds 4500, you must update the property /default/asnParser/bufferLength=20000 in /opt/CSCObac/api/conf/api.properties and in /opt/CSCObac/rdu/conf/rdu.properties.

Configuring SNMPv3 Cloning on RDU and DPE for Secure Communication with PacketCable MTAs

Prime Cable Provisioning lets you enable an external network manager for SNMPv3 access to MTA devices. Additionally, the RDU is capable of performing SNMPv3 operations in a specific MTA.

To enable this capability, set the security key material at the DPEs and RDU. After the key material has been set, the Prime Cable Provisioning application programming interface (API) calls that are used to create cloned SNMPv3 entries are enabled.

Note Enabling this capability impacts provisioning performance.

Creating the Key Material and Generating the Key

Creating the key material is a two-step process:

1. Run a script command on the RDU.
2. Run a CLI command on the DPE.

Note This shared secret is not the same shared secret as the CMTS or the Prime Cable Provisioning shared secrets.

To create the key material:

Step 1 From the *BPR_HOME/rdu/bin* directory, run this script on the RDU:

```
# generateSharedSecret.sh password
```

where *password* is any password, from 6 to 20 characters, that you create. This password is then used to generate a 46-byte key. This key is stored in a file, called *keymaterial.txt*, that resides in the *BPR_HOME/rdu/conf* directory.

Step 2 Run the **service packetcable** *1..1* **snmp key-material** DPE CLI command, with the *password* used in Step 1 to generate that key, on all DPEs for which this voice technology is enabled. This command generates the same 46-byte key on the DPE and ensures that the RDU and DPEs are synchronized and can communicate with the MTA securely.

Euro PacketCable

Euro-PacketCable services are essentially the European equivalent of North American PacketCable services with the following differences:

- Euro PacketCable uses different MIBs.
- Euro PacketCable uses a different set of device certificates (*MTA_Root.cer*) and service provider certificates (Service Provider Root).

  For Euro-PacketCable certificates, the *kdc.ini* file must have the *euro-packetcable* property set to true. The KDC supports Euro-PacketCable (tComLabs) certificate chains. The following is a sample Euro PacketCable-enabled KDC configuration file.

```
[general]
interface address = 10.10.10.1
FQDN = servername.cisco.com
maximum log file size = 10000
n saved log files = 100
log debug level = 5 minimum
ps backoff = 150 maximum
ps backoff = 300
euro-packetcable = true
```

  When using Euro PacketCable, ensure that the value of the PacketCable property */pktcbl/prov/locale* is set to EURO. The default is NA (for North America). You can specify the locale in the Configuration File utility. See Using Configuration File Utility for Template, page 19-33, for more information.

Euro-PacketCable MIBs

Euro-PacketCable MIBs are essentially snapshots of draft-IETF MIBs. MTA configuration files consist of SNMP VarBinds that reference the MIBs. There are substantial differences between the North American PacketCable and Euro-PacketCable MIBs; therefore, the North American PacketCable and Euro-PacketCable configuration files are incompatible. During installation, sample files for North American PacketCable (*cw29_config.tmpl*) and Euro PacketCable (*ecw15_mta_config.tmpl*) are copied to the *BPR_HOME/rdu/samples* directory.

Prime Cable Provisioning ships with the following Euro-PacketCable MIBs:

- DOCS-IETF-BPI2-MIB
- INTEGRATED-SERVICES-MIB
- DIFFSERV-DSCP-TC
- DIFFSERV-MIB
- TCOMLABS-MIB
- PKTC-TCOMLABS-MTA-MIB
- PKTC-TCOMLABS-SIG-MIB

Configuring Euro-PacketCable MIBs

To configure Prime Cable Provisioning to use Euro-PacketCable MIBs, you must change the Prime Cable Provisioning RDU property that specifies the MIBs to be loaded. By default, this property contains the PacketCable MIBs.

You can change the property in one of the following ways:

- Modify *rdu.properties* and restart the RDU.
- On the administrator user interface, navigate to **Configuration > Defaults > System Defaults** and replace the MIB list with the list shown below. You do not need to restart the RDU.
- Use the Prov API *changeSystemDefaults()* call. You do not need to restart the RDU.

The property name is */snmp/mibs/mibList* (properties file) or `SNMPPropertyKeys.MIB_LIST` (the Prov API constant name). The property value is a comma-separated value (CSV) consisting of the required MIB names, as shown:

```
/snmp/mibs/mibList=SNMPv2-SMI,SNMPv2-TC,INET-ADDRESS-MIB,CISCO-SMI,CISCO-TC,SNMPv2-MIB,RFC
1213-MIB,IANAifType-MIB,IF-MIB,DOCS-IF-MIB,DOCS-IF-EXT-MIB,DOCS-BPI-MIB,CISCO-CABLE-SPECTR
UM-MIB,CISCO-DOCS-EXT-MIB,SNMP-FRAMEWORK-MIB,DOCS-CABLE-DEVICE-MIB,DOCS-QOS-MIB,CISCO-CABL
E-MODEM-MIB,DOCS-IETF-BPI2-MIB,INTEGRATED-SERVICES-MIB,DIFFSERV-DSCP-TC,DIFFSERV-MIB,TCOML
ABS-MIB,PKTC-TCOMLABS-MTA-MIB,PKTC-TCOMLABS-SIG-MIB
```

Configuring DPoE

The DOCSIS Provisioning of Ethernet Passive Optical Network (DPoE) 1.0 is a standard for provisioning EPON access technology using the existing DOCSIS provisioning flow. DPoE network offers IP high speed data services equivalent to DOCSIS networks, where the DPoE network acts like a DOCSIS CMTS. The DPoE system and DPoE Optical Network Unit appear to act like a DOCSIS CM also known as virtual CM(vCM). Prime Cable Provisioning uses the existing DOCSIS device type for DPoE vCM devices. DPoE configuration files contain a mixture of DOCSIS and DPoE-specific TLVs.

To identify a DPoE vCM, refer to the DHCP discover data captured under Request Dictionary displayed under Device Details page. Details similar to the following example show up and if the text in bold appear in page, then it is DPoE vCM.

Example:

```
v-i-vendor-opts = enterprise-id 4491, (oro 1 2)
chaddr = 00:00:00:00:0d:12
relay-agent-info = (circuit-id 1 80:01:03:ef), (remote-id 2 00:00:00:00:0d:12),
(v-i-vendor
-opts 9 enterprise-id 4491, (cmts-capabilities 1 (docsis-version 1 03:00
), (dpoe-system-version 1 01:00), (dpoe-system-pbb 4 10248294639d, 1a9eb
ee4971b, 26d07cd85ab2, 33800cf1abbb, 3b87c25dffbb, 47bd40a08f95, 4fc50b5
3a070, 5768bd554059, 591cf857aea1, 638c2d178f8f, 6d932a665ec9, 74efc6fc0
60b, 7a602d489587)))
relay-agent-circuit-id = 01:04:80:01:03:ef
client-id-created-from-mac-address = 0
dhcp-class-identifier = AIC Echo,docsis3.0:
hlen = 06
giaddr = 4.0.0.1
vendor-encapsulated-options = (device-serial-number 4 000000000d12),
(hardware-version-number 5 v3.2.1
), (software-version-number 6 v1.0.2), (boot-rom-version 7 BOOT1.0), (ve
ndor-oui 8 000000), (vendor-name 10 XEROX CORPORATION), (dpoe-embedded-c
omponents-list 55 ECM)
dhcp-parameter-request-list = {1,3,6,7,12,15,51,54,4,2,67,66}
client-id = ff:00:00:00:00:00:03:00:01:00:00:00:00:0d:12
```

Sample DPoE Configuration file

The sample DPoE configuration files are available under the installed package at location:

- Static file: dpoe_vcm.cm -- /opt/CSCObac/rdu/samples/docsis
- Groovy file: example_dpoe_vcm.groovy -- /opt/CSCObac/rdu/samples/groovy
- Template file: dpoe_vcm.tmpl -- /opt/CSCObac/rdu/templates

Note Prime Cable Provisioning supports provisioning of DPoE vCMs in IPv4 mode only. Also, only downstream of computer devices from a DPoE vCMs is supported.

See DPoE Option Support, page A-22 for the DPoE TLVs.

Differences between DPoE and DOCSIS Provisioning

The provisioning of a DPoE vCM is nearly identical to the provisioning of DOCSIS CM. This allows existing DOCSIS-based back-office systems (such as provisioning servers) to support DPoE vCM provisioning with minimal changes. However, there are minor differences between DPoE vCM and DOCSIS CM provisioning:

- The DPoE specifications do not support IPv6 or PacketCable Voice services. Only IP (HSD) and MEF services are supported.
- The DPoE System (CMTS) supplies additional relay agent DHCP options. The DHCPv4 Relay Agent CMTS capabilities option contains additional sub-options (sub-option 2: DPoE System Version Number and sub-option 4: DPoE System DHCPv4 PBB service option).
- The DPoE vCM does not request ToD. The DPoE System (CMTS) supplies the time reference directly to the vCM.
- The DPoE vCM uses the same DHCPv4 option-60 value as DOCSIS 3.0 CM (i.e., docsis3.0). By itself, the DHCPv4 option-60 value is not sufficient to identify the device as a DPoE vCM.
- The DPoE vCM uses a new eSAFE DHCP option-43 sub-option 55 for specifying the list of eSAFE devices behind the eCM.
- The DPoE vCM supports new configuration file TLVs not supported by DOCSIS 3.0 MULPI (i.e., TLVs [22/23].14, [22/23].14.1, [22/23].14.2, [22/23].14.5, [22/23].14.6, [22/23].15, [22/23].15.1, [22/23].15.2).
- The DPoE vCM does not require or support all configuration file TLVs required by DOCSIS 3.0 MULPI. When the DPoE system encounters a TLV that is not supported, then the DPoE system ignores the TLV and allow the DPoE ONU to register normally.

DPoE Workflow

DPoE workflow is same as DOCSIS workflow. See DOCSIS Workflow, page 7-1 for details.

Configuring CableHome

This section describes the activities that must be performed to ensure a satisfactory CableHome deployment. There are two versions of the CableHome technology: secure (SNMP) and non-secure (DHCP). This chapter deals exclusively with the non-secure version.

This section assumes that you are familiar with the contents of the CableHome Specification CH-SP-CH1.0-I05-030801.

CableHome Workflow

To successfully configure Prime Cable Provisioning for provisioning using the non-secure CableHome technology, you must perform the tasks described in Chapter 6, “Configuring Prime Cable Provisioning Components”, in addition to those described in this section.

Table 7-6 describes the tasks you must perform on Prime Cable Provisioning to support CableHome.

Table 7-6 CableHome Workflow

	Task	Refer to...
Step 1	Configure the RDU	
	a. Configure provisioned DHCP Criteria. Add all the DHCP Criteria that will be used by the non-secure CableHome devices that you will provision.	Configuring DHCP Criteria, page 11-12
	b. Configure provisioned Class of Service. Add the Class of Service that may be used by any provisioned non-secure CableHome device.	Configuring Class of Service, page 11-1
	c. Configure the promiscuous mode of operation.	System Defaults, page 11-10
Step 2	Configure the DPE	
Step 3	Configure Network Registrar	
	Configure the client classes/scope-selection tags to match those added for the provisioned non-secure CableHome DHCP Criteria.	*Cisco Prime Network Registrar 8.1 User Guide*

Configuring Prime Network Registrar

This section describes how to configure Prime Network Registrar, the cable modem configuration system (CMTS).

Step 1 Create selection tags for provisioned and unprovisioned WAN-MAN and also for provisioned WAN-Data.

Configure unprovisioned and provisioned client classes and scopes for cable modems, as specified in *Cisco Prime Network Registrar 8.1 User Guide*.

Step 2 Configure unprovisioned and provisioned client classes and scopes for WAN-MAN.

Step 3 Configure provisioned client classes and scopes for WAN-Data.

Step 4 Add routes to all the subnets.

Configuring RDU

To configure CableHome support on the RDU, perform these configurations:

- Configuring CableHome WAN-MAN, page 7-22
- Configuring CableHome WAN-Data, page 7-22

Configuring CableHome WAN-MAN

1. Create a DHCP Criteria for the provisioned WAN-MAN. To do this, set the client class to a client-class name that is configured in the Network Registrar CableHome WAN-MAN.
2. Create a Class of Service for the provisioned WAN-MAN.
   - Set the */cos/chWanMan/file* to a CableHome configuration file appropriate for the Class of Service.
   - Set the */chWanMan/firewall/file* to the desired firewall configuration file.

Configuring CableHome WAN-Data

Configure these WAN-Data parameters whenever you want portal services to obtain the WAN-Data IP addresses:

1. Create DHCP Criteria for WAN-Data.
2. Create Class of Service for WAN-Data.

Configuring DPE

To configure the DPE to support the CableHome technology:

Step 1 Open the CableHome device provisioning WAN-MAN config file and verify that DHCP Option 60 is set to either CableHome1.0 or CableHome1.1. Some manufacturers use a proprietary MIB object to instruct a device to behave as a pure cable modem, a non-CableHome router, or a CableHome router. The device appears as a Computer whenever the device DHCP packet does not contain CableHome1.0 or CableHome1.1 in the DHCP Option 60.

Step 2 If you want the portal services to obtain IP addresses for WAN-Data:

- Ensure that the WAN-MAN configuration file contains TLV 28 that sets cabhCdpWanDataIpAddrCount to a value that is greater than 0.
- In the cable modem configuration file, set the maximum number of devices to include the number of WAN-Data IP addresses.

Step 3 To enable self-provisioning when the CableHome device boots:

- In the *unprov-wan-man.cfg* portal services configuration file, set the portal services in the passthrough mode.
- In the cable modem configuration file, set the maximum number of devices to at least 2 to allow provisioning of the WAN-MAN and a computer. The computer can directly access sign-up web pages to be self-provisioned.

CHAPTER 8

Configuring Secure Communication

Prime Cable Provisioning secures all TCP based interactions through the use of Secure Socket Layer (SSL) protocol. SSL is a standard based protocol that enables secure communication.
Prime Cable Provisioning supports both SSL 3.0 and TLS 1.0 with TLS 1.0 being the default.

SSL protects the following interactions:

- Clients using the Prime Cable Provisioning API to interact with the RDU.
- DPE and RDU interactions.
- CPNR-EP and RDU interactions.
- Client interaction with the new web services interface.

Certain SSL property configuration can be done using the changeSSLProperties.sh tool. For details about the tool, see Chapter 28, "Using changeSSLProperties.sh".

Key and Certificate Management

The RDU stores the certificates that the SSL protocol requires for authentication in a keystore. This keystore is a file that stores cryptographic keys and certificates. The keystore is generated with the help of a tool available with the JRE called the keytool. The generated certificates are validated for SSL communication before establishing the SSL socket. The keystore files are stored under BPR_HOME/lib/security folder by default. The keystore location is configurable in Prime Cable Provisioning. The bprAgent must be restarted after changing the default keystore location.

You can use Prime Cable Provisioning to configure the server certificate keystore and the cacerts keystore by using the keytool utility. The keytool is a key and certificate-management utility, which you use to administer the certificates on the RDU server and the clients. The keytool utility resides in the Prime Cable Provisioning default installation directory, at BPR_HOME/jre/bin/keytool.

Note You must execute the keytool utility bundled with this Prime Cable Provisioning version, because the keystore file format varies between keytool releases.

There are two keystores on the RDU server. The keystores are the cacerts keystore and the server certificates keystore.

- The cacerts keystore contains public key certificates that the components trust for authenticating the server certificates.
- The server certificates keystore contains the private key and the associated certificate chain for the server-side certificate, which is used to authenticate the clients.

All component SSL services share a single cacerts keystore. This keystore can contain any number of signing authority certificates. The name of the cacerts keystore is fixed, and it must always reside in BPR_HOME/jre/lib/security directory. Prime Cable Provisioning ships with a default cacerts keystore, which can be manipulated by adding and removing signing authority certificates.

Configuring SSL Post Installation

While installing Prime Cable Provisioning, you are asked if you want to configure SSL (secure-mode communication). If you choose yes, the SSL mode is enabled and you are prompted to enter the RDU certificate details. In case you choose no or for some reason, SSL does not get enabled, no RDU certificate is created.

This section explains how to configure SSL in case you have not done it during the installation.

Configuring SSL on RDU

You can use the -ssl option to enable/disable SSL on RDU.

To enable SSL on the RDU server:

Step 1 Using the keytool utility changeSSLProperties.sh, located at BPR_Home/bin, generate the keystore for RDU. This creates a server certificate keystore, which contains the private key and the associated client public key certificates.

Note Before importing the certificate into the RDU certificate keystore, you must import the public certificate of the signing authority into the cacerts keystore.

Step 2 Set the private key password for RDU by running the following command:

```
./changeSSLProperties -ssl -cpkp
```

Step 3 Stop the RDU.

```
/etc/init.d/bprAgent stop rdu
```

Step 4 Enable SSL on RDU by running the following command:

```
./changeSSLProperties -ssl rdu enable
```

Step 5 Start the RDU.

```
/etc/init.d/bprAgent start rdu
```

Step 6 Configure the clients to use the new server certificate keystore from the RDU.

Step 7 To ensure that the changes you make to the keystore take effect, you must restart the clients by executing the command **/etc/init.d/bprAgent restart rdu** from the watchdog agent command line (see Using Prime Cable Provisioning Process Watchdog from CLI, page 23-2).

Configuring SSL on DPE

This section describes how to configure SSL on the DPE services.

You can configure DPE security options from the DPE CLI. For more information, see the *Cisco Prime Cable Provisioning, 5.0 DPE CLI Reference*.

To enable SSL on DPE:

Step 1 Ensure you have imported the rootCA.crt file to BPR_HOME/jre/lib/security/cacerts trust store.

Step 2 Run the command dpe rdu-server <host name> <port number> <secure> where the value for secure must be set to true.

For example:

```
dpe rdu-server bac-rhel5-vm80 49188 true
```

In the above command, you must enter the correct secure port number that RDU listens to. By default, RDU listens to the 49188 port.

Step 3 Restart the DPE by using the **dpe reload** command to ensure that the changes take effect.

Configuring SSL on API Client

You can configure SSL on any API client by using the api.properties file.

To enable SSL on an API client:

Step 1 Import the signed rootCA certificate into the JRE of the API client.

Step 2 In the api.properties file, for the property /server/rdu/secure/enabled, set the value to true.

For example:

```
/server/rdu/secure/enabled=true
```

Step 3 Provide the secure RDU server details using the property /rdu/secure/servers. The value for the property is a comma separated list of host and port number as shown below

For example:

```
/rdu/secure/servers=<hostname:port, hostname2:portnum2>
/rdu/secure/servers=bac-rhel5-vm80:49188,64.103.255.6:49188
```

Step 4 Restart the API client to ensure that the changes take effect.

Configuring SSL on Cisco Prime Network Registrar Extension Point

You can use the changeNRProperties.sh tool to set the secure communication on Cisco Prime Network Registrar EP.

To enable SSL on CPNR-EP:

Step 1 Copy the rootCA.pem file to the BPR_HOME/lib/security folder.

Step 2 Run the following command:

```
./changeNRProperties.sh -ssl enable
```

Step 3 Set the secure port:

```
./changeNRProperties.sh -p 49188
```

Step 4 Restart the DHCP server.

Overriding PACE Connection Settings using api.properties

Prime Cable Provisioning allows you to override your existing PACE connection signatures for API clients without having to recompile your existing Java sources. This is achieved by defining additional properties in your api.properties. Properties such as port information and switching between secure and non-secure connections can be controlled via your api.properties.

To enable secure communication between an API client and it's RDU:

Step 1 Set /server/rdu/secure/enabled property to true. Before you enable the secure communication, ensure that you have configured your certificates properly.

For example:

```
/server/rdu/secure/enabled=true
```

Step 2 Define a property /rdu/secure/servers, with the RDU hostname and secure port details. In case your API client talks to multiple RDUs then define a comma separated list of all your secure host names and port details.

For example:

```
/rdu/secure/servers=<hostname1:port1>,<hostname2:port2>,<hostname3:port3>
```

If you change the default non-secure port number(49187) on the RDU, for the change to take effect, you need to set the property /rdu/unsecure/servers with the correct port information. You need not modify the Java sources.

For example:

```
/rdu/unsecure/servers=<hostname1:port1>,<hostname2:port2>,<hostname3:port3>, with your RDU
hostnames and ports
```

At any given time, an API client can communicate to an RDU either in secure or non-secure mode but not both. If it attempts to communicate to the RDU in a mode different from the existing one, an exception is raised. In order to create a connection in the other mode, you must release all live PACE connections of the existing mode. However, the same API client can communicate with other RDUs in different modes.

Note If you have defined the same host name for both secure and non-secure communication, then the secure communication takes precedence over non-secure communication.

Signing a Certificate

While installing Prime Cable Provisioning, you can either self-sign the certificate of opt to get it signed from an external signing authority. In case of a self-signed certificate, the certificate encrypts your communication interactions. Since a self-signed certificate is not signed by a signing authority, web browsers flag the certificate as potentially risky. To avoid this you must import your signed certificate as a replacement.

Signing a Certificate Through an External Authority

This sections explains how to get the certificate signed through an external authority.

To sign a server certificate:

Step 1 Generate a new private key for the RDU using keytool in case you have not created it while installing the RDU server. See Generating Private Key for a New RDU Certificate, page 8-9, for details.

Step 2 Generate a certificate-signing request (CSR). See Generating a Certificate-Signing Request, page 8-10, for details.

Step 3 Request a public certificate from the signing authority by using CSR.

Step 4 Load the public key of the signing authority into the cacerts keystore. See Importing Signing Authority Certificate into Cacerts Keystore, page 8-11, for details.

Step 5 Load the signed server certificate into the server keystore. See Importing Signed Certificate into Server Certificate Keystore, page 8-12, for details.

Step 6 Restart the RDU by using the command **/etc/init.d/bprAgent restart rdu** from the watchdog agent command line (see Using Prime Cable Provisioning Process Watchdog from CLI, page 23-2).

Note To enable the use of client certificates for server authentication, ensure that the public certificate of the signing authority for server certificates is loaded into the cacerts keystore. Follow the procedure described in Importing Signing Authority Certificate into Cacerts Keystore, page 8-11, for details.

Self-signing a Certificate

Use the changeSSLProperties.sh tool to create a self-signed certificate. For the exact usage of the tool, see Using changeSSLProperties.sh, page 28-16.

On the RDU server

Step 1 Create a RDU keypair using the -gk option. Keep note of the rducert alias certificate password.

Step 2 Export the self-signed certificates using -exp option.

Step 3 Set the private key password for the RDU using the -cpkp option. Make use of the certificate password used while creating certificate using -gk option.

Step 4 Stop the RDU by executing /etc/init.d/bprAgent stop rdu

Step 5 Enable the SSL mode for the RDU server using -ssl rdu enable option.

Step 6 Start the RDU by executing /etc/init.d/bprAgent start rdu

On DPE server

Step 7 Copy the RDU rootCA.crt file that is being generated in the previous step and is located under the BPR_HOME/lib/security directory to a desired location in the DPE server.

Step 8 Use the -imp option to import the certificate to the truststore.

Step 9 Enable RDU SSL communication on DPE by running the command, *dpe rdu-server <hostname><port> true*.

Step 10 For the changes to take effect, reload the DPE or PWS server.

On PWS server

Step 11 Copy the RDU rootCA.crt file that is being generated in the previous step and is located under the BPR_HOME/lib/security directory to a desired location in the PWS server.

Step 12 Use the -imp option to import the certificate to the truststore.

Step 13 Add the secure host details by executing changeSSLproperties.sh -csp api. This displays the host and port details.

Step 14 Add or modify the list with the correct host name and port details.

Step 15 For the changes to take effect, reload the PWS server.

On CNR-EP server

Step 16 Copy the RDU rootCA.pem file that is being generated in the previous step and is located under the BPR_HOME/lib/security directory to the BPR_HOME/lib/security directory of the CNR-EP server.

Step 17 Enable RDU SSL communication on CNR-EP by running the command changeNRProperties.sh -ssl enable.

Step 18 Set the secure port by executing changeNRProperties.sh -p 49188.

Step 19 For the changes to take effect, reload the DHCP server by executing the command: CNR_HOME/local/usrbin/nrcmd dhcp reload.

Importing an Existing Signed Server Certificate

If you already have the signed server certificate and you want to load it into the keystore, you must know the private key that is associated with the certificate. In this case, instead of following the procedure described above, follow the steps outlined in this section. Use the PCKS#12 file format, which combines both the private key and the signed certificate. You can load this file into a keystore by using the **keystore import-pkcs12** command.

To configure a server certificate with an existing signed server certificate:

Step 1 Load the existing private key and certificates into a RDU-compatible file, used in authenticating the RDU to SSL clients, by using the **keystore import-pkcs12** command.

When using this command, the syntax is:

```
# ./keystore import-pkcs12 keystore-filename pkcs12-filename keystore-password
key-password
export-password export-key-password
```

- *keystore-filename*—Identifies the keystore file to create. If it already exists, it will be overwritten.

Note Remember to specify the full path of the keystore file.

- *pkcs12-filename*—Identifies the PKCS#12 file from which you intend to import the key and certificate.
- *keystore-password*—Identifies the private key password and the keystore password that you used when you created your keystore file. This password must be between 6 and 30 characters.
- *key-password*—Identifies the password used to access keys within RDU keystore. This password must be between 6 and 30 characters.
- *export-password*—Identifies the password used to decrypt the key in the PKCS#12 file. The export password must be between 6 and 30 characters.
- *export-key-password*—Identifies the password used to access keys within the PKCS#12 keystore. This password must be between 6 and 30 characters.

For example:

```
# ./keystore import-pkcs12 example.keystore example.pkcs12 changeme changeme changeme
changeme
% Reading alias [1]

% Reading alias [1]: key with format [PKCS8] algorithm [RSA]

% Reading alias [1]: cert type [X.509]

% Created JKS keystore: example.keystore

% OK
```

Step 2 Copy the new keystore file into the RDU BPR_HOME/dpe/conf directory.

Step 3 At the CLI, configure one of the RDU services to use the new keystore. See Configuring SSL Post Installation, page 8-2, for details.

Step 4 Restart the RDU the command **/etc/init.d/bprAgent restart rdu** from the watchdog agent command line (see Using Prime Cable Provisioning Process Watchdog from CLI, page 23-2).

Using Keytool Commands

The keytool utility uses command arguments to configure the keystore. Table 8-1 lists the keytool commands and their descriptions.

Table 8-1 Keytool Commands

-alias *alias*	Identifies the identity assigned to a keystore entry, which stores the certificate chain and the private key. Subsequent keytool commands must use the same alias to refer to the entity.
-dname *dname*	Identifies the X.500 Distinguished Names used to identify entities, such as those that the subject and the issuer named.
-file *csr_file*	Identifies the CSR file to be exported.
-file *cert_file*	Identifies the file from which the certificate is to be read.
-keyalg *keyalg*	Identifies the algorithm to be used for key-pair creation. The values are DSA (default) and RSA.
-keysize *keysize*	Specifies a keysize, whose values must be in multiples of 64 bits.
-keypass *keypass*	Identifies the password assigned to a key pair.
-keystore *keystore*	Customizes the name and location of a keystore.
-noprompt	Specifies that no prompts are to be issued during an import operation.
-provider *provider_class_name*	Identifies the name of the cryptographic service provider's master class file when the service provider is not listed in the security properties file.
-rfc	Specifies that the output of the MD5 fingerprint of a certificate, which appears in printable-encoding format.
-storepass *storepass*	Identifies the password assigned to a keystore.
-sigalg *sigalg*	Specifies the algorithm to be used to sign the certificate.
-storetype *storetype*	Identifies the type assigned to a keystore or an entry into a keystore.
-trustcacerts	Specifies that additional certificates are considered for the chain of trust.
-v	Specifies that the output of the MD5 fingerprint of a certificate is printed in human-readable format.
-validity *valDays*	Identifies an expiration period. The default is 90 days.

Note For additional information on keytool and general certificate-management concepts, refer to Oracle documentation.

Generating Private Key for a New RDU Certificate

The **keytool -genkey** command generates a key pair (a public key and an associated private key), and wraps the public key into an X.509 self-signed certificate, which is stored as a single-element certificate chain. This certificate chain and the private key are stored in a new keystore entry identified by alias.

Note The purpose of an *alias* is to uniquely identify a key pair within the keystore, in case you have multiple key pairs. In the context of Prime Cable Provisioning, the *alias* used for the RDU key is critical. Prime Cable Provisioning uses *rducert* as its default *alias*. In case you have changed the *alias*, add or update the /secure/rdu/certificateAlias property with the new *alias* in the rdu.properties file.

Note If you are directly using the keytool to generate the RDU keypair, you must use the *changeSSLProperties.sh -cpkp* command to update the private key password that you provided while creating the keypair to the rdu.properties file.

Example 8-1 Keytool -genkey

The following example uses *.keystore* as the name of the keystore file.

```
# ./keytool -genkey -alias rducert -storetype JCEKS -validity 730 -keyalg RSA -keystore
/opt/CSCObac/lib/security/.keystore

Enter keystore password: changeit
Re-enter new password: changeit
What is your first and last name?
  [Unknown]:  BAC Testing
What is the name of your organizational unit?
  [Unknown]:  NMTG
What is the name of your organization?
  [Unknown]:  Cisco Systems Inc.
What is the name of your City or Locality?
  [Unknown]:  Bangalore
What is the name of your State or Province?
  [Unknown]:  KAR
What is the two-letter country code for this unit?
  [Unknown]:  IN
Is CN=BAC Testing, OU=NMTG, O=Cisco Systems Inc., L=Bangalore, ST=KAR, C=IN correct?
  [no]:  yes
Enter key password for <rducert>
        (RETURN if same as keystore password):
```

Displaying Self-Signed Certificate

The **keytool -list** argument displays the contents of the keystore entry identified by alias. If you do not specify an alias, the entire contents of the keystore appear.

If you combine **-list** with **-v**, the certificate chain associated with the alias appears. The following **keytool -list** sample output displays the keystore containing a single self-signed certificate.

Example 8-2 Keytool -list

```
# ./keytool -list -v -storetype JCEKS -keystore /opt/CSCObac/lib/security/.keystore
Enter keystore password: changeit
Keystore type: JCEKS
Keystore provider: SunJCE
Your keystore contains 1 entry
Alias name: rducert
Creation date: Aug 21, 2012
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=BAC Testing, OU=NMTG, O=Cisco Systems Inc., L=Bangalore, ST=KAR, C=IN
Issuer: CN=BAC Testing, OU=NMTG, O=Cisco Systems Inc., L=Bangalore, ST=KAR, C=IN
Serial number: 50331e4f
Valid from: Tue Aug 21 11:06:15 IST 2012 until: Mon Nov 19 11:06:15 IST 2012
Certificate fingerprints:
         MD5:  8F:03:43:33:51:9B:DB:C5:0E:27:B5:B4:A1:FE:97:83
         SHA1: A1:FB:63:CD:58:44:A0:CA:1A:A2:41:9B:09:C1:CC:5E:EB:66:B9:96
         Signature algorithm name: SHA1withRSA
         Version: 3
```

Generating a Certificate-Signing Request

At this point in the procedure, the keystore contains a private key and a X.509 self-signed certificate. If the RDU tries to respond with this certificate to a client's initial handshake, the client will reject the certificate with a TLS alert `bad CA`, indicating that the certificate authority that the client trusted did not sign the certificate. Therefore, the signing authority that the client trusts must sign the certificate.

Note To support SSL, the clients must have a list of preconfigured public certificates of signing authorities that they trust.

The **keytool -certreq** command parameter generates a certificate-signing request (CSR). This command generates the CSR in the industry standard PKCS#10 format.

Example 8-3 Keytool -certreq

The following example uses a keystore with a pre-existing self-signed certificate under **alias** *rducert* to generate a certificate-signing request and output the request into the `train-1.csr` file.

```
# ./keytool -alias rducert -certreq -file /opt/CSCObac/lib/security/rducert.csr -storetype 
JCEKS -keystore /opt/CSCObac/lib/security/.keystore
Enter keystore password: changeit
```

The next step is to submit the CSR file to your signing authority. Your signing authority or your administrator, who is in possession of the private key for the signing authority, will generate a signed certificate based on this request. From the administrator, you must also obtain the public certificate of the signing authority.

Verifying the Signed Certificate

After you have received the signed certificate, use the **keytool -printcert** command to verify if the self-signed certificate is in the correct file format and uses the correct owner and issuer fields. The command reads the certificate from the **-file** *cert_file* parameter, and prints its contents in a human-readable format.

Example 8-4 Keytool -printcert

The *rootCA.crt* file in this example identifies the signed certificate that the administrator provides.

```
# ./keytool -printcert -file rootCA.crt
Owner: CN=BAC Testing, OU=NMTG, O=Cisco Systems Inc., L=Bangalore, ST=KAR, C=IN
Issuer: CN=BAC Testing, OU=NMTG, O=Cisco Systems Inc., L=Bangalore, ST=KAR, C=IN
Serial number: 50331e4f
Valid from: Tue Aug 21 11:06:15 IST 2012 until: Mon Nov 19 11:06:15 IST 2012
Certificate fingerprints:
         MD5:  8F:03:43:33:51:9B:DB:C5:0E:27:B5:B4:A1:FE:97:83
         SHA1: A1:FB:63:CD:58:44:A0:CA:1A:A2:41:9B:09:C1:CC:5E:EB:66:B9:96
         Signature algorithm name: SHA1withRSA
         Version: 3
```

Note The keytool can print X.509 v1, v2, and v3 certificates, and PKCS#7-formatted certificate chains comprising certificates of that type. The data to be printed must be provided in binary-encoding format, or in printable-encoding format (also known as Base64 encoding) as defined by the RFC 1421.

Importing Signing Authority Certificate into Cacerts Keystore

Before importing the certificate into the server certificate keystore, you must import the public certificate of the signing authority into the cacerts keystore; because when a certificate is being imported into the keystore, the keytool checks if a chain of trust can be established between the certificate and its signing authority. If a chain of trust cannot be established, an error message appears.

Note The cacerts file bundled with Prime Cable Provisioning ships with several root certificate common third-party signing authorities. You can manage the cacerts keystore by using the keytool utility. The default cacerts keystore password is **changeit**. The cacerts database file resides in the *BPR_HOME/jre/lib/security* directory.
The cacerts keystore does not need to be copied anywhere. The client will use the new keystore as soon as it is restarted.

Example 8-5 Keytool -import Signing Authority Certificate

```
# ./keytool -import -alias rducert-file rootCA.crt -keystore
/opt/CSCObpr/jre/lib/security/cacerts
Enter keystore password:  changeit
Owner: CN=BAC Testing, OU=NMTG, O=Cisco Systems Inc., L=Bangalore, ST=KAR, C=IN
Issuer: CN=BAC Testing, OU=NMTG, O=Cisco Systems Inc., L=Bangalore, ST=KAR, C=IN
Serial number: 50331e4f
Valid from: Tue Aug 21 11:06:15 IST 2012 until: Mon Nov 19 11:06:15 IST 2012
Certificate fingerprints:
         MD5:  8F:03:43:33:51:9B:DB:C5:0E:27:B5:B4:A1:FE:97:83
         SHA1: A1:FB:63:CD:58:44:A0:CA:1A:A2:41:9B:09:C1:CC:5E:EB:66:B9:96
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

Note The keytool can import X.509 v1, v2, and v3 certificates, and PKCS#7-formatted certificate chains comprising certificates of that type. The data to be imported must be provided in binary-encoding format, or in printable-encoding format (also known as Base64 encoding) as defined by the RFC 1421.

Importing Signed Certificate into Server Certificate Keystore

Once you import the public certificate of the signing authority into the cacerts keystore, you must import the signed server certificate into the RDU server certificate keystore. You will already have a keystore with private key and corresponding self-signed certificate (public key).

By importing the certificate reply (signed certificate), the keystore is modified to associate the signed certificate with the existing private key in the server certificate keystore.

When importing the certificate reply into the keystore, you must use the **-trustcacerts** flag with the **-import** command for certificates in the *cacerts* file to be used to establish chains of trust with the certificate reply in the subject's keystore.

Example 8-6 Keytool -import (Signed Server Certificate)

```
# ./keytool -import -trustcacerts -file rducert.crt -keystore
/opt/CSCObac/lib/security/.keystore -alias rducert -storetype JCEKS
Enter key password: changeme
Enter keystore password:  changeme
Certificate reply was installed in keystore
Certificate was added to keystore
```

After you import the signed server certificate into the RDU server certificate keystore, use the keytool **-printcert** command to verify the keystore contents, as outlined in Verifying the Signed Certificate, page 8-10. The **-printcert** output should now show the issuer to be the signing certificate authority, and that a chain of trust has been established using the signing authority with the root trusted certificate.

Troubleshooting SSL

Prime Cable Provisioning supports SSL logging for RDU, DPE, CNR extensions and API clients. This helps in handling connection issues during SSL communication.

The SSL communication logs are updated in the following log files:

- rdu.log - You must enable debug logging with secure messaging on for SSL log to appear in rdu.log.
- dpe.log - You must enable secure messaging via DPE CLI to enable DPE SSL logging.
- name_dhcp_1_log - You must enable trace level 4 for the DHCP process for debugging DHCP SSL issues.
- The client API log can be configured and set to any name based on the configuration.

CHAPTER 9

Configuring IPv6

Prime Cable Provisioning supports the CableLabs DOCSIS standard: DOCSIS 3.0. The DOCSIS 3.0 standard introduces key new features that build on previous DOCSIS standards. These features include:

- Provisioning of IPv6 devices, which include:
  - DOCSIS-compliant cable modems and CMTS
  - Computers
  - Any STB compliant with CableLabs OpenCable Application Platform
  - Variants of eSAFE (embedded Service/Application Functional Entities) devices such as eCM.

  Cisco Prime Cable Provisioning provides services required to provision IPv6 devices, such as IPv6 support for TFTP and ToD services. It also processes configuration files for IPv6 devices.
- Expanded addressability

  The main benefit of IPv6 is its expanded addressing capability. IPv6 addresses increase the address space from 32 to 128 bits, providing for a virtually unlimited number of networks and systems.
- IPv6 provisioning and management of cable modems. This provisioning flow includes:
  - Supported IP modes— provisions DOCSIS cable modems in the IPv4, IPv6. Supports IPv4, IPv6 and dual stack mode for device type computer. See Single Stack versus Dual Stack for more information about dual stack.

  Note Cable modems can forward IPv4 and IPv6 traffic regardless of the IP provisioning mode.

  - DHCPv6—The DOCSIS provisioning flow specifies the use of DHCP for IPv6, also known as DHCPv6. For details on the DHCPv6 provisioning flow in DOCSIS, see DOCSIS DHCPv6 Workflow, page 14-4.

Before you enable to provision devices in IPv6, ensure that you enable IPv6 on your system. To enable your machine to support IPv6:

For Solaris:

Step 1 Log in as *root*, and enter these commands:

```
# ifconfig intf inet6 plumb up
```

where *intf* identifies the interface on which you want to enable IPv6.

Note Ensure that you plumb in the loopback interface in addition to the physical Ethernet interface. For example:

```
# ifconfig bge0 inet6 plumb up
# ifconfig lo0 inet6 plumb up
```

Step 2 Run these commands:

```
# /usr/lib/inet/in.ndpd
# touch /etc/hostname6.intf
```

where *intf* identifies the interface on which you want to enable IPv6.

For Linux:

Step 1 Log in as *root*.

Step 2 Modify the configuration file of the kernel module loader, /etc/modules.conf or /etc/conf.modules, to include the following statement:

```
# alias net-pf-10 ipv6  # automatically load IPv6 module on demand
```

Step 3 Test the IPv6 configuration using the following command:

```
# ifconfig
```

If inet6 addr: 2001:e30:1400:1:208:c7ff:fecf:9d0a/64 Scope:Global string is displayed in the result, IPv6 is successfully enabled.

You must enable IPv6 provisioning group capabilities in Prime Cable Provisioning to provision IPv6 devices. To enable the provisioning group capabilities, you need to enable the corresponding IPv6 interface and services in the DPE. See the following sections of *Cisco Prime Cable Provisioning 5.0 DPE CLI Guide* for additional information.

- *interface ip provisioning* to configure IPv6 interface
- *service tftp* to enable IPv6 TFTP service in DPE
- *service tod* to enable IPv6 TOD service in DPE

Other concepts related to IPv6:

- IPv6 Addressing, page 9-3
- Single Stack versus Dual Stack, page 9-3
- DHCP Options for IPv6, page 9-4
- Configuration Workflows for IPv6, page 9-4

IPv6 Addressing

IPv6 addresses are 128 bits long and are represented as a series of 16-bit hexadecimal fields separated by colons (:). The A, B, C, D, E, and F in hexadecimal are case insensitive. For example:

```
2031:0000:130f:0000:0000:09c0:876a:130b
```

A few shortcuts to this addressing are:

- Leading zeros in a field are optional, so that 09c0 can be written 9c0, and 0000 as 0.
- Successive fields of zeros (any number of them) are represented as ::, but only once in an address (because if used more than once, the address parser has no way of identifying the size of each block of zeros). So, the previous address can be written:

  ```
  2031:0:130f::09c0:876a:130b
  ```

The use of the double-colon abbreviation makes many addresses small, for example ff01:0:0:0:0:0:1 becomes ff01::1.

Link-local addresses have a scope limited to the link, and use the prefix fe80::/10. Loopback addresses have the address ::1. Multicast addresses are identified by the prefix ff00::/8 (there are no broadcast addresses in IPv6).

The IPv4-compatible addresses in IPv6 are the IPv4 decimal quad addresses prefixed by ::. For example, an IPv4 address that would be interpreted as ::c0a8:1e01 can be written as ::192.168.30.1.

Single Stack versus Dual Stack

RFC 4213 defines dual stack as a technique to provide complete support for both Internet protocols—IPv4 and IPv6—in hosts and routers. Any network stack that supports both IPv4 and IPv6 is called a dual stack, and a host implementing a dual stack is called a dual-stack host.

Prime Cable Provisioning provisions cable modems in the following IP modes:

- IPv4 only—In this mode, the cable modem requests a DHCPv4 server for an IPv4 address and related operational parameters.
- IPv6 only—In this mode, the cable modem requests a DHCPv6 server for an IPv6 address and related operational parameters. The modem uses the IPv6 address to obtain the current time-of-day and a configuration file.
- Dual stack—In this mode, the cable modem acquires both IPv6 and IPv4 addresses and parameters through DHCPv6 and DHCPv4 almost simultaneously, prioritizing the use of the IPv6 address to acquire time-of-day and a configuration file. Dual stack is supported only for computers.

Note Prime Cable Provisioning saves discovered data only for the most recent IP mode that a cable modem is provisioned in. So if a device boots in DHCPv4, then in DHCPv6, only DHCPv6 data is discovered and stored.

While provisioning in the IPv4 and IPv6 modes, the cable modem operates with only one IP address type (v4 or v6) at any given time. For this reason, the IPv4 and IPv6 modes of provisioning are called single-stack modes.

In the dual-stack mode, you can manage the cable modem via IPv4 and IPv6 addresses simultaneously. In this mode, the modem acquires a second IP address after it is operational. Using this feature, you can provide streamlined migration from IPv4 to IPv6 in DOCSIS networks.

Note Prime Cable Provisioning does not support CM dual stack mode but the CPE dual stack mode is fully supported if the CPE is a computer. The Customer Premises Equipment (CPE) is the device which is connected to the Cable Modem (CM), which in turn connected to the CMTS through the HFC network.

DHCP Options for IPv6

The DOCSIS 3.0 standard defines several new options for DHCPv4 and DHCPv6. DHCPv6 options do not use any DHCPv4 options; they are unique and separate. For the list of DHCPv6 options that supports, see the *Cisco Prime Network Registrar 8.1 User Guide*.

Configuration Workflows for IPv6

Configuring Prime Cable Provisioning to support IPv6 involves two distinct workflows:

- Configuring the DPEs in the provisioning group. See Table 6-6.
- Configuring the Cisco Prime Network Registrar servers in your network. See Table 6-7.

CHAPTER 10

Configuring Syslog Utility to Receive Alerts

This chapter explains how to configure the syslog daemon. In case of a local data server, you can configure the syslog utility on any Prime Cable Provisioning component server to receive alerts from the system. For receiving the syslogs in a centralized server from all the Prime Cable Provisioning components, you can configure the syslog daemon either on any Prime Cable Provisioning component such as RDU, PWS, DPE, CPNR and KDC or on a separate server. These component servers are referred to as Prime Cable Provisioning server in this chapter.

Note Configuring the syslog file is an optional task.

Syslog is a client-server protocol that manages the logging of information on UNIX.
Prime Cable Provisioning generates alerts through the syslog service. Prime Cable Provisioning syslog alerts are not a logging service; they notify that a problem exists, but do not necessarily define the specific cause of the problem.

The information related to the problem resides in the appropriate Prime Cable Provisioning log files, rdu.log and dpe.log. If you choose to configure the syslog file, syslog alerts are directed to a separate log file.

Configuring Syslogs on a Local Server

To configure the syslog utility on a local Prime Cable Provisioning server (Solaris and Linux):

Step 1 Log in as *root* on the server.

Step 2 At the command line, create the log file.

For example:

```
# touch /var/log/bac.log
```

Step 3 Open the /etc/syslog.conf file with a text editor, such as *vi*.

Step 4 Add the following lines to the /etc/syslog.conf file:

```
local6.alert       /var/log/bac.log
local6.info        /var/log/bac.log
```

Note You must insert one or more tabs between the local6:info and */var/log/bac.log* information.

Step 5 Save and close the /etc/syslog.conf file.

Step 6 To force the syslog utility to take the new configuration, at the command line, enter:

```
# ps -ef  | grep syslogd
   root      217   1    0     Nov   26      ?    0:00      /usr/sbin/syslogd

   kill -HUP 217
```

Note The process ID (PID) in this example is 217, but may change when you run **ps -ef | grep syslogd**. Use the correct output from that command as the input to **kill -HUP**.

Syslog is now ready to receive alerts from Prime Cable Provisioning.

Configuring Centralized Solaris Server to Receive Syslogs

On Solaris machines, the LOG_FROM_REMOTE property specifies whether server messages are logged. By default, this property is enabled.

To configure a centralized server to receive syslog alters from a Prime Cable Provisioning server:

Step 1 Log into the server as root.

Step 2 By default the LOG_FROM_REMOTE property is enabled. In case it is not, you can enable it by setting it’s value to true as shown in the following commands.

```
# svccfg -s svc:/system/system-log setprop config/log_from_remote = true
# svcadm refresh svc:/system/system-log
```

Step 3 Create a dummy file.

```
# touch /var/log/messages
```

Step 4 Add the following configuration in /etc/syslog.conf file:

```
local6.info            @loghost
local6.info            /var/log/messages
```

Step 5 Restart the syslog daemon.

```
# svcadm restart system-log
# tail -f /var/log/messages
```

Note Always use Tab while modifying /etc/syslog.conf. Using the space bar shows errors while you restart syslogd.

Configuring a Server to Send Syslog to Centralised Server on Solaris

After you configure syslog daemon on a centralized server, you must configure the Prime Cable Provisioning server to send messages to the centralized server. To do this, edit the /etc/hosts file on the RDU/DPE server as explained below.

Step 1 Determine the IP address and fully qualified host name of the Prime Cable Provisioning server logging host.

Step 2 Log into the server as *root*.

Step 3 To enable the server logging hostname, add the following entry in the /etc/hosts file:

For example;

```
IP-address    fully-qualified-domain-name    hostname    "loghost"
```

The /etc/hosts file has the nickname loghost, for the server.

Step 4 Edit the /etc/syslog.conf file to send the syslog messages to the server.

For example;

```
local6.info    ifdef(`LOGHOST', /var/log/messages, @loghost)
```

Step 5 Restart the syslog daemon to get the Prime Cable Provisioning server logging started.

```
# svcadm restart system-log
```

To test whether the syslog server is receiving the messages, stop the RDU server. The DPE and CPNR servers will send a message indicating the connection failure.

Configuring a Centralized Linux Server to Receive Syslogs

By default, syslog daemon on a centralized server does not expect to receive messages from the Prime Cable Provisioning servers. You must configure the centralized server for the syslog daemon to start listening to these messages.

The syslog daemon checks the /etc/syslog.conf file to determine the expected names and locations of the log files it should create. It also checks the /etc/sysconfig/syslog file to determine the various modes in which it should operate. The syslog daemon will not receive server messages unless the SYSLOGD_OPTIONS variable in this file has a -r included in it as shown below:

```
# Options to syslogd
# -m 0 disables 'MARK' messages.
# -r enables logging from RDU/DPE server machines
# -x disables DNS lookups on messages received with -r
# See syslogd(8) for more details
 SYSLOGD_OPTIONS="-m 0 -r"
# Options to klogd
# -2 prints all kernel oops messages twice; once for klogd to decode, and
#    once for processing with 'ksymoops'
# -x disables all klogd processing of oops messages entirely
# See klogd(8) for more details
KLOGD_OPTIONS="-2"
```

You must restart the syslog daemon for the changes to take effect. The server listens on UDP port 514, which you can verify using one of the following netstat command variations:

– **# netstat -a | grep syslog**

```
udp        0      0 *:syslog               *:*
```

– **# netstat -an | grep 514**

```
udp        0      0 0.0.0.0:514            0.0.0.0:*
```

Configuring a Server to Send Syslog to Centralised Server on Linux

After you configure syslog daemon on the centralized server, you must configure the Prime Cable Provisioning server to send messages to it. To do this, edit the /etc/hosts file on the server.

Step 1 Determine the IP address and fully qualified hostname of the server logging host.

Step 2 Log in as *root* on the server

Step 3 To enable the server logging hostname, add the following entry in the /etc/hosts file:

For example:

```
IP-address    fully-qualified-domain-name    hostname    "loghost"
```

In the example, the /etc/hosts file has a nickname loghost, for the server.

Step 4 Edit the /etc/syslog.conf file to send the syslog messages to the server.

For example:

```
local6.info             @loghost
local6.info             /var/log/messages
```

Step 5 Restart the syslog daemon to start Prime Cable Provisioning server logging.

To test whether the syslog server is receiving the messages, stop the RDU server. The DPE and CPNR servers will send a message indicating the connection failure.

CHAPTER 11

Configuring Prime Cable Provisioning Using Admin UI

This chapter describes the Prime Cable Provisioning configuration tasks that you perform by selecting the options in the Configuration menu:

Configuring Class of Service

Using the Prime Cable Provisioning Admin UI, you can configure the Class of Service offered to your customers. For example, you can associate DOCSIS options with different DOCSIS Class of Service. You use the Prime Cable Provisioning Admin UI to add, modify, or delete any selected Class of Service.

Adding a Class of Service

To add a Class of Service:

Step 1 From the Manage Class of Service page, select the device type from the ClassofService drop-down list.

Step 2 Click **Add**.

Step 3 Enter the name for the new Class of Service.

Step 4 If you want to change the Class of Service, choose it from the Type drop-down list. For example, assume that you want to create a new Class of Service called Gold-Classic for DOCSIS modems. You might enter **Gold-Classic** as the Class of Service Name, and choose **DOCSISModem** from the service type drop-down list.

Step 5 Click **Assign Domain** and select the domain to which the property must belong. Click **Apply** to save the changes.

Step 6 Click **Add Property** and choose a property and enter its corresponding value in the Property Value field. For example, if you choose the property name */cos/docsis/file,* enter **Gold-Classic.cm** in the Property Value field, and continue with the rest of this procedure.

Note When adding a DOCSIS Modem Class of Service, you must specify the */cos/docsis/file* property with the value being the name of a previously added file. This file is used when provisioning a DOCSIS device that has this Class of Service.
Prime Cable Provisioning provides automatic selection of a cable modem configuration file that enables the highest DOCSIS version compatible with the modem. To enable this feature, you must configure the Class of Service with multiple configuration files, one for each DOCSIS level. Use the following properties to allow the selection of a configuration file specific to a DOCSIS version:

- */cos/docsis/file/1.0*—Selects a configuration file specific to DOCSIS 1.0.
- */cos/docsis/file/1.1*—Selects a configuration file specific to DOCSIS 1.1.
- */cos/docsis/file/2.0*—Selects a configuration file specific to DOCSIS 2.0.
- */cos/docsis/file/3.0/ipv4*—Selects a configuration file specific to DOCSIS 3.0 in the IPv4 mode.
- */cos/docsis/file/3.0/ipv6*—Selects a configuration file specific to DOCSIS 3.0 in the IPv6 mode.

When adding a PacketCable Class of Service, you must specify the */cos/packetCableMTA/file* property with the value being the name of a previously added file. This file is used when provisioning a PacketCable device that has this Class of Service.

When adding a CableHome WAN-MAN Class of Service, you must specify the */cos/cableHomeWanMan/file* property with the value being the name of a previously added file. This file is used when provisioning a CableHome WAN-MAN device that has this Class of Service.

Step 7 Click **Save** to add the property to the Class of Service.

Step 8 Click **Submit** to finalize the process.

After submitting the Class of Service, the Manage Class of Service page appears to show the newly added Class of Service for the particular device type.

Modifying a Class of Service

You modify your Class of Service by selecting the various properties and assigning appropriate property values. When creating a Class of Service for the first time you must select all the required properties and assign values to them. If you make a mistake, or your business requirements for a certain Class of Service change, you can either change the property value before submitting your previous changes or delete the Property Name:Property Value pair altogether.

Note Changes to the Class of Service object trigger the Configuration Regeneration Service (CRS) to regenerate configurations for all affected devices and send configurations to the DPEs. The CRS performs this task as a background job.

You can view the status of the CRS from the View RDU Details page.

To modify Class of Service properties, select the Class of Service for the specific device type. Make the necessary changes and click **Submit**.

Each property added to a Class of Service appears when you click **Submit**. After doing so, a confirmation page appears to regenerate the configurations for the devices with the selected Class of Service.

Deleting a Class of Service

You can delete any existing Class of Service, but before you attempt to do so, ensure that no devices are associated with that Class of Service. To delete a Class of Service, select the Class of Service for the specific device type that you want to delete and click **Delete**.

Tip When large numbers of devices associated with a Class of Service need to be deleted, use the Prime Cable Provisioning application programming interface (API) to write a program to iterate through these devices to reassign another Class of Service to the devices.

Note You cannot delete a Class of Service if it is designated as the default Class of Service or if devices are associated with it. Therefore, you cannot delete the **unprovisioned-docsis** Class of Service object.

Configuring Custom Properties

Custom properties let you specify additional customizable device information to be stored in the RDU database. To configure custom properties, click **Configuration > Custom Property**. Manage Custom Properties page is displayed, you use this page to add or delete custom properties.

Caution Although you can delete custom properties while they are currently in use, doing so could result in unexpected behavior.

After the custom property is defined, you can use it in the property hierarchy. See Property Hierarchy, page 20-7.

Configuring Defaults

You can access the default settings for the overall system, including the Regional Distribution Unit (RDU), Network Registration extensions, and all supported technologies. To configure or view default settings, click **Configuration > Defaults.** The Configure Defaults page appears.

To access specific defaults page, click the specific link from the Default links on the left of the screen.

This section describes:

- CableHome WAN Defaults, page 11-4
- Computer Defaults, page 11-5
- DOCSIS Defaults, page 11-5

- Network Registrar Defaults, page 11-6
- PacketCable Defaults, page 11-8
- RDU Defaults, page 11-8
- System Defaults, page 11-10
- STB Defaults, page 11-12

CableHome WAN Defaults

There are two distinct CableHome WAN default screens: one for WAN-Data devices and one for WAN-MAN devices. In either case, select the desired defaults from the list on the left pane.

- When you select the CH WAN-Data Defaults link, the CableHome WAN-Data Defaults page appears. Use this page to configure the WAN-Data device.
- When you select the CH WAN-MAN Defaults link, the CableHome WAN-MAN Defaults page appears. Use this page to configure the WAN-MAN device type.

Each WAN default page contains identical fields as described in Table 11-1.

Table 11-1 Configure Defaults–CH WAN-Data/CH WAN-MAN Defaults Page

Field or Button	Description
CableHome WAN-Data Defaults/CableHome WAN-MAN Defaults	
Extension Point	Identifies the extension point to execute when generating a configuration for a WAN device.
Disruption Extension Point	Identifies the extension point to be executed to disrupt a WAN device.
Service level Selection Extension Point	Identifies the extension used to determine the DHCP Criteria and Class of Service required for a device.
Default Class of Service	Identifies the current default Class of Service for a WAN-Data. New, unrecognized WAN devices are assigned to this Class of Service. Use the drop-down list to select a new default value.
Default DHCP Criteria	Identifies the current default DHCP Criteria for a specific device technology. New, unrecognized WAN devices are assigned this default DHCP Criteria. Use the drop-down list to select a new default value.
Automatic FQDN Generation	Automatically generates a host and domain name for the device. Two selectable options are available: • Enabled—Automatic generation of the FQDN is enabled. • Disabled—Automated FQDN generation is disabled. **Note** See Chapter 16, “Automatic FQDN Generation”, for additional information.
CableLabs Configuration Filename Script	Identifies the Groovy script to be used to generate the dynamic TFTP filename. **Note** This field appears only when you select the CableHome WAN MAN Defaults link.

Computer Defaults

When you select the Computer Defaults link, the list of default values currently applied to the computers supported by Prime Cable Provisioning appears. See Table 11-1 for the description of the fields that appear on this page.

Note Changes to the Default Class of Service or Default DHCP Criteria cause regeneration to occur. Other changes made to this page do not affect existing devices.

DOCSIS Defaults

When you select the DOCSIS Defaults link, the list of default values currently applied to the cable modems supported by Prime Cable Provisioning appears. The fields in this page are similar to the fields explained in Table 11-1. There are a few extra fields that appear in this page and those are explained in Table 11-2.

Table 11-2 Configure Defaults–DOCSIS Defaults Page

Field or Button	Description
TFTP Modem Address Option	Identifies whether the TFTP modem address option is enabled.
TFTP Time Stamp Option	Identifies whether the TFTP server will issue a timestamp.
Note If you enable either or both of the TFTP options on this page, that appropriate TFTP information is included in the TFTP file before it is sent to the DOCSIS cable modem.	
CMTS Shared Secret	Identifies the character string that Prime Cable Provisioning uses in the calculation of the CMTS MIC in the configuration file. The CMTS uses it to authenticate the configuration file that a cable modem submits to the CMTS for authorization.
CMTS Default DOCSIS Version	Specifies the default DOCSIS version used by all CMTSs. If you do not enter a DOCSIS version in this field, it will default to version 1.0.
Relay Agent IP Address to CMTS Version Mapping file	Identifies the mapping file used by the CMTS. This file specifies the DOCSIS version that the CMTS will use.
Extended CMTS MIC Option	Identifies whether the Extended CMTS MIC (EMIC) option is enabled. **Note** Only if this field is enabled do subsequent fields in this section appears.
Extended CMTS MIC HMAC Type	Identifies the default Hash-based Message Authentication Code (HMAC) type for EMIC calculation. Choose one of the following HMAC type: • MD5 • MMH16 **Note** By default, MMH16 is used for EMIC calculation.

Table 11-2 Configure Defaults–DOCSIS Defaults Page (continued)

Field or Button	Description (continued)
Extended CMTS MIC Digest Explicit Option	Identifies whether the Extended CMTS MIC Digest explicit digest option is enabled. By default, Extended CMTS MIC explicit digestion is used for EMIC calculation.
Extended CMTS MIC Shared Secret	Identifies the character string that Prime Cable Provisioning uses in the calculation of the Extended CMTS MIC in the configuration file. The CMTS uses it to authenticate the configuration file that a cable modem submits to the CMTS for authorization.
Extended CMTS MIC Fields	Identifies the TLVs that are to be included in Extended CMTS MIC calculation.

Note Changes to the default Class of Service or default DHCP Criteria cause regeneration to occur. Changes to any TFTP option come into effect starting from the next TFTP transfer.

Network Registrar Defaults

Prime Cable Provisioning provides Cisco Prime Network Registrar (NR) extension points that allow Prime Cable Provisioning to pull information from incoming DHCP packets to detect a device’s technology. The extension points also let Prime Cable Provisioning respond to device DHCP requests with options that correspond to the configuration stored at the DPE.

When you select the NR Defaults link, the list of default values currently applied to Network Registrar extensions appears. Table 11-3 identifies the fields that appear on this page.

Table 11-3 Configure Defaults–Network Registrar Defaults Page

Field or Button	Description
NR Extension Point Settings (Cisco BAC 2.7)	
Attributes Required in Request Dictionary	Identifies a comma-separated list of attributes that the Network Registrar request dictionary must include when sending a request to the RDU for configuration generation.
Attributes from Request Dictionary as Bytes	Identifies a comma-separated list of attributes pulled out of the Network Registrar request dictionary as bytes when sending a request to the RDU to generate a device configuration.
Attributes from Request Directory as Strings	Identifies a comma-separated list of attributes pulled from the Network Registrar request dictionary as strings when sending a request to the RDU to generate a device configuration.
NR Extension Point Settings (Cisco BAC 4.0, 4.1 or higher)	

Table 11-3 Configure Defaults–Network Registrar Defaults Page (continued)

Field or Button	Description
Attributes Required in DHCPv4 Request Dictionary	Identifies a comma-separated list of attributes that the Network Registrar DHCPv4 request dictionary must include for Network Registrar extensions to submit a request to the RDU to generate a device configuration. The default value for this field is the relay agent remote ID option. If you do not set the **relay-agent-remote-id** value in this field, Network Registrar extensions reject devices from triggering a request for configuration generation.
Attributes from DHCPv4 Request Dictionary as Bytes	Identifies a comma-separated list of attributes pulled from the Network Registrar DHCPv4 request dictionary as bytes when sending a request to the RDU to generate a device configuration.
Attributes from DHCPv4 Request Dictionary as Strings	Identifies a comma-separated list of attributes pulled from the Network Registrar DHCPv4 request dictionary as strings when sending a request to the RDU to generate a device configuration.
Attributes Required in DHCPv6 Request Dictionary	Identifies a comma-separated list of attributes that the Network Registrar DHCPv6 request dictionary must include for Network Registrar extensions to submit a request to the RDU to generate a device configuration. The default value for this field is **none**.
Options Required in DHCPv6 Request Dictionary	Specifies a comma-separated list of DHCP options that the Network Registrar DHCPv6 request dictionary must include for Network Registrar extensions to submit a request to the RDU to generate a device configuration.
Attributes from DHCPv6 Request Dictionary as Bytes	Identifies a comma-separated list of attributes pulled from the Network Registrar DHCPv6 request dictionary as bytes when sending a request to the RDU to generate a device configuration.
Options from DHCPv6 Request Dictionary as Bytes	Specifies a comma-separated list of DHCP options pulled from the Network Registrar DHCPv6 request dictionary as bytes when sending a request to the RDU to generate a device configuration.
Attributes Required in DHCPv6 Relay Dictionary	Identifies a comma-separated list of attributes that the Network Registrar DHCPv6 relay dictionary must include for Network Registrar extensions to submit a request to the RDU to generate a device configuration. The default value for this field is **peer-address**.
Options Required in DHCPv6 Relay Dictionary	Identifies a comma-separated list of DHCP options that the Network Registrar DHCPv6 relay dictionary must include for Network Registrar extensions to submit a request to the RDU to generate a device configuration.
Attributes from DHCPv6 Relay Dictionary as Bytes	Identifies a comma-separated list of attributes pulled out of the Network Registrar DHCPv6 relay dictionary as bytes for Network Registrar extensions to submit a request to the RDU to generate a device configuration.

Table 11-3 Configure Defaults–Network Registrar Defaults Page (continued)

Field or Button	Description
Options from DHCPv6 Relay Dictionary as Bytes	Identifies a comma-separated list of DHCP options pulled out of the Network Registrar DHCPv6 relay dictionary as bytes for Network Registrar extensions to submit a request to the RDU to generate a device configuration.
NR Extension Point Environment Settings	
Attributes from Environment Dictionary	Identifies a comma-separated list of attributes pulled out of the Network Registrar environment dictionary as strings when sending a request to the RDU to generate a device configuration.

Note Changes made to this page do not take effect until the Network Registrar extensions are reloaded.

PacketCable Defaults

The PacketCable Defaults page identifies those defaults necessary to support the PacketCable voice technology. When you select the PacketCable Defaults link, the list of default values currently applied to PacketCable devices appears. The fields in this page are similar to the fields explained in Table 11-1. There are a few extra fields that appear in this page and those are explained in Table 11-4.

Table 11-4 Configure Defaults–PacketCable Defaults Page

Field or Button	Description
SNMP Set Timeout	Identifies the SNMP set timeout in seconds.
MTA Provisioning Notification	Notification that an MTA event has taken place. An event occurs when the MTA sends its provisioning complete inform based on the selected choice. Options available include: • On Failure • On Success • During Provisioning • Always • Never

RDU Defaults

When you select the RDU Defaults link, the defaults settings that you have configured for the RDU appear. Use this page to configure the RDU to communicate with Network Registrar. For additional information, see the *Cisco Prime Network Registrar 8.1 User Guide*.

Table 11-5 describes the fields that appear on the RDU Defaults page.

Table 11-5 Configure Defaults–RDU Defaults Page

Field or Button	Description
Configuration Extension Point	Identifies the common extension points executed before any other technology extension point is executed.
Device Detection Extension Point	Identifies the extension point used to determine a device type (for example, DOCSIS or computer) based on information pulled from the device DHCP Discover requests.
Publishing Extension Point	Identifies the extension point to be used for an RDU publishing plug-in. This information is useful when you need to publish RDU data into another database.
Extension Point JAR File Search Order	Specifies the sequence in which the classes are searched in the JAR files that are listed in the preceding four fields.
CRS	Identifies whether the Configuration Regeneration Service (CRS) is enabled. There are two options: • Enable—Enables the CRS within Prime Cable Provisioning. • Disable—Disables the CRS within Prime Cable Provisioning.
Scripts recompile	By default, script compilation mode is enabled and if enabled, the scripts get compiled and cached during file addition.
Number of Sessions per User	Specifies the maximum number of allowed sessions for a user. You could specify any value between 1 to 100. The default value for this property is 100. If this property value is not assigned to a user, the value available in the RDU defaults is considered.
Authentication Mode	Identifies the authentication mode to be used. The options are: • Local—Authenticates the user in the local RDU database. • Radius—Authenticates the user, using the Radius server.
Server Authentication Mode Property Details	

Configuration Details for Radius Authentication

This table lists the fields required for configuring Radius authentication.

Table 11-6 Configure Defaults–RDU Defaults Page–Server Authentication Mode Property Details–RADIUS mode

Field or Button	Description
RADIUS Class	Identifies the class containing the Radius authentication extensions. The default value is "**com.cisco.provisioning.cpe.extensions.builtin.authentication.RadiusAuthentication**".
Primary Host	Identifies the primary IP address of the Radius server.
RADIUS Primary SharedSecret	Identifies the primary shared secret used to authenticate the Radius server user.

Table 11-6 Configure Defaults–RDU Defaults Page–Server Authentication Mode Property Details–RADIUS mode

Field or Button	Description
RADIUS Primary Port	Identifies the primary authentication port number of the Radius server. The default port number is 1812.
RADIUS Secondary Host	Identifies the secondary IP address of the Radius server, which is optional.
RADIUS Secondary SharedSecret	Identifies the secondary shared secret used to authenticate the Radius server user, which is optional.
RADIUS Secondary Port	Identifies the secondary authentication port number of the Radius server, which is optional.
RADIUS Timeout	Specifies the maximum length of time for which RDU waits for a response when trying to connect to the Radius server. The value will be specified in milliseconds and the default value is 1000 milliseconds. The value can be between 1000-5000 milliseconds.
RADIUS Retries	Specifies the maximum number of times RDU attempts to connect with the Radius server. The default value is 1 and the value can be between 1-5.

Note If the Radius time out exceeds 10000 milliseconds then BAC authentication will fail. Radius time out and retries must be configured so that it does not exceed greater than 10000 milliseconds.

System Defaults

When you select the Systems Defaults link, the System Defaults page appears. Table 11-7 describes the fields that appear on this page.

Note You can configure the default values using the Prime Cable Provisioning API.

Table 11-7 Configure System Defaults Page

Field or Button	Description
System Defaults	
SNMP Write Community String	Identifies the default write community string for any device that may require SNMP information. The default write community string is **private**.
SNMP Read Community String	Identifies the default read community string for any device that can read or access the SNMP MIB. The default read community string is **public**.

Table 11-7 Configure System Defaults Page (continued)

Field or Button	Description
System Defaults	
Default Device Type for Device Detection	Identifies the default device type for a device not previously registered in the RDU. The options include: • DOCSIS • COMPUTER • PacketCableMTA • STB • CableHomeWanMan • CableHomeWanData • None **Note** If the device detection extension is unable to identify the device type, the "default type" (for example, COMPUTER) specifies the device type. If you set the Default Device Type to None, the device record is not added to the RDU.
Maximum Diagnostics Device Count	Identifies the maximum number of MAC addresses (devices) that you can troubleshoot at any one time.
MIB List	Identifies a list of MIBs used by the RDU that do not require restarting the RDU.
Supplemental MIB List	Identifies an extended list of MIBs used by the RDU.
Excluded MIB Tokens	Defines those keywords, or tokens, that cannot be redefined by a MIB.
Excluded Supplemental MIB Tokens	Defines those additional keywords, or tokens, that cannot be redefined by a MIB and do not appear in the Excluded MIB Tokens list.
Promiscuous Policy Settings	
CableHome WanData Promiscuous Mode	Enables or disables CableHome WAN-Data devices in the promiscuous mode.
CableHome WanMan Promiscuous Mode	Enables or disables CableHome WAN-MAN devices in the promiscuous mode.
Computer Promiscuous Mode	Enables or disables computers in the promiscuous mode.
PacketCable Promiscuous Mode	Enables or disables PacketCable devices in the promiscuous mode.
STB Promiscuous Mode	Enables or disables STBs in the promiscuous mode.
CableHome WanData Promiscuous DHCP Criteria	Identifies the DHCP Criteria used to provision WAN-Data devices in the promiscuous mode.
CableHome WanMan Promiscuous DHCP Criteria	Identifies the DHCP Criteria used to provision WAN-MAN devices in the promiscuous mode.
Computer Promiscuous DHCP Criteria	Identifies the DHCP Criteria used to provision computers in the promiscuous mode.

Table 11-7 Configure System Defaults Page (continued)

Field or Button	Description
System Defaults	
Packetcable Promiscuous DHCP Criteria	Identifies the DHCP Criteria used to provision PacketCable devices in the promiscuous mode.
STB Promiscuous DHCP Criteria	Identifies the DHCP Criteria used to provision STBs in the promiscuous mode.
CableHome WanData Promiscuous Class of Service	Identifies the Class of Service used to provision WAN-Data devices in the promiscuous mode.
CableHome WanMan Promiscuous Class of Service	Identifies the Class of Service used to provision WAN-MAN devices in the promiscuous mode.
Computer Promiscuous Class of Service	Identifies the Class of Service used to provision computers in the promiscuous mode.
Packetcable Promiscuous Class of Service	Identifies the Class of Service used to provision PacketCable devices in the promiscuous mode.
STB Promiscuous Class of Service	Identifies the Class of Service used to provision STBs in the promiscuous mode.
CableLabs Configuration Filename Script	Identifies the Groovy script to be used to generate the dynamic TFTP filename.

STB Defaults

The STB Defaults page identifies those defaults necessary to support any STB compliant with CableLabs OpenCable Application Platform. This page contains identical fields as described in Table 11-1.

Note Subsequent device configurations will include the changes you implement here. However, all existing configurations are not changed. To make the changes in any existing configuration, you must regenerate the configuration using the API.

Configuring DHCP Criteria

In Prime Cable Provisioning, DHCP Criteria describe the specific criteria for a device when selecting a scope in Network Registrar. For example, a DHCP Criteria called **provisioned-docsis** has an inclusion selection tag called **tagProvisioned**. The DHCP Criteria is associated with a DOCSIS modem. When this modem requests an IP address from the Network Registrar, Network Registrar looks for scopes associated with the scope-selection tag **tagProvisioned**.

To access the DHCP Criteria page, choose **Configuration > DHCP Criteria.** The Manage DHCP Criteria page appears, listing the DHCP Criteria that identify the technology DHCP Criteria that you have added.

Adding DHCP Criteria

To add a DHCP Criteria:

Step 1 From the Manage DHCP Criteria page, click **Add**.

The Add DHCP Criteria page appears.

Step 2 Enter the name of the DHCP Criteria you want to create.

Step 3 Enter the DHCP Criteria client-class name.

Step 4 Enter the inclusion and exclusion selection tags.

Note When creating new DHCP Criteria, the client-class and inclusion and exclusion selection tag names you enter must be the exact names from within Network Registrar. For additional information on client class and selection tags, see the *Cisco Prime Network Registrar 8.1 User Guide,* and *CLIFrame.html* in the */docs* directory. You should specify either the client class, or inclusion and exclusion selection tag names, when creating a new DHCP Criteria.

Click **Assign Domain** and select the domain to which the property must belong. Click **Apply** to save the changes.

Step 5 To add a new property to the selected DHCP Criteria, click **Add Property**. Select the property from the drop-down and enter a value. Click **Save**.

Step 6 Click **Add**.

Step 7 Click **Submit**.

After the DHCP Criteria is successfully added in the RDU database, it will be visible in the Manage DHCP Criteria Page.

Modifying DHCP Criteria

Note Once you change the DHCP Criteria, subsequent device configurations will include the changes you implement. All existing configurations are regenerated, although the devices on the network will not get the new configuration until they are rebooted.

To modify existing DHCP Criteria, click the DHCP Criteria link that you want to modify. Make the desired changes to the client class, inclusion and exclusion selection tags, and the property value settings. Click **Submit**.

Deleting DHCP Criteria

Deleting DHCP Criteria using the administrator application does not delete the actual DHCP server configurations from the DHCP server. You must delete the DHCP server configurations manually.

To delete an existing criteria, select the DHCP Criteria you want to delete and click **Delete**.

Note You can delete a DHCP Criteria only if there are no devices associated with that criteria, and it is not designated as the default DHCP Criteria. If a DHCP Criteria has devices associated with it, you must associate a different DHCP Criteria before deleting the criteria.

Managing Files

Using the Prime Cable Provisioning administrator user interface, you can manage the TFTP server files or template files for dynamic generation for DOCSIS, PacketCable MTAs, and WAN-MAN files, or software images for devices. Use this page to add, delete, replace, or export any file type, including:

- Template files—These are text files that contain DOCSIS, PacketCable, or CableHome options and values that, when used with a particular Class of Service, provide dynamic file generation.

  Note Template files can be created in any text editor, but must have a *.tmpl* file type. For additional template information, see Template Files–An Overview, page 19-15.

- Static configuration files—These files are used as a configuration file for a device. For example, a static configuration file, called *gold.cm*, would identify the gold DOCSIS Class of Service. Prime Cable Provisioning treats this file type like any other binary file.
- Firmware images—These are images of device firmware, which can be downloaded to devices to upgrade their functionality. Prime Cable Provisioning treats this file type like any other binary file. These firmware images can also include IOS images for Cisco devices.

Table 11-8 describes the fields that appear on the View Files page.

Table 11-8 View Files Page

Field or Button	Description
Search Type	Identifies the types of searches that you can perform for files using the Prime Cable Provisioning administrator user interface. The options include: • Search by File Name—Searches for files using the filename pattern that you specify. • Search by File Type—Searches for files using the file type that you specify. The options include: – Firmware File—Specifies a firmware image file. – CableLabs Configuration File—Specifies a static configuration file for CableLabs. – CableLabs Configuration Script—Specifies a configuration script file for CableLabs. – CableLabs Configuration Template—Specifies a configuration template file for CableLabs. – CableLabs Configuration Filename Script—Specifies a configuration filename with a script for CableLabs. – Generic File—Specifies a generic file. – JAR File—Specifies a JAR file. – MIB File—Specifies a MIB file. – Script File—Specifies a script file.
Search Criteria	Identifies the filename or file type. You can use an asterisk (*) as a wildcard character to search for partial filenames. For example, you can enter ***.cm** to list all files ending with the *.cm* extenOasion. An example of an invalid wildcard is bronze*.
Files	Displays a list of files that match the search criteria. **Note** The check boxes immediately to the left of any selected item in this list must be checked before the item can be deleted.
View	Displays the details of the selected file.
File Type	Identifies the type of file.
Export	Exports any selected file to the client’s computer.

Adding Files

To add an existing file:

Step 1 From the View Files page, click **Add**.

Step 2 Select the File Type from the drop-down list.

Step 3 Enter the path to the source file.

If you do not know the exact name of the source file, use **Browse** to navigate to the desired directory and select the file.

Step 4 Enter the name of the file.

If you are adding a CableLabs Configuration File or a Firmware File, you must also complete these steps, otherwise go to Step 6.

a. When adding a CableLabs Configuration File or a Firmware File, you can deliver the files that you add to the RDU to the DPE. To do so, click the **Enabled** radio button corresponding to the Is Deliverable field.

   While Prime Cable Provisioning sets a deliverable status for each file type, you can change the default setting only for a CableLabs Config File or a Firmware File. The following list describes the default deliverable status for each file type:

   - Firmware File—Enabled
   - CableLabs Configuration File—Enabled
   - CableLabs Configuration Template—Disabled
   - Generic File—Disabled
   - JAR File—Disabled
   - MIB File—Disabled
   - CableLabs Configuration Script—Disabled

b. In the case of a Firmware File, additionally enter the file version and a suitable description for that version.

Step 5 Click **Assign Domain** and select the domain to which the file must belong. Click **Apply** to save the changes.

Step 6 Click **Submit.**

Note File sizes up to 4 MB are supported. If the size of the file that you are adding is over 4 MB, an error appears.

The View Files page appears, indicating that the file has been added.

Deleting Files

To delete an existing file, locate the file you want to delete using the search option. Choose the appropriate file or files and click **Delete.**

Caution Deleting a template file that is not directly linked to a Class of Service, but is referenced by another template file that is linked to a Class of Service, will cause the configuration regeneration service to fail.

Note You cannot delete a file that has a Class of Service associated with it. You must remove the Class of Service association before proceeding. See Configuring Class of Service, page 11-1, for additional information.

Publishing Provisioning Data

Prime Cable Provisioning has the capability to publish the provisioning data it tracks to an external datastore in real time. To do this, a publishing plug-in must be developed to write the data to the desired datastore. The Manage Publishing page identifies information such as the plug-in name, its current status (whether it is enabled or disabled), and switch to enable or disable it.

You can enable as many plug-ins as required by your implementation, but remember that the use of publishing plug-ins can decrease system performance.

Note Prime Cable Provisioning does not ship with any publishing plug-ins. You must create your own plug-ins and load them into Prime Cable Provisioning in the same way as JAR files are (see Adding Files, page 11-15). Then, manage the plug-ins from the Manage Publishing page.

Publishing Datastore Changes

To enable or disable a publishing plug-in:

Step 1 Choose **Configuration** > **Publishing** on the Primary Navigation bar.

Step 2 Click on the appropriate status indicator to enable or disable the required plug-in. Note that as you click the status, it toggles between the two states.

Modifying Publishing Plug-In Settings

These settings are a convenient way for plug-in writers to store plug-in settings in the RDU for their respective datastore. To modify the publishing plug-in settings:

Step 1 Choose **Configuration** > **Publishing** on the Primary Navigation bar.

Step 2 Click the link corresponding to the plug-in you want to modify. The Modify Publishing Plug-Ins page appears.

Table 11-9 identifies the fields shown in the Modify Publishing Plug-Ins page.

Table 11-9 Modify Publishing Plug-Ins Page

Field	Description
Plug-In	Identifies the publishing plug-in name.
Server	Identifies the server name on which the datastore resides.
Port	Identifies the port number on which the datastore resides.
IP Address	Identifies the IP address of the server on which the datastore resides. This address is usually specified when the server name is not used.
User	Identifies the user to allow access to the data stored.
Password	Identifies the user's password, which allows access to the data stored.
Confirm Password	Confirms the password entered above.

Step 3 Enter the required values in the Server, Port, IP Address, User, Password, and Confirm Password fields. These are all required fields and you must supply this information before proceeding.

Step 4 Click **Submit** to make the changes to the selected plug-in.

Property Encryption

By default, the values assigned to custom properties and device properties at various entity levels like CoS, DHCP criteria and devices are stored as plain objects such as string, integer or boolean in the RDU. Considering this data to be sensitive, Prime Cable Provisioning lets you encrypt these properties. These property values are decrypted and returned as actual plain objects when you retrieve them using the API calls.

To enable property encryption:

Step 1 Choose **Configuration > Property Encryption** on the Primary Navigation bar.

You can configure a custom property for encryption while creating it.

1. Choose **Configuration > Custom Property**.
2. In the Add Custom Property page, select the field Encrypt Property.

All custom properties tagged for encryption in this page show up in the Property Encryption page.

Step 2 You can also add a new property for encryption from the Property Encryption page. Click Add Property.

Step 3 Select the property to be encrypted and click **Save**.

Device properties can be added for encryption only through the Property Encryption page.

Note To disable encryption, select the properties from the Property Encryption list and click **Delete**.

CHAPTER 12

Configuring Groups Using Admin UI

Group management allows you to create, change, and delete groups and group types. Within the context of Prime Cable Provisioning, group types can be considered as sets of groups, while groups themselves make up the group type.

Managing Group Types

Access the Manage Groups page by selecting Groups from the Main Menu. Group Type is the default setting when this page appears.

Adding a Group Type

To add a new group type:

Step 1 From the Manage Groups page, click **Add**.

Step 2 Enter a name for the new group type.

Note If you previously added custom properties, you can choose the appropriate Property Name from the drop-down list and enter the required Property Value. Click **Add** to increase the number of applicable Property Name/Property Value pairs.

Step 3 Enter the priority value for the new group type.

The value can range between 1 and 100. The value 1 has the highest priority and 100 has the lowest. For example, if the priority values of two member groups are 5 and 20, then the group with priority value 5 has more priority than the group with priority value 20.

By default, the Group Type Priority is set to 50. If two member groups have the same priority value, the group type names are sorted in alphabetical order to decide the priority.

Step 4 Click **Submit**.

The new group type is recorded in the RDU.

Managing Groups

You can create and modify groups, delete unwanted groups, relate and unrelate groups and group types, and view the devices that you associated with a group.

Adding a New Group

To add a new group:

Step 1 On the Manage Groups page, select **Groups** from the Search Type drop-down list.

Step 2 Click **Add Group.**

Step 3 Enter the new group name and select the appropriate Group Type for this group.

Note If you previously added custom properties, you can choose the appropriate Property Name from the drop-down list and enter the required Property Value. Click **Add** to increase the number of applicable Property Name/Property Value pairs.

Step 4 Click **Submit**.

The new group is recorded in the RDU.

Searching for Devices in a Group

To view devices associated with a group:

Step 1 From the Manage Groups page, select the **Groups** option from the **Search Type** drop-down list.

Step 2 You can choose to search either by group type or group name.

- By Group type—Provides a drop-down list of predefined groups.
- By Group name—Provides a Group or Group Wildcard field in which you can enter the name of the name or a wildcard (*) character.

Step 3 Click **Search**.

Step 4 Click the **View Details** icon under the Devices parameter corresponding to the group.

Doing this displays the Group Search function on the Manage Devices page.

Step 5 From the Manage Devices page, select the appropriate Group Type. See Searching for Devices, page 18-1, for additional information on search functions.

The devices associated with the group appear.

Relating and Unrelating Group Types to Groups

The relate and unrelate functions are used to establish a relationship between specific groups and group types.

To either relate or unrelate this relationship:

Step 1 From the Manage Groups page, choose Groups from the Search Type drop-down list.

Step 2 Choose the group type for which you want to relate or unrelate groups using the group type or group name search criteria.

Step 3 Click **Search**.

Step 4 Click the Relate to Group or Unrelate from Group link.

Depending on the link you clicked, either the Relate Group or the Unrelate Group page appears.

Step 5 Select the appropriate Group Type from the drop-down list, and select the group to which the group is to be related or unrelated.

Step 6 Click **Submit**.

The Manage Groups page appears.

CHAPTER 13

Configuring RBAC Using Admin UI

This chapter describes the Security feature of Prime Cable Provisioning. Use this feature to configure and manage various levels of security. For conceptual information about the RBAC feature, see RBAC Management, page 3-14.

For better user management and security, Prime Cable Provisioning introduces Role Based Access Control (RBAC) that provides an approach to restrict access to system functions and resources to authorized users. Roles are composed of fine grain privileges. A privilege is a base unit of enforcement. A role groups a set of privileges into a logical job function to enable the customization of authorization policies beyond the default provided out of the box.

Prime Cable Provisioning comes with some default out of the box (OOTB) roles, privileges, users, user groups and domains that you can leverage from. Apart from these default configurations, you can also define your own setup to meet your organization requirements. The default OOTB configurations cannot be edited or deleted.

There are four levels of checks.

- URL access check - Enforcement by web facing components such as Admin UI or web services.
- Operation/Method level check - Enforcement done by the components protecting access to operations. This type of access check is primarily performed in the RDU and DPE CLI. It is meant to ensure that the user has the correct privileges to invoke operations.
- Instance level check - Enforcement to ensure that the user has access to a specific object. This enforcement is performed in the RDU and leverage database capabilities.
- Property level check - Enforcement to ensure that the user has write access to a specific property. This enforcement is performed in the RDU.

Configuring Security

Use the Security menu to configure and manage various levels of security. You can:

- Add, modify, or delete domains, see Domain Management, page 13-2.
- Add, modify, clone or delete roles using the default privileges, see Role Management, page 13-3.
- Add, modify, clone or delete user groups, assign roles to the user groups, see User Group Management, page 13-4.
- Map the existing external user groups to the Prime Cable Provisioning user groups, see User Group Mapping, page 13-5.
- Add, modify, or delete users, assign roles and domains to these users, see User Management, page 13-5.

Domain Management

Domains are a set of instances with various objects such as Device, COS, File, DPE, NR, Provisioning Group, and DHCP Criteria. Domain represents a collection of these objects grouped for instance level access control. Only authenticated users with the appropriate access privileges will be able to view the instances that exist in their domains.

Domains are represented hierarchically with all the custom domains added as sub domains of the default domain RootDomain. A user who has access to a parent domain can access all of the sub domains of that parent domain.

Adding a Domain

Note By default, Domain management related pages or widgets are not available in the Admin UI. Even the instance level authz field is not displayed.
To enable them, the property */adminui/enableDomainAdministration* must be set to true. This property can be set in the adminui.properties file located at BPR_HOME/rdu/conf. Restart the tomcat server after making the changes to the property file.

To add a domain:

Step 1 Choose **Security > Domain Management**.

Step 2 Select a parent domain and click **Add Domain** to display the Add Domain page.

Step 3 Enter the new domain's name. Domains must have unique names across the system.

Step 4 Enter a short description of the new domain.

Tip Use the description field to identify the domain or any detail that uniquely identifies the new domain.

Step 5 Click **Save**.

Role Management

A role is a job function that defines a set of capabilities a user or user group can perform. These capabilities are governed by the privileges assigned to the role. Privileges allow the user to perform operations like create, read, update, and delete objects and its properties in Prime Cable Provisioning. Privileges are in-built in Prime Cable Provisioning and cannot be modified, see Table 13-2 for the list of default privileges.

A set of default out-of-the-box roles are available for use. You also can create custom roles with any set of in-built privileges assigned to a custom role. These roles are loaded into the RDU database after installing Prime Cable Provisioning.

Adding a New Role

To add a role:

Step 1 Choose **Security > Role Management**.

Step 2 Click **Add Role** to display the Add Role page.

Step 3 Enter the new role’s name.

Step 4 Enter a short description of the new role.

Tip Use the description field to identify the role or any detail that uniquely identifies the new role.

Step 5 To add privileges to the new role:

a. Click the Privileges tab and then click Add Privileges.

b. Check the appropriate check box to determine the new role’s privilege.

c. Click **Apply**.

Step 6 To add the properties that can be modified for this role:

a. Click the Modifiable Device Properties tab and then click Add Modifiable Device Properties.

b. Check the appropriate check box to add the permission to modify the selected device properties to the new role.

c. Click **Apply**.

Step 7 To add custom property:

a. Click the Custom Properties tab and then click Add Custom Properties.

b. Check the appropriate check box to add the custom property to the new role.

c. Click **Apply**.

Note If you are adding a custom property through the Admin UI, it can be of type String only. However, if you are adding it using the Configuration.addRole API, it need not be restricted to the type String alone.

Step 8 Click **Save**.

Modifying a Role

The default roles listed in Table 13-3 cannot be modified. To modify a custom role, select the role and click **Edit.** Make the necessary edits and click **Save**.

Note If you are modifying a custom property through the Admin UI, it can be of type String only. However, if you are using the Configuration.changeRoleProperties API, it need not be restricted to the type String alone.

User Group Management

A user group is a collection of users. Similar to a user, a user group can also be assigned roles. Users who belong a user group will inherit all the roles assigned to that the user group. Those roles are constrained to only be valid on the resources that are also members of the group. A user can be a member of more than one group. The set of privileges the user gains is the aggregate of all those from the role.

From the User Group Management option you can add, modify, delete and clone user groups.

Adding a New User Group

To add a user group:

Step 1 Choose **Security > User Group Management**.

Step 2 Click **Add User Group** to display the Add User Group page.

Step 3 Enter the new user group's name.

Step 4 Enter a short description of the new user group.

Tip Use the description field to identify the user group's role or any detail that uniquely identifies the new user group.

Step 5 Click Add Roles.

Step 6 Check the appropriate check box to determine the new user group's role.

Step 7 Click **Apply**.

Step 8 Click **Save**.

Note Only the administrator can delete any other user groups that appears in the User Group Management page. You cannot delete the default user group, called Administrators.

User Group Mapping

Prime Cable Provisioning provides user-group mapping, which enables mapping of an external user-group name to a Prime Cable Provisioning user-group name. An external group can be mapped to any existing Prime Cable Provisioning user-group. In the example table below Operator is mapped to ProvGroupAdmin and Admin is mapped to administrators.

Table 13-1 lists examples for User Group mapping.

Table 13-1 Example User Group Mapping

External user-group	RDU user-group name
Operator	ProvGroupAdmin
Admin	Administrators

In the user-group mapping table, set of external group names must be unique and not duplicates. However, more then one external group can map to an internal user-group.

Note Before deleting an internal user-group, all the mappings to that user-group should be deleted.

Adding a User Group Mapping

To create a new user group mapping:

Step 1 Choose **Security > User Group Mapping.**

Step 2 Click **Add User Group Mapping**. A new blank row appears.

Step 3 Enter the existing external user group name in the Remote Group Name field.

Note Remote Group Name field is case sensitive.

Step 4 Select the user group to be mapped from the User Group Name drop-down list.

Step 5 Click **Save**.

User Management

Managing users involves adding, modifying, and deleting users who administer Prime Cable Provisioning. Depending on your privileges you can use this menu to add, modify, and delete users. This menu displays all users configured to use Prime Cable Provisioning and identifies their user groups.

Prime Cable Provisioning provides role based access to a user with specific privileges to ensure access control and the integrity of provisioning data. A user can be assigned roles that determine the scope of actions they can perform in Prime Cable Provisioning. A user can also be added to user groups with pre-assigned roles.

The assigned username appears near the top-right corner of every screen on the administrator user interface.

Note During migration from an acceptable previous release to Prime Cable Provisioning, all migrated read-only users are assigned to the out of the box read only role and RootDomain. Similarly, all the read-write users are assigned to the out of the box read only role and RootDomain.

You can administer users only if you have user related privileges.

Adding a New User

Adding a new user is a simple process of entering the user's name and creating a password. However, while creating a new user you must specify number of sessions, assign a role, or add the user to a user group or domain to be able to gain privileges to perform specific actions.

Note Prime Cable Provisioning comes with one Admin user already created; you cannot create this user again.

To add a new user:

Step 1 Choose **Security > User Management**.

Step 2 Click **Add** to display the Add User page.

Step 3 Enter the new user's name.

Step 4 You can restrict the number of concurrent sessions a user can have by modifying the value in the **Number of sessions allowed** field. If you do not specify any value in this filed, the number of sessions allowed for the user would be decided on the value of the field at the RDU Defaults page.

Step 5 Enter a short description of the new user.

Tip Use the description field to identify the user's job, position, or any detail that uniquely identifies the new user.

Step 6 Enter a password and confirm it. Ensure that the password that you enter has at least 8 characters.

Step 7 To add roles to the new user:

- **a.** Click the Roles tab and then click Add Roles.
- **b.** Check the appropriate check box to determine the new user's role.
- **c.** Click **Apply**.

Step 8 To add the new user to a user group:

- **a.** Click the Usergroup tab and then click Add Usergroups.
- **b.** Check the appropriate check box to add the new user to the user groups.
- **c.** Click **Apply**.

Step 9 To add the new user to a domain:

- **a.** Click the Domain tab and then click Add Domains.
- **b.** Check the appropriate check box to add the new user to the domain.

c. Click **Apply**.

Step 10 Click **Save**.

Note Remember to record and store the new user's password in a safe place to help prevent loss or theft and possible unauthorized entry.

Default Configurations

This section describes the default configurations of Prime Cable Provisioning.

Default Privileges

Table 13-2 lists the default privileges in Prime Cable Provisioning.

Table 13-2 Default Privileges

Privileges	Description
All	
*	The user is granted access to all the objects. It is equivalent to granting all the privileges to the user.
Class of Service	
PRIV_COS_CREATE	Enables adding a new COS object in the system.
PRIV_COS_READ	Enables viewing a COS object and all of its properties. Enables the selection of COS objects.
PRIV_COS_UPDATE	Enables modifying any property of a COS object.
PRIV_COS_DELETE	Enables deleting a COS object from the system.
DHCP Criteria	
PRIV_DHCP_CRITERIA_CREATE	Enables adding a new DHCP Criteria object in the system.
PRIV_DHCP_CRITERIA_READ	Enables viewing a DHCP Criteria object and all of its properties. Enables the selection of DHCP Criteria objects.
PRIV_DHCP_CRITERIA_UPDATE	Enables modifying any property of a DHCP Criteria object.
PRIV_DHCP_CRITERIA_DELETE	Enables deleting a DHCP Criteria object from the system.
Files	
PRIV_FILE_GENERIC_CREATE	Enables adding generic files into the system.
PRIV_FILE_GENERIC_READ	Enables viewing, searching, selecting, and exporting properties and data of generic files.
PRIV_FILE_GENERIC_UPDATE	Enables replacing a generic file.
PRIV_FILE_GENERIC_DELETE	Enables deleting a generic from the system.

Table 13-2 Default Privileges (continued)

Privileges	Description
PRIV_FILE_CABLELABS_CONF_SCRIPT_CREATE	Enables adding CableLabs script file into the system.
PRIV_FILE_CABLELABS_CONF_SCRIPT_READ	Enables viewing, searching, selecting, and exporting properties and data of CableLabs script file.
PRIV_FILE_CABLELABS_CONF_SCRIPT_UPDATE	Enables replacing a CableLabs script file.
PRIV_FILE_CABLELABS_CONF_SCRIPT_DELETE	Enables deleting a CableLabs script file from the system.
PRIV_FILE_CABLELABS_CONF_TMPL_CREATE	Enables adding CableLabs template file into the system.
PRIV_FILE_CABLELABS_CONF_TMPL_READ	Enables viewing, searching, selecting, and exporting properties and data of CableLabs template file.
PRIV_FILE_CABLELABS_CONF_TMPL_UPDATE	Enables replacing a CableLabs template file.
PRIV_FILE_CABLELABS_CONF_TMPL_DELETE	Enables deleting a CableLabs script template from the system
PRIV_FILE_CABLELABS_STATIC_CONF_CREATE	Enables adding CableLabs static config file into the system.
PRIV_FILE_CABLELABS_STATIC_CONF_READ	Enables viewing, searching, selecting, and exporting properties and data of CableLabs static config file.
PRIV_FILE_CABLELABS_STATIC_CONF_UPDATE	Enables replacing a CableLabs static script config file.
PRIV_FILE_CABLELABS_STATIC_CONF_DELETE	Enables deleting a CableLabs static script config file from the system.
PRIV_FILE_DCFG_CREATE	Enables adding a Dynamic Configuration Filename Generation (DCFG) script into the system.
PRIV_FILE_DCFG_READ	Enables viewing, searching, selecting, and exporting properties and data of DCFG script.
PRIV_FILE_DCFG_UPDATE	Enables replacing a DCFG script.
PRIV_FILE_DCFG_DELETE	Enables deleting a DCFG script from the system.
PRIV_FILE_FIRMWARE_CREATE	Enables adding a firmware image into the system.
PRIV_FILE_FIRMWARE_READ	Enables viewing, searching, selecting, and exporting properties and data of firmware image.
PRIV_FILE_FIRMWARE_UPDATE	Enables replacing firmware image.
PRIV_FILE_FIRMWARE_DELETE	Enables deleting a firmware image from the system.
PRIV_FILE_JAR_CREATE	Enables adding a JAR file into the system.
PRIV_FILE_JAR_READ	Enables viewing, searching, selecting, and exporting properties and data of a JAR file.
PRIV_FILE_JAR_UPDATE	Enables replacing a JAR file.
PRIV_FILE_JAR_DELETE	Enables deleting a JAR file.

Table 13-2 Default Privileges (continued)

Privileges	Description
PRIV_FILE_MIB_CREATE	Enables adding a MIB file into the system.
PRIV_FILE_MIB_READ	Enables viewing, searching, selecting, and exporting properties and data of a MIB file.
PRIV_FILE_MIB_UPDATE	Enables replacing a MIB file.
PRIV_FILE_MIB_DELETE	Enables deleting a MIB file from the system.
Devices, DeviceType	
PRIV_DEVICE_CREATE	Enables adding a new device into the system.
PRIV_DEVICE_READ	Enables viewing device properties, searching for devices, and selecting devices. Also permits the use of show device-config in the DPE CLI
PRIV_DEVICE_UPDATE	Enables changing any property of a device object.
PRIV_DEVICE_DELETE	Enables deleting a device from the system.
PRIV_DEVICE_REGEN	Enables invoking regeneration of the device configuration.
PRIV_DEVICE_OPERATION	Enables operations on this device.
RDU	
PRIV_RDU_READ	Enables viewing RDU status.
PRIV_RDU_EVENT	Required to register for RDU events.
Node Type and Node	
PRIV_GROUP_CREATE	Enables creating a group.
PRIV_GROUP_READ	Enables viewing and selecting all groups.
PRIV_GROUP_UPDATE	Enables updating a group.
PRIV_GROUP_DELETE	Enables deleting a group.
LicenseKey	
PRIV_LICENSE	Enables adding, updating, and deleting all the license keys. Viewing license is not protected.
Publishing	
PRIV_PUBLISHING	Enables all, read, and update.
CRS	
PRIV_CRS_CREATE	Enables creating CRS
PRIV_CRS_READ	Enables viewing and searching of the requests queued by CRS.
PRIV_CRS_UPDATE	Enables a user to pause or resume a CRS job
PRIV_CRS_DELETE	Enables a user to delete a CRS job.
ProvGroup	
PRIV_PROVGROUP_READ	Enables viewing Provisioning Group properties
PRIV_PROVGROUP_UPDATE	Enables updating ProvGroup properties
PRIV_PROVGROUP_DELETE	Enables deleting a provisioning group from Prime Cable Provisioning.
DPE	

Table 13-2 Default Privileges (continued)

Privileges	Description
PRIV_DPE_READ	Enables viewing DPE status. For DPE CLI, permits Disabled Mode.
PRIV_DPE_UPDATE	Permits DPE CLI Enabled Mode
PRIV_DPE_DELETE	Permits deleting a DPE.
PRIV_DPE_SECURITY	All security related admin operations including changing dpe admin password and configuring authentication.
NR	
PRIV_NR_READ	Enables viewing CNR status.
PRIV_NR_UPDATE	Enables updating NR extension point
PRIV_NR_DELETE	Enables deleting a CNR from Prime Cable Provisioning.
User	
PRIV_USER_CREATE	Enables adding users.
PRIV_USER_READ	Enables viewing or searching users.
PRIV_USER_UPDATE	Enables changing user properties.
PRIV_USER_DELETE	Enables deleting users.
PRIV_USER_SECURITY	Enables assigning roles, user group to users. Also allows the setting of a user's number of allowed sessions. **Note** PRIV_USER_SECURITY is a powerful privilege and must be used with caution.
Role	
PRIV_ROLE_CREATE	Enables adding roles.
PRIV_ROLE_READ	Enables reading or search roles and privileges.
PRIV_ROLE_UPDATE	Enables changing role properties.
PRIV_ROLE_DELETE	Enables deleting roles.
Domain	
PRIV_DOMAIN_CREATE	Enables adding domains.
PRIV_DOMAIN_READ	Enables viewing and selecting operations on domains.
PRIV_DOMAIN_UPDATE	Enables updating operations.
PRIV_DOMAIN_DELETE	Enables deleting operations.
Custom Property	
PRIV_PROPERTY_CREATE	Enables adding a new customer property.
PRIV_PROPERTY_READ	Enables viewing of RDU Defaults and custom property.
PRIV_PROPERTY_UPDATE	Enables modifying a custom property.
PRIV_PROPERTY_DELETE	Enables deleting a custom property.
System Defaults	
PRIV_SYSDEF_READ	Enables viewing system properties e.g. GetRDUDefaults.

Table 13-2 Default Privileges (continued)

Privileges	Description
PRIV_SYSDEF_UPDATE	Enables modifying system properties e.g. ChangeRDUDefaults
Logging	
PRIV_LOGGING	Setting logging/debug levels. Viewing logs.
PRIV_AUDIT_LOGGING	Enables viewing Audit logs.
User Group	
PRIV_USERGROUP_CREATE	Enables create operations
PRIV_USERGROUP_READ	Enables all read and selection operations
PRIV_USERGROUP_UPDATE	Enables update operations
PRIV_USERGROUP_DELETE	Enables delete operations

Default Roles

Table 13-3 lists the default roles in Prime Cable Provisioning.

Table 13-3 Default Roles

Role Name	Description
Admin	Admin is the super administrator and has all capabilities, including modifying device property value.
COSAdmin	COS admin can add, delete, update, search, and export all COS and their properties.
DeviceAdmin	Device admin can add, delete, search, relate, un-relate, regenerate, and device operation privileges on all available devices. DeviceAdmin also has permissions to read and modify all attributes and properties.
DHCPAdmin	DHCP admin can add, delete, update, and search all DHCP Criteria and their properties.
FileAdmin	File admin can add, delete, update, search, and export all files and their properties.
ProvGroupAdmin	Provisioning Group admin can view and update Provisioning Group properties. ProvGroupAdmin can also view, update, and delete servers of a Provisioning Group as well as their properties. This role permits all operations on the DPE CLI.
RDUAdmin	RDUAdmin can view, add, and delete all RDU default and system properties. This role can read, add, and delete permissions on license, read and modify permissions on all available publishing plug-in, manage CRS, and manage MIBs.
ReadOnly	ReadOnly has read permission on all available resources, except user, user group, domain, and roles.
ReadWrite	ReadWrite has create, read, modify, and delete permission on all available resources except user, user group, domain, and roles.

Table 13-3 Default Roles (continued)

Role Name	Description
	ReadOnly and ReadWrite roles are provided for backward compatibility only. These roles do not have access to any security related features like user, user-group, domain, role, and user-group-mapping that are introduced in Prime Cable Provisioning 5.0. **Note** If instance level is enabled, ReadWrite will not add any resource which supports instance level check.
SecurityAdmin	SecurityAdmin has add, delete, relate, un-relate permissions on all available groups and modify permission on all attributes.
UserAdmin	UserAdmin can add, delete, modify, read, relate, un-relate all available users and properties for user.

Default User Groups

Table 13-4 lists the default user groups in Prime Cable Provisioning.

Table 13-4 Default User Groups

User Group	Description	Role
Administrators	This user group consists super users.	Admin

Default Domains

Table 13-5 lists the default domains in Prime Cable Provisioning.

Table 13-5 Default Domains

Domain	Members
RootDomain	All current objects

Default Users

Table 13-6 lists the default user in Prime Cable Provisioning.

Table 13-6 Default Users

User	Assigned Role	Assigned User Group	Assigned Domain
admin	Admin	Administrators	RootDomain

PART 3

Provisioning CPEs

This part contains the following sections:

- DOCSIS Provisioning
- Lease Query
- PacketCable Provisioning
- Provisioning CPEs in Promiscuous Mode
- Provisioning Devices Using Admin UI
- Managing Dynamic File Configuration
- CPE Provisioning Overview

DOCSIS Provisioning

This chapter describes the provisioning flow in a Cisco Prime Cable Provisioning DOCSIS deployment. It also provides information required before configuration and describes the available tools.

DOCSIS Workflow

This section describes the provisioning workflow contained in the DOCSIS Provisioning Specification for DHCPv4 and DHCPv6.

- DOCSIS DHCPv4 Workflow, page 14-1
- DOCSIS DHCPv6 Workflow, page 14-4

DOCSIS DHCPv4 Workflow

Figure 14-1 shows the provisioning workflow contained in the DOCSIS Provisioning Specification for DHCPv4. Each step is described subsequently.

Figure 14-1 DOCSIS DHCPv4 Provisioning Flow

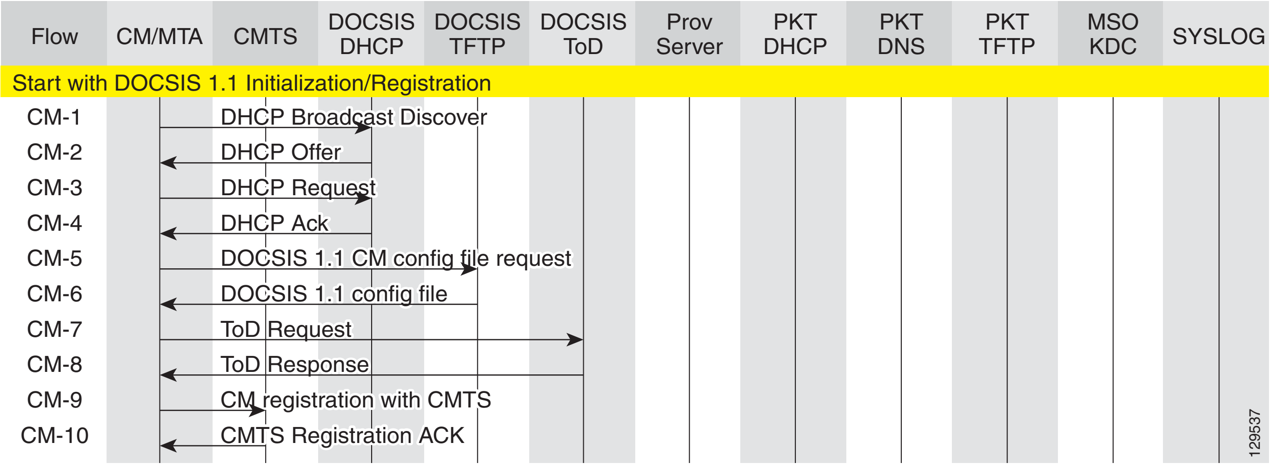


Table 14-1 describes the potential problems that can exist at various DOCSIS provisioning steps illustrated in Figure 14-1.

Table 14-1 DOCSIS DHCPv4 Workflow Description

Step	DOCSIS DHCPv4 Workflow	Potential Problems
CM[1]-1	DHCP Discover	• init(d) state • No addresses available • Incorrect Prime Cable Provisioning shared secret • Incorrectly configured Class of Service • DOCSIS template parsing errors (invalid option, include file - not found, and so on) **Cisco Prime Network Registrar DHCP** • Incorrect DHCP configuration • DHCP server not there in provisioning group **Prime Cable Provisioning Network Registrar Extension** • Network Registrar extension cannot contact DPEs • Network Registrar extension fails to find any DPEs in provisioning group • Verify extensions are connected to the RDU • Network Registrar extension gets DPE cache miss, sends request to RDU **RDU** • No appropriate scopes defined (or do not match configuration in RDU) • Incorrect IP address of RDU • Incorrect RDU port • RDU cannot be pinged from DPE • Configuration generation failing at RDU • RDU licenses exceeded, not configured • Device detection failing at RDU **DPE** • DPEs not assigned to provisioning group • DPEs cannot be pinged from the DHCP server • DPE interfaces not enabled for provisioning

Table 14-1 DOCSIS DHCPv4 Workflow Description (continued)

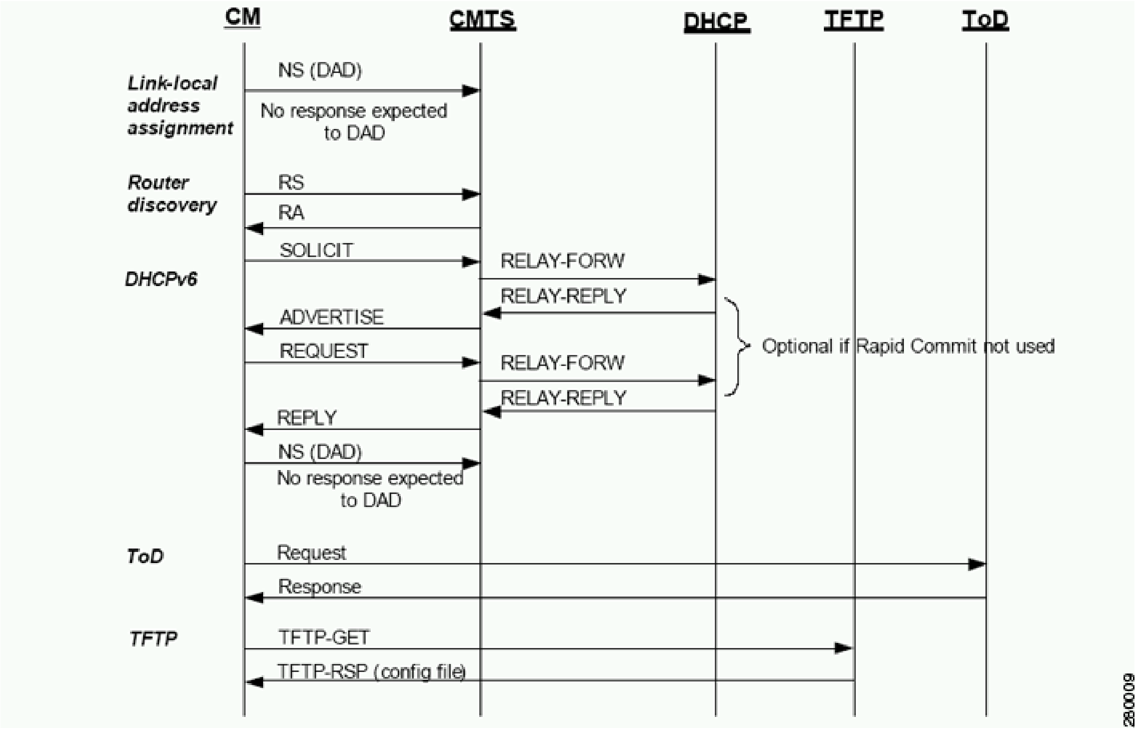
Step	DOCSIS DHCPv4 Workflow	Potential Problems
CM-2	DHCP Offer	Routing issues between DHCP and the cable modem termination system (CMTS)
CM-3	DHCP Request	• init(i) state • DHCP server did not provide all required parameters
CM-4	DHCP Ack	
CM-5	TFTP Request	• Init(o) state • Routing issues between CMTS and DPE • No route from TFTP server (DPE) to modem • DPE cache miss (static file, and RDU down or does not have file) • File not found at TFTP server (DPE) • DPE cache miss (dynamic file) • DPE IP validation failure (for example, the IP address of the device is not what was expected, the Dynamic Shared Secret is enabled on CMTS, or a hacker is spoofing as a DOCSIS modem)
CM-6	TFTP Response	Routing issues between DPE and CMTS
CM-7	ToD Request	init(t) state - No route from time server (DPE) to modem
CM-8	ToD Response	
CM-9	CM registration with CMTS	• reject(m) - * CMTS shared secret mismatch with Prime Cable Provisioning or DPE DOCSIS shared secret • reject(c) - * delivered incorrect DOCSIS configuration file (1.1 file to 1.0 cable modem)
CM-10	CMTS registration Ack	Acceptable states are: • online • online(d) • online(pk) • online(pt)

[1] CM = cable modem

DOCSIS DHCPv6 Workflow

Figure 14-2 shows the provisioning workflow contained in the DOCSIS Provisioning Specification for DHCPv6. Each step is described subsequently.

Figure 14-2 DOCSIS DHCPv6 Provisioning Flow



The DOCSIS provisioning workflow for DHCPv6 involves the cable modem establishing IPv6 connectivity, which includes assigning:

- Link-local address
- Default router
- IPv6 management address
- Other IPv6 configuration

Table 14-2 describes the potential problems that can exist at various DOCSIS provisioning steps illustrated in Figure 14-2.

Table 14-2 DOCSIS DHCPv6 Workflow Description

Workflow	Description	Potential Problems
Provisioning Phase: Link-local address assignment		
The cable modem constructs an IPv6 link-local address from the EUI-64 (64-bit Extended Unique Identifier), which is derived from the MAC address of the interface.		
NS (DAD)	The cable modem uses an NS (Neighbor Solicitation) message to perform duplicate address detection (DAD). DAD verifies if the constructed link-local address is already in use. If there is no response to the NS, the cable modem determines that the link-local address is not in use. If a response is returned, it implies that the link-local address conflicts with the MAC address, and the cable modem stops the provisioning process.	
Provisioning Phase: Router Discovery		
The cable modem uses router discovery to find a default router and identify prefixes on a HFC link.		
RS	The cable modem sends an RS (Router Solicitation) to the CMTS to trigger transmission of the periodic Router Advertisement message (RA).	
RA	The CMTS router sends periodic RAs, each of which contains the: • List of IPv6 prefixes assigned to the link • Directive to use DHCPv6 • Availability of the CMTS router as the default router	

Table 14-2 DOCSIS DHCPv6 Workflow Description (continued)

Workflow	Description	Potential Problems
Provisioning Phase: DHCPv6		
Solicit	The cable modem sends a Solicit message to locate DHCP servers.	• init6(s) state • No IPv6 addresses available • Incorrect Prime Cable Provisioning shared secret • Incorrectly configured Class of Service • DOCSIS template parsing errors (invalid option, include file - not found, and so on) **Network Registrar DHCP** • Incorrect DHCPv6 configuration • DHCP server not there in provisioning group • No appropriate prefixes defined (or do not match Prime Cable Provisioning RDU configuration) **Prime Cable Provisioning Network Registrar Extension** • Network Registrar extension cannot contact DPEs • Network Registrar extension fails to find IPv6 DPEs in provisioning group • Verify extensions are connected to the RDU • Network Registrar extension gets DPE cache miss, sends request to RDU **RDU** • Incorrect IP address of RDU • Incorrect RDU port • RDU cannot be pinged from DPE • Configuration generation failing at RDU • RDU licenses exceeded, not configured • Device detection failing at RDU **DPE** • DPEs not assigned to provisioning group • DPEs cannot be pinged from DHCP server • DPE interfaces not enabled for IPv6 provisioning • Provisioning group not enabled for IPv6 provisioning

Table 14-2 DOCSIS DHCPv6 Workflow Description (continued)

Workflow	Description	Potential Problems
Relay-Forw	The relay agent forwards the complete DHCPv6 message received from the cable modem to the DHCPv6 server. The relay agent adds relay agent message fields and options, such as: • Peer-address • Link-address • Interface ID	
Relay-Repl	The relay agent extracts the server response and forwards it to the cable modem, via the CMTS.	
Advertise	The DHCP server, in response to the Solicit message that it received from the cable modem, returns an Advertise message to indicate that it is available for DHCP service.	• init6(a) state • Routing issues between DHCP and CMTS
Request	On receiving the Advertise message, the cable modem sends a Request message to request configuration parameters, including IP addresses, from a specific server.	• init6(r) state • DHCP server did not provide all required parameters
Relay-Forw	The relay agent forwards the message to the DHCPv6 server.	
Relay-Repl	The relay agent extracts the server response and forwards it to the cable modem, via the CMTS.	
Reply	The CMTS forwards the REPLY message received from the DHCP server, containing assigned addresses and configuration parameters.	init6(i) state

Note DHCPv6 clients can be provisioned in the Rapid Commit mode. Rapid commit features a two-message exchange, instead of the usual four-message exchange. The two-message exchange involves a Solicit and a Reply, while the four-message exchange involves the Solicit–Advertise–Request–Reply. All these messages are wrapped in a Relay-Forw or Relay-Repl message if they go through a relay agent.

If rapid commit is enabled, the DHCP server responds to a Solicit (that is wrapped in a Relay-Forw message) with a Reply (that is wrapped in a Relay-Repl message). If you disable rapid commit, the DHCP server responds with an Advertise (that is wrapped in a Relay-Reply) message.

Table 14-2 DOCSIS DHCPv6 Workflow Description (continued)

Workflow	Description	Potential Problems
NS (DAD)	Once the DHCPv6 message exchange is complete, the cable modem confirms if the link-local address is not already in use via DAD. If it does not receive a response, then it deems the IP address acquisition to be successful.	
Provisioning Phase: ToD		
Request	After obtaining an IPv6 address, the cable modem requests the time of day from the RFC 868 time server. The IPv6 addresses for servers are supplied through DHCPv6 options.	init6(t) state - No route from time server (DPE) to modem
Response		
Provisioning Phase: TFTP		
TFTP-Get	The cable modem, using TFTP, downloads the configuration file. The IPv6 addresses for servers and the name of the configuration file are made available via DHCPv6.	• init6(o) state • Routing issues between CMTS and DPE • No route from TFTP server (DPE) to modem • DPE cache miss (static file, and RDU down or does not have file) • File not found at TFTP server (DPE) • DPE cache miss (dynamic file) • DPE IP validation failure (for example, the IP address of the device is not what was expected, the Dynamic Shared Secret is enabled on CMTS, or a hacker is spoofing as a DOCSIS modem)
TFTP RSP (config file)		Routing issues between DPE and CMTS
The cable modem is now provisioned for IPv6 operations.		

Prime Cable Provisioning Features for DOCSIS Configurations

This section describes Prime Cable Provisioning value-added features as they relate to the DOCSIS technology.

Dynamic Configuration TLVs

The DPE adds the following TLVs (Type Length Value) when it receives a TFTP request for dynamic DOCSIS configuration:

- TLV 19: TFTP Server Timestamp (optional)—Displays in the Configure DOCSIS Defaults page as the TFTP Time Stamp Option. See Table 11-2 for more information. This TLV requires NTP synchronization on CMTS and DPE.
- TLV 20 and TLV 59: TFTP Server Provisioned Modem Address for IPv4 and IPv6 (optional)—Displays in the Configure DOCSIS Defaults page as the TFTP Modem Address Option. See Table 11-2 for more information.

  Note The TFTP IP validation feature on the DPE is incompatible with the Cisco CMTS DSS feature. See DPE TFTP IP Validation, page 14-10. If DSS is set on the Cisco CMTS, you must ensure that the TFTP Server Provisioned Modem Address is disabled.

  In some cases, the CM might not accept the configuration and might stay in reject(IP) state. If it is the CMTS that acts as TFTP proxy, the DPE TLV 19/20 features must be disabled. This can be configured from the Admin UI.

  The CMTS DSS feature is more effective at theft prevention than the CableLabs standard CabeLabs TLV 19/20 features.

- TLV 6: CM MIC Configuration Setting (required)
- TLV 7: CMTS MIC Configuration Setting (required)—Displays in the Configure DOCSIS Defaults page as the CMTS Shared Secret. See Table 11-2 for more information.
- TLV 43.6.x: Extended CMTS MIC Configuration Setting (required)—Displays in the Configure DOCSIS Defaults page as the CMTS Shared Secret. See Table 11-2 for more information.

Note When configuring CMTS MIC, note the following CMTS IOS release dependencies:

- DOCSIS 2.0 CMTS MIC requires CMTS IOS 12.3BC when including TLV 39 or TLV 40.
- Certain CMTS IOS commands are assumed to be configured by Prime Cable Provisioning:
  - **ip dhcp relay information option**
  - **no ip dhcp relay information check**
  - **cable helper-address** *x.x.x.x. x.x.x.x* is the IP address of the Network Registrar DHCP server.

    In an IPv6 environment, you must use the following command instead of **cable helper-address**: **ipv6 dhcp relay destination** *ipv6-address* [*interface-type interface-number*]
  - **cable dhcp-giaddr primary**

DPE TFTP IP Validation

For dynamic configuration files, the DPE TFTP server verifies if the IP address of the TFTP client matches the expected DOCSIS cable modem IP address. If it does not match, the request is dropped. This feature is incompatible with the Cisco CMTS DMIC feature.

Use the **no service tftp** *1..1* **ipv4 | ipv6 verify-ip** command to disable the verification of requestor IP addresses on dynamic configuration TFTP requests. For detailed information, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.

Support for DOCSIS 1.0, 1.1, 2.0, and 3.0

Cisco Prime Cable Provisioning supports DOCSIS 1.0, 1.1, 2.0, and 3.0. For information describing the TLVs, see Template Grammar, page 19-15, and for a list of options that this Prime Cable Provisioning release supports in each DOCSIS version, see DOCSIS Option Support, page A-1.

Dynamic DOCSIS Version Selection

Prime Cable Provisioning can detect a cable modem’s DOCSIS version from an incoming DHCP request. It can also detect the DOCSIS version of the CMTS in one of two ways:

- By using the CMTS as a relay agent that transmits its DOCSIS version, using DHCPv4 Option 82 and DHCPv6 Option 17.
- From a customer-supplied source that provides a mapping of GIADDR to the CMTS DOCSIS version.

Using this DOCSIS version, Prime Cable Provisioning, if so configured, determines the optimum DOCSIS configuration file for the device. This is the lowest common DOCSIS version between the device and the CMTS. For example, if the device supports DOCSIS 2.0 and the CMTS supports DOCSIS 1.1, the DOCSIS 1.1 file is used.

Determining the DOCSIS Version of the Modem

Prime Cable Provisioning can detect a cable modem’s DOCSIS version from an incoming DHCP request, in which a string included in the Vendor Class Identifier field (Option 60) identifies the modem capabilities. For example, as in “docsis1.1:*xxxxxx*” where *xxxxxx* is an ASCII representation of the modem capabilities. The service-level selection extension uses the characters between “docsis” and the “*:xxxxxxx*” hex string as the DOCSIS version for the modem.

Determining the DOCSIS Version of the CMTS

Prime Cable Provisioning enables the CMTS to serve as a relay agent to provide the DOCSIS version of the CMTS. This feature is enabled via:

- DHCPv4 Relay Agent Option 82, which allows the CMTS to transmit (or advertise) specific capabilities of the CMTS. This option is a DOCSIS DHCP vendor-identifying option and carries the DOCSIS version of the CMTS.
- DHCPv6 Vendor-specific Information Option 17, which allows you to specify vendor-specific information. This option is carried in the Relay-forward and Relay-reply messages and transmits information between the DHCPv6 relay agent and the DHCPv6 server.

As in earlier versions, this Prime Cable Provisioning version also determines the DOCSIS version of the CMTS via the DHCP GIADDR field, which specifies the IP address of the CMTS interface. In this method, the service-level selection extension for DOCSIS modems looks for the */docsis/cmts/version/giaddrToVersionMap* property. The value of this property is the name of an external file containing a mapping of the GIADDR to the DOCSIS version.

You must name this mapping file *giaddr-docsis-map.txt* and add it to the RDU. You can add the *giaddr-docsis-map.txt* file to the RDU from the:

- API via the *Configuration.addFile()* call.
- Administrator user interface via **Configuration > Files**. See Adding Files, page 11-15.

The *giaddr-docsis-map.txt* file must include the necessary information in the following format:

```
IPv4_dotted_decimal_address_string,DOCSIS_version_string
```

- *IPv4_dotted_decimal_address_string*—Specifies the IP address of the CMTS interface.
- *docsis_version_string*—Identifies the DOCSIS version that the cable modem supports.

For example, if the CMTS interface has IP address 10.30.0.1 with DOCSIS version 1.0, the file would includes the following:

```
10.30.0.1 1.0
```

The service-level extension uses the GIADDR address contained in the DHCP packet to look up the DOCSIS version of the CMTS. If the GIADDR is not found in the mapping file, the extension uses the value of the */docsis/cmts/version/default* property for the DOCSIS version of the cable modem. The default value of this property is **1.0**.

To dynamically update the *giaddr-docsis-map.txt* file, edit it and replace it in the RDU via the *replaceExternalFile* API or via the administrator user interface.

Note If the properties for the DOCSIS version selection are not specified on the Class of Service, the original file is used, allowing for systematic upgrades across the network.

Selecting Service Level Based on DOCSIS Version

After the DOCSIS version of the modem and the CMTS are determined, the service-level selection extension determines the minimum DOCSIS version supported and configures the */docsis/version* property in the service level. The value of this property is set to the DOCSIS version string, such as 1.1.

Note You can specify the DOCSIS version using the Configuration File Utility. For more information, see Using Configuration File Utility for Template, page 19-33. This function that the file utility performs is different from RDU verification, during which the RDU DOCSIS Version Selector feature determines the latest DOCSIS version supported by a CMTS.

DOCSIS Configuration File Based on DOCSIS Version

Prime Cable Provisioning determines the filename of the DOCSIS configuration file that is to be sent to the modem using the DOCSIS version.

The following Class of Service properties are supported by the Prime Cable Provisioning administrator user interface and the API:

```
/cos/docsis/file/1.0
/cos/docsis/file/1.1
/cos/docsis/file/2.0
/cos/docsis/file/3.0/IPv4
/cos/docsis/file/3.0/IPv6
```

Optionally, you can add these properties to a DOCSIS Class of Service to associate a DOCSIS configuration filename with a particular DOCSIS version. Each of these properties, when set, causes the RDU to establish a database relationship between the Class of Service and the file named by the property value, as is done for the existing DOCSIS configuration filename property.

If the DOCSIS version property is present, Prime Cable Provisioning forms a property name by appending the DOCSIS version string that is given by that property value to the name of the property that provides the DOCSIS configuration filename:

```
/cos/docsis/file/docsis_version_string
```

The service-level extension looks for this property name in the property hierarchy for the modem. When the DOCSIS version property is found, it uses the property value as the DOCSIS configuration filename. If the DOCSIS version property is not found, Prime Cable Provisioning uses the DOCSIS configuration filename property without the DOCSIS version suffix and supplies the filename to specify in the device configuration.

CHAPTER 15

Lease Query

The Prime Cable Provisioning RDU queries Network Registrar for the IP address of devices using the DHCP lease query protocol. Prime Cable Provisioning then uses this information for device disruption and for reporting details of both IPv4 and IPv6 devices.

Prime Cable Provisioning supports the following configurations:

- Lease Query Autoconfiguration, page 15-1
- Lease Query Source IP Address, page 15-2

Lease Query Autoconfiguration

The RDU performs name resolution to determine the IP addresses of Network Registrar servers to which it sends lease queries. In case of a DNS failure, lease queries can fail. In Cisco Prime Cable Provisioning, you can directly configure the IP addresses of Network Registrar servers in a provisioning group to which the RDU must send lease query requests.

When you enable automatic configuration, the RDU adjusts its lease query configuration to set both IPv4 and IPv6 address lists from the Network Registrar servers in the provisioning group. It performs this task after comparing the current information registered with the server to the information stored in the RDU database. If the Prime Cable Provisioning Network Registrar extensions have moved from one provisioning group to another, the lease query configuration is changed to delete:

- IP addresses that are present in the lease query configuration on the previous provisioning group object.
- IP addresses that are no longer present in the IP address list.

The RDU searches the lease query configuration to verify if the provisioning group is configured to use the specified extension. If the provisioning group is not configured to use the extension, the RDU selects an address from the addresses being registered with the Network Registrar server and adds to the provisioning group’s lease query configuration.

If you disable this autoconfiguration, the RDU does not change its lease query configuration upon registering with the Network Registrar server. This feature is, by default, enabled.

To enable or disable autoconfiguration of lease query addresses in a provisioning group, you can set the LeaseQuery AutoConfig option from the administrator user interface. See Monitoring Provisioning Groups, page 21-3.

Lease Query Source IP Address

In earlier Cisco BAC versions, the lease query feature relied on the operating system to select the source interface and the source port for sending lease query requests. While this is the default behavior in this release, you can also configure the RDU to send lease query requests using a specific interface.

Configuring Lease Query

Prime Cable Provisioning, by default, binds to the IP addresses and ports that are described in Table 15-1.

Table 15-1 Lease Query Address for Binding

Protocol	IP Address	Port
IPv4	Wildcard[1]	67
IPv6	Wildcard	547

[1] The wildcard is a special local IP address. It usually means “any” and can only be used for bind operations.

If port 547 and port 67 are available on the RDU, you need not perform any special configuration to send lease query requests. If while installing the RDU, the installation program detects that either of these ports is being used by another process, it recommends that you use the dynamic ports that the operating system selects.

For example:

```
DHCPv4/DHCPv6 lease query port(s) (Udp/67 and Udp/547) is in use.
Configuring the RDU to use a dynamic port for DHCPv4/DHCPv6 lease query.
```

The installation program automatically enables selection of dynamic ports by setting zero values to the following properties in the *BPR_HOME/rdu/conf/rdu.properties* file:

```
/cnrQuery/clientSocketAddress=0.0.0.0:0
/cnrQuery/ipv6/clientSocketAddress=[::]:0
```

You can also configure the IP address and port of your choice for lease query communication using the same properties. For example:

```
/cnrQuery/clientSocketAddress=10.1.2.3:166
/cnrQuery/ipv6/clientSocketAddress=[2001:0DB8:0:0:203:baff:fe12:d5ea]:1547
```

Using these properties, the RDU binds to the IP address and the port that you specify.

Note When you manually change properties in the *rdu.properties* file, remember to restart the RDU. Use the **/etc/init.d/bprAgent restart rdu** command.

Note The names of the below properties, earlier prefixed with */cnrQuery* have been changed to start with */dhcpLeaseQuery*:
/cnrQuery/retries -> /dhcpLeaseQuery/retries (default:1)
/cnrQuery/timeout -> /dhcpLeaseQuery/timeout (default: 500)
/cnrQuery/requireAllAnswers -> /dhcpLeaseQuery/requireAllAnswers (default: false)

Configuring Cisco Prime Cable Provisioning as Relay Agent for Lease Query

You can configure Prime Cable Provisioning to act as a relay agent. The relay agent option is:

- Enabled by default for IPv4
- Disabled by default for IPv6

For IPv4 Lease Query

For Prime Cable Provisioning to act as a relay agent for an IPv4 lease query, Prime Cable Provisioning provides the GIADDR (the IP address to which the DHCP server should reply) in the Lease Query Request packet. The RDU, by default, uses the primary IP address on the machine for this purpose.

Note Ensure that all DHCP servers in your deployment can reach this IP address. Also, the IP address that you use in this property must exist on the machine on which you have installed the RDU.

To change the IP address used in the GIADDR field, you must change the value of the */cnrQuery/giaddr* property in the *rdu.properties* file. For example, if you wanted to change the GIADDR to 10.10.10.1, you would add:

```
/cnrQuery/giaddr=10.10.10.1
```

When you manually change properties in the *rdu.properties* file, remember to restart the RDU using the **/etc/init.d/bprAgent restart rdu** command.

For IPv6 Lease Query

To configure Prime Cable Provisioning to act as a relay agent for an IPv6 lease query, you must include the following properties in the *rdu.properties* file.

/cnrQuery/ipv6/linkAddress=IPv6 address

/cnrQuery/ipv6/peerAddress=IPv6 address

For example:

```
/cnrQuery/ipv6/linkAddress=2001:0DB8:0:0:203:baff:fe12:d5ea
/cnrQuery/ipv6/peerAddress=2001:0DB8:0:0:203:baff:fe12:d5ea
```

Note The values that you enter for link address and peer address depend on the network configuration in which Prime Cable Provisioning and Network Registrar are operating. In simple cases, you must set the link address and peer address to an IPv6 address of the RDU host. This IPv6 address must be routable to Network Registrar.

Restart the RDU using the **/etc/init.d/bprAgent restart rdu** command.

Examples

This example features output for an IPv6 lease query request with the relay agent option enabled:

```
rdu.example.com: 2007 10 18 19:40:30 EDT: %BAC-RDU-7-DEBUG_DHCP_IF_IPV6:
PACE-2:ServerBatch[Batch:rdu.example.com/10.10.10.1:1b994de:115b52abeb4:80000278]:
Peer[rdu.example.com:33743]: Querying single prov group for DUID
[00:03:00:01:23:45:67:89:98:56] via DHCPv6 LEASEQUERY packet [version V6, message-type 12,
hop-count 0, link-address 2001:0DB8:0:0:203:baff:fe12:d5ea, peer-address
2001:0DB8:0:0:203:baff:fe12:d5ea, (relay_msg (9) option (52 bytes) version V6,
message-type 14, transaction-id 13401290, (client-identifier (1) option (9 bytes)
00:11:22:33:44:55:66:77:88), (lq-query (44) option (31 bytes) query-type 2, link-address
0:0:0:0:0:0:0:0, (client-identifier (1) option (10 bytes)
00:03:00:01:23:45:67:89:98:56)))]
```

Enabling AIC Echo

Using the AIC Echo option, you can configure Network Registrar to send its reply to the source port of the client from which the request was made, instead of the standard port.

For example, if a client whose IP address is 10.1.1.1 forwards a request using port 1456 and AIC Echo is disabled on the server, then the server returns the reply to the standard client port. Depending on the protocol stack, the standard client port is:

- 67 for IPv4
- 546 for IPv6

If AIC Echo is enabled, the reply is forwarded to port 1456.

If you are requesting IPv4 lease queries, AIC Echo is disabled by default. This option is used only if the default IPv4 binding port is changed.

If you are requesting IPv6 lease queries, then AIC ECHO is enabled by default. However, because IPv6 lease query messages are not relayed by default, this option is used to get lease query responses back to port 547 instead of to standard client port 546.

Debugging Lease Query

Using the Info-level logging (6-Information) at the RDU, you can view important details related to lease-query processing. (To set the log level at the RDU, see Using the RDU Log Level Tool, page 25-5.)

For debugging the lease query feature, you can use these properties:

- *dhcpleasequeryv4*—Debugs IPv4 lease queries
- *dhcpleasequeryv6*—Debugs IPv6 lease queries

IPv6 Lease Query Use Cases

This section describes the following IPv6 lease query use cases:

- One lease per client across all (two) Network Registrar servers in a provisioning group
- Multiple leases per client across all (two) Network Registrar servers in a provisioning group
- Multiple leases per client on a single Network Registrar server
- Leases for devices with delegated prefix

One lease per client across all (two) Network Registrar servers in a provisioning group

With no failover protocol defined for IPv6 yet, typically, only one Network Registrar server in a provisioning group has lease information for a client. In this case, where there are two Network Registrar servers in a provisioning group, the RDU sends lease query requests to both the servers, but receives a response only from one. The IP address provided in that response will be used.

You can view this IP address from the:

- Administrator user interface, on the **Devices > Device Details** page.
- API, using the `client-ipaddress` attribute in the lease query map.

Multiple leases per client across all (two) Network Registrar servers in a provisioning group

In rare instances, when both Network Registrar servers in a provisioning group have the lease for the same client, both servers respond with a lease query reply. In this case, as per the DHCPv6 Leasequery draft, the response with the most recent OPTION_CLT_TIME (client-last-transaction-time) is used.

Multiple leases per client on a single Network Registrar server

If a client has leases on two different links on the same server, Network Registrar includes all the link addresses in the OPTION_LQ_CLIENT_LINK option while replying. Prime Cable Provisioning then queries Network Registrar for each individual link and gets all the IP addresses. With this list, Prime Cable Provisioning uses the first IP address that is not a loopback or multicast address for device disruption.

From the administrator user interface, you can view the list of IP addresses obtained in this process on the **Devices > Device Details** page.

Leases for devices with delegated prefix

You can send lease query requests for devices with assigned IP addresses, or a delegated prefix, or both.

From the administrator user interface, you can view the IP addresses and prefixes on the **Devices > Device Details** page. To get this IP address via the API, use the `iaprefix` attribute in the lease query map.

CHAPTER 16

PacketCable Provisioning

This chapter describes the provisioning flow in a Cisco Prime Cable Provisioning PacketCable deployment.

Automatic FQDN Generation

When configuring the PacketCable voice technology, a fully qualified domain name (FQDN) must reside in the Prime Cable Provisioning database for each voice device, because the KDC queries the registration server for that FQDN. The Prime Cable Provisioning automatic FQDN generation feature is not limited to any single voice technology; it can be used by any Prime Cable Provisioning technology.

Automatically Generated FQDN Format

Prime Cable Provisioning automatically generates FQDNs using the MAC address of a device or using the DHCP Unique Identifier (DUID) of an IPv6 device.

An automatically generated FQDN using a MAC address follows this format:

```
prefix{htype-hlen-aa-bb-cc-dd-ee-ff | 00:00:00:00:00:00:00:00}suffix.domain
```

- *prefix*, *suffix*, and *domain*—Identify the information that you set from the Prime Cable Provisioning administrator user interface or the provisioning API.
- *htype*, *hlen*, and aa-bb-cc-dd-ee-ff—Identify the device MAC address. For example, 1,6,aa-bb-cc-dd-ee-ff.
- 00:00:00:00:00:00:00:00—Identifies the DUID of an IPv6 device.

The entry of a prefix and suffix property is optional. If you do not specify these properties, and a hostname is not specified during PacketCable MTA provisioning and, if neither the prefix nor suffix property is defined in the Prime Cable Provisioning property hierarchy, the device MAC address or the device DUID followed by the domain name is used as the generated FQDN.

The FQDN format changes if you specify only the:

- Prefix and the device ID:
  ```
  prefix{htype-hlen-aa-bb-cc-dd-ee-ff | 00:00:00:00:00:00:00:00}.domain
  ```
- Suffix and the device ID:
  ```
  {htype-hlen-aa-bb-cc-dd-ee-ff | 00:00:00:00:00:00:00:00}suffix.domain
  ```

For example:

- A device with prefix **aaa**, suffix **bbb**, and MAC address **1,6,aa:bb:cc:dd:ee:ff** will have this FQDN generated:

  ```
  aaa1-6-aa-bb-cc-dd-ee-ffbbb.domain
  ```

- A device with only MAC address **(1,6,aa:bb:cc:dd:ee:ff)** will have this FQDN generated:

  ```
  1-6-aa-bb-cc-dd-ee-ff.domain
  ```

- A device with prefix **aaa**, suffix **bbb**, and DUID **00:00:00:00:00:00:00:00** will have this FQDN generated:

  ```
  aaa00-00-00-00-00-00-00-00bbb.domain
  ```

- A device with only DUID **00:00:00:00:00:00:00:00** will have this FQDN generated:

  ```
  00-00-00-00-00-00-00-00-00-aa.domain
  ```

- A device with prefix **aaa** and MAC address **1,6,aa:bb:cc:dd:ee:ff** will have this FQDN generated:

  ```
  aaa1-6-aa-bb-cc-dd-ee-ff.domain
  ```

- A device with suffix **bbb** and MAC address **1,6,aa:bb:cc:dd:ee:ff** will have this FQDN generated:

  ```
  1-6-aa-bb-cc-dd-ee-ffbbb.domain
  ```

When configuring for PacketCable and other technologies, the domain name property must also be configured. If you do not specify a domain name while provisioning a PacketCable MTA, the Prime Cable Provisioning property hierarchy is searched and, if it is not found, the MTA is not provisioned.

If you do specify the domain name during MTA provisioning, that domain name is used regardless of the domain name property that is specified in the Prime Cable Provisioning property hierarchy.

Properties for Automatically Generated FQDNs

Properties can be defined at any acceptable point in the Prime Cable Provisioning property hierarchy. You can use the System Defaults, Technology Defaults, DHCP Criteria, or Class of Service to accomplish this, and you can also do this at the device level.

FQDN Validation

There are a few things to consider when entering the information that is used to generate an FQDN. These include:

- Use only valid alphanumeric characters in the generated FQDN.
- Keep the length of each label (characters between the dots in the generated FQDN) to fewer than 63 characters.
- Do not allow the overall length of the generated FQDN to exceed 254 characters.

Note The FQDN supports host and domain names as per RFC1035.

Sample Automatic FQDN Generation

This section provides an example of creating an automatically generated FQDN.

Step 1 Choose the appropriate Class of Service, and set the */fqdn/domain* property value to the DNS domain for all devices using this Class of Service.For the purposes of this example, assume that the domain in use is example.com, and that you want to provision a set of PacketCable devices into that domain.

Note If you do not specify a domain, devices in the Class of Service will not receive a DHCP configuration from Prime Cable Provisioning.

Step 2 Click **Submit**.

In this example, a device with MAC address 1,6,aa:bb:cc:dd:ee:ff will yield an automatically generated FQDN of 1-6-aa-bb-cc-dd-ee-ff.example.com.

Additionally, enable the Automatic FQDN Generation radio button in the device’s default configuration. See Chapter 11, “Configuring Defaults”.

CHAPTER 17

Provisioning CPEs in Promiscuous Mode

When a device boots, it requests a configuration from Prime Cable Provisioning and it is this configuration that determines the level of service for the device. During this process, the DHCP server requests the RDU to build a configuration for the device. The RDU generates a configuration and forwards it to all the DPEs that service the provisioning group that the device belongs to. The DPEs can now provide the device with its configuration without going to the RDU.

Prime Cable Provisioning automatically recognizes devices, assigns the appropriate class of service, dynamically creates and generates device configuration files, and activates subscribers.
Prime Cable Provisioning provides a single device management platform to support multiple technologies including DOCSIS, and PacketCable.

Prime Cable Provisioning allows service providers to implement either or both of the following workflow models:

- Preprovisioning: Devices are assigned to subscribers and recorded in advance in the provisioning application. When subscribers plug them in, Prime Cable Provisioning automatically assigns the appropriate service level and activates them.
- Autoprovisioning: When subscribers self-register for service, subscriber devices are captured and recorded in the provisioning application. Subscribers are required to register for service before Prime Cable Provisioning configures the device and activates the service.

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

- DHCP IP address selection
- Bandwidth
- Data rates
- Flow control
- Communication speeds
- Level of service

A configuration can contain DHCP configuration and TFTP files for any device. When you install and boot an unprovisioned device, it is assigned a default technology-specific configuration. You can change the default configuration for each technology that Prime Cable Provisioning supports.

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.
- A DPE is repopulating its cache.

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

Promiscuous Access for Devices

This section describes the objects and the properties that are used to control the configuration of devices that are granted promiscuous access.

Devices are said to be given promiscuous access if they are allowed to boot and be configured without being preregistered in Prime Cable Provisioning. Promiscuous access is typically used for devices, such as computers, that appear behind a registered DOCSIS modem. If promiscuous access is not enabled for unknown devices behind a registered DOCSIS modem, the devices receive the default service level.

To grant promiscuous access to a device, you must:

- Enable or disable the promiscuous policy for unknown devices of a given type. Devices for which promiscuous access is enabled are configured according to the policy, instead of receiving the default configuration.
- Specify the Class of Service meant for unknown devices of a given type if the devices are to be given promiscuous access.
- Specify the DHCP Criteria meant for unknown devices of a given type if the devices are to be given promiscuous access.

Configuring Promiscuous Access

Table 17-1 describes the ways in which you can configure a promiscuous policy for a device. The Provisioning group and Technology-specific promiscuous mode configuration are not possible through the Admin UI.

Table 17-1 Configuring Promiscuous Access for Devices

Configuration Scope	Using API Calls...
Provisioning group of relay agent—For example, you can configure promiscuous access to allow computers only behind any registered relay agent device in a specific provisioning group.	*getProvGroupDetails* *changeProvGroupProperties*
Class of Service object of relay agent—For example, you can configure promiscuous access to allow computers only behind DOCSIS modems that are associated with a specific Class of Service.	*addClassOfService* *changeClassOfServiceProperties* *getClassOfServiceProperties*
DHCP Criteria of relay agent—For example, you can configure promiscuous access to allow computers only behind DOCSIS modems that are associated with a specific DHCP Criteria.	*addDHCPCriteria* *changeDHCPCriteriaProperties* *getDHCPCriteriaDetails*
Technology-specific defaults—For example, you can configure promiscuous access for computers behind DOCSIS modems using the technology defaults for DOCSIS modems.	*changeDefaults* *getDefaults*
System-wide defaults—Global system defaults	*changeSystemDefaults* *getSystemDefaults*

Promiscuous Access and Property Hierarchy

You can configure a promiscuous policy on a number of objects in Prime Cable Provisioning. It is, therefore, important to understand the settings that take precedence. While the policy is configured using properties, the precedence of properties is determined by the Prime Cable Provisioning property hierarchy. The first object in the property hierarchy that has a specific property determines the value that Prime Cable Provisioning is to use.

Prime Cable Provisioning looks up the properties of the promiscuous policy in the property hierarchy of the device's relay agent. For example, for a computer, Prime Cable Provisioning looks up the promiscuous policy settings in the property hierarchy of the cable modem, which functions as a relay for the computer. For more details about property hierarchy, see Property Hierarchy, page 20-7. For more details about promiscuous policy see Properties for Configuring Promiscuous Policy, page 17-4.

Note When you set the promiscuous policy using technology defaults, the properties must be set on objects associated with the relay agent, not the target device type. For example, to enable promiscuous access for computers behind a DOCSIS modem, you can enable promiscuous access on technology defaults for the DOCSIS modem, but not on technology defaults for computers.

The promiscuous policy properties specify the Class of Service, the DHCP Criteria, and whether promiscuous access is enabled or disabled for each device type. If promiscuous mode is enabled for a device, but a search of the device's relay agent hierarchy does not locate a match of the Class of Service or DHCP Criteria properties, the default Class of Service or DHCP Criteria for non-promiscuous access are used. For example, if Prime Cable Provisioning is configured to grant promiscuous access to computers, but it cannot locate a promiscuous Class of Service, DHCP Criteria, or both, then it uses the default Class of Service, DHCP Criteria, or both for the computer.

The Class of Service and the DHCP Criteria defaults are configured on the technology defaults of the target device (instead of its relay agent) using these properties:

- Class of Service—*/default/classOfService*

  The API constant is `TechnologyDefaultsKeys.DEFAULT_CLASS_OF_SERVICE`.
- DHCP Criteria—*/default/dhcpCriteria*

  The API constant is `TechnologyDefaultsKeys.DEFAULT_DHCP_CRITERIA`.

Generating Configurations for Promiscuous Devices

The configuration for promiscuous devices is generated under these conditions:

- The device first appears online and is given promiscuous access.
- An out-of-date DPE is populating its cache and requests configurations for a specific provisioning group.
- Regeneration of the configuration is explicitly requested for the device via the API call *regenConfigs*.
- Configuration of the relay agent device for a promiscuous access device is being regenerated.
- Changes to the promiscuous policy (or other configuration changes) prompt the Prime Cable Provisioning Configuration Regeneration Service (CRS) service to regenerate configurations of affected devices.

Every time a configuration for a promiscuous device is regenerated, it uses the newly configured promiscuous policy (for example, the Class of Service currently specified for promiscuous computers). However, if the Class of Service or DHCP Criteria of a device is changed via the API after the device appears online as a promiscuous device, then from then on, the device is not considered promiscuous and is unaffected by any changes that you make to the promiscuous policy. The device is henceforth considered registered.

Properties for Configuring Promiscuous Policy

To configure promiscuous access for devices, you must configure the properties associated with specific device types that Prime Cable Provisioning supports. You can enable or disable promiscuous access for the device types.

- Enabled—Enables promiscuous access for devices within the scope associated with the API call that Table 17-1 describes.
- Disabled—Disables promiscuous access. If the property does not exist, the default is the disabled setting.

See Table 17-2 for a list of properties on which you configure promiscuous access.

Promiscuous policy properties are divided into read-write and read-only properties. This section describes the read-write and read-only properties that you must configure to enable promiscuous access for devices and those that you set to select Class of Service or DHCP Criteria for these devices.

Read-Write Properties

Note Table 17-1 describes the applicable API calls for all the properties that are described in this section.

Table 17-2 describes the properties that you can use to enable promiscuous access.

Table 17-2 Properties for Enabling Promiscuous Access

Property Name	Description
/promiscuousMode/enable/Computer	Sets a Boolean value of “true” or “false” in the relay agent’s property hierarchy: • true—Enables promiscuous access for computers behind such a relay. • false—Disables promiscuous access for computers behind such a relay. If the property does not exist in the relay agent's property hierarchy, promiscuous access for computers behind such a relay is disallowed and the devices receive default access.
	API Constant `PolicyKeys.COMPUTER_PROMISCUOUS_MODE_ENABLED`

Table 17-2 Properties for Enabling Promiscuous Access (continued)

Property Name	Description
/promiscuousMode/enable/ PacketCableMTA	Sets a Boolean value of “true” or “false” in the relay agent’s property hierarchy: • true—Enables promiscuous access for PacketCable MTAs behind such a relay. • false—Disables promiscuous access for PacketCable MTAs behind such a relay. If the property does not exist in the relay agent's property hierarchy, promiscuous access for PacketCable MTAs behind such a relay is disallowed and the devices receive default access.
	API Constant `PolicyKeys.PACKET_CABLE_MTA_PROMISCUOUS_MODE_ENABLED`
/promiscuousMode/enable/STB	Sets a Boolean value of “true” or “false” in the relay agent’s property hierarchy: • true—Enables promiscuous access for STBs behind such a relay. • false—Disables promiscuous access for STBs behind such a relay. If the property does not exist in the relay agent's property hierarchy, promiscuous access for STBs behind such a relay is disallowed and the devices receive default access.
	API Constant `PolicyKeys.STB_PROMISCUOUS_MODE_ENABLED`
/promiscuousMode/enable/ CableHomeWanData	Sets a Boolean value of “true” or “false” in the relay agent’s property hierarchy: • true—Enables promiscuous access for CableHome WAN-Data devices behind such a relay. • false—Disables promiscuous access for CableHome WAN-Data devices behind such a relay. If the property does not exist in the relay agent's property hierarchy, promiscuous access for WAN-Data devices behind such a relay is disallowed and the devices receive default access.
	API Constant `PolicyKeys.CABLE_HOME_WAN_DATA_PROMISCUOUS_MODE_ENABLED`

Table 17-2 Properties for Enabling Promiscuous Access (continued)

Property Name	Description
/promiscuousMode/enable/ CableHomeWanMan	Sets a Boolean value of "true" or "false" in the relay agent's property hierarchy: • true—Enables promiscuous access for CableHome WAN-MAN devices behind such a relay. • false—Disables promiscuous access for CableHome WAN-MAN devices behind such a relay. If the property does not exist in the relay agent's property hierarchy, promiscuous access for WAN-MAN devices behind such a relay is disallowed and the devices receive default access.
	API Constant `PolicyKeys.CABLE_HOME_WAN_MAN_PROMISCUOUS_MODE_ENABLED`
/promiscuousMode/enable/	Use this property to enable or disable promiscuous access for new types of devices by appending the property name with the name of a valid device type. Sets a Boolean value of "true" or "false."
	API Constant `PolicyKeys.PROMISCUOUS_MODE_PREFIX`

Table 17-3 describes the read-write properties that you must configure to select Class of Service for devices granted promiscuous access.

Table 17-3 Promiscuous Access–Read-Write Properties for Class of Service

Class of Service Property Name	Description
/provisioning/cpeClassOfService/ Computer	Specifies the name of an existing Class of Service that will be selected for promiscuous computers
	API Constant `PolicyKeys.COMPUTER_CLASS_OF_SERVICE`
/provisioning/cpeClassOfService/ PacketCableMTA	Specifies the name of an existing Class of Service that will be selected for promiscuous PacketCable MTAs
	API Constant `PolicyKeys.PACKET_CABLE_MTA_CLASS_OF_SERVICE`
/provisioning/cpeClassOfService/ STB	Specifies the name of an existing Class of Service that will be selected for promiscuous set-top boxes
	API Constant `PolicyKeys.STB_CLASS_OF_SERVICE`
/provisioning/cpeClassOfService/ CableHomeWanMan	Specifies the name of an existing Class of Service that will be selected for promiscuous CableHome WAN-Data devices
	API Constant `PolicyKeys.CABLEHOME_WAN_DATA_CLASS_OF_SERVICE`

Table 17-3 Promiscuous Access–Read-Write Properties for Class of Service (continued)

Class of Service Property Name	Description
/provisioning/cpeClassOfService/ CableHomeWanData	Specifies the name of an existing Class of Service that will be selected for promiscuous CableHome WAN-MAN devices **API Constant** `PolicyKeys.CABLEHOME_WAN_MAN_CLASS_OF_SERVICE`
/provisioning/cpeClassOfService/	Specifies an existing Class of Service that will be selected for devices of the specified device type. Use this property name with a valid device type name. You can use this property for custom device types. **API Constant** `PolicyKeys.PROMISCUOUS_COS_PREFIX`

Table 17-4 describes the read-write properties that you must configure to select DHCP Criteria for devices granted promiscuous access.

Table 17-4 Promiscuous Access–Read-Write Properties for DHCP Criteria

DHCP Criteria Property Name	Description
/provisioning/cpeDhcpCriteria/ Computer	Specifies the name of an existing DHCP Criteria object that will be selected for promiscuous computers **API Constant** `PolicyKeys.COMPUTER_DHCP_CRITERIA`
/provisioning/cpeDhcpCriteria/ PacketCableMTA	Specifies the name of an existing DHCP Criteria object that will be selected for promiscuous PacketCable MTAs **API Constant** `PolicyKeys.PACKET_CABLE_MTA_DHCP_CRITERIA`
/provisioning/cpeDhcpCriteria/STB	Specifies the name of an existing DHCP Criteria object that will be selected for promiscuous set-top boxes **API Constant** `PolicyKeys.STB_ DHCP_CRITERIA`
/provisioning/cpeDhcpCriteria/ CableHomeWanData	Specifies the name of an existing DHCP Criteria object that will be selected for promiscuous CableHome WAN-Data devices **API Constant** `PolicyKeys.CABLEHOME_WAN_DATA_ DHCP_CRITERIA`
/provisioning/cpeDhcpCriteria/ CableHomeWanMan	Specifies the name of an existing DHCP Criteria object that will be selected for promiscuous CableHome WAN-MAN devices **API Constant** `PolicyKeys.CABLEHOME_WAN_MAN_ DHCP_CRITERIA`

Table 17-4 Promiscuous Access–Read-Write Properties for DHCP Criteria (continued)

DHCP Criteria Property Name	Description
/provisioning/cpeDhcpCriteria/	Specifies an existing DHCP Criteria object that will be selected for devices of the specified device type. Use this property name with a valid device type name. You can use this property for custom device types.
	API Constant
	`PolicyKeys.PROMISCUOUS_DC_PREFIX`

Read-Only Properties

Table 17-5 covers read-only promiscuous properties that you must configure to select the Class of Service and the DHCP Criteria for devices. Together with the read-write properties specified in the previous section, these read-only properties help determine the current system configuration.

Table 17-5 Promiscuous Access–Read-Only Properties

Property Name	Description	
/isSystemWide/default/promiscuous	Returns a "true" value if a given Class Of Service or DHCP Criteria object is referenced as system-wide default for promiscuous devices.	
	Applicable API Calls	**API Constant**
	getClassOfServiceProperties *getDHCPCriteriaDetails*	`PolicyKeys.IS_SYSTEM_WIDE_` `DEFAULT_PROMISCUOUS`
/referencedBy/deviceTypes/ forPromiscuousDevices	Returns a list of Device Type object (technology) names that reference a given Class of Service or DHCP Criteria object in promiscuous policy properties	
	Applicable API Calls	**API Constant**

	getClassOfServiceProperties
	getDHCPCriteriaDetails
		`PolicyKeys.REFERENCED_BY_`
		`DEVICE_TYPE_FOR_`
		`PROMISCUOUS_DEVICES`

/related/classesOfService	Returns a list of Class of Service object names that are used by a given Class of Service or DHCP Criteria object in promiscuous policy properties **Note** You can use this property as a shortcut to obtain the Class of Service list. You can also obtain this list by reading individual promiscuous policy properties set on this object.	
	Applicable API Calls	**API Constant**
	getClassOfServiceProperties *getDHCPCriteriaDetails*	`PolicyKeys.RELATED_CLASS_` `OF_SERVICE`

Table 17-5 Promiscuous Access–Read-Only Properties (continued)

<table>
<tr><th>Property Name</th><th colspan="2">Description</th></tr>
<tr><td rowspan="3"><i>/related/dhcpCriteria</i></td><td colspan="2">Returns a list of DHCP Criteria object names that are used by a given Class of Service or DHCP Criteria object in promiscuous policy properties
<b>Note</b> You can use this property as a shortcut to obtain the DHCP Criteria list. You can also obtain this list by reading individual promiscuous policy properties set on this object.</td></tr>
<tr><td><b>Applicable API Calls</b></td><td><b>API Constant</b></td></tr>
<tr><td><i>getClassOfServiceProperties</i>
<i>getDHCPCriteriaDetails</i></td><td><code>PolicyKeys.RELATED_DHCP_ CRITERIA</code></td></tr>
</table>

Custom Policy for Promiscuous Devices

You can configure promiscuous policy for a device using the properties specified in the above section. When additional logic is required, however, you can implement custom logic using extensions and custom properties. Custom properties allow for the definition of new properties, which can then be stored on any object via the API.

To augment the promiscuous device policy, you can use these extensions:

- Device Detection—Determines the technology type of the device (usually based on DHCP request data). Information that this extension detects is placed in a Device Detection Context that other extensions then use.
- Service-Level Selection—Selects the appropriate Class of Service and DHCP Criteria objects for a device. The promiscuous policy properties determine the Class of Service and DHCP Criteria for devices with promiscuous access.
- Configuration Generation—Generates the configuration for a device and, if necessary, for the devices behind it. Configurations are regenerated for promiscuous devices behind the relay agent based on the policy that the service-level selection extension selects. You may need to change the extension only if you want to augment the default behavior of regenerating configuration for devices behind a relay agent.

CHAPTER 18

Provisioning Devices Using Admin UI

This chapter describes how to configure devices using the Prime Cable Provisioning Admin UI. Use the Devices menu to provision and manage various devices. You can:

- Search for a specific device or for a group of devices that share criteria that you specify. See Searching for Devices, page 18-1.
- Add, modify, or delete devices in the RDU database. See:
  - Adding Device Records, page 18-9
  - Deleting Devices, page 18-10
  - Deleting Devices, page 18-10
- View device data, such as configuration, and properties. See Viewing Device Details, page 18-4.
- Regenerate device configurations. See Regenerating Device Configurations, page 18-10.
- Relate and unrelate any device to a specific group. See Relating and Unrelating Devices, page 18-11.
- Reset, or reboot, a device. See Resetting Devices, page 18-12.

Device Management

The Manage Devices page appears when you click the **Devices** tab on the primary navigation bar. You can also click the Devices link on the Main Menu to get to the Manage Devices page.

Searching for Devices

Using Prime Cable Provisioning, you can search for device information in a number of ways.

To select the search type, from the Manage Devices page, click the Search Type drop-down list. Subsequent search pages contain screen components that may be unique to the search type that you selected.

The Manage Devices page uses two separate but related areas to generate search results that allow you to manage the devices in your network. These areas are the:

- Search Type drop-down list, which defines which search to perform.
- An additional value field, which qualifies the search type that you selected. These fields include IP Address, MAC Address or MAC Address wildcard, Group Name (Group Type), and Owner ID.

Some searches that you can perform allow the use of a wildcard character (*) to enhance the search function. Prime Cable Provisioning provides specific wildcards for each search, as described in Table 18-1.

Note We do not recommend using the wildcard search (*) in systems that support hundreds of thousands, or more, devices. Such a search can return thousands of results, and use extensive system resources so as to impact performance.

Table 18-1 Searches Supported for Device Management

Menu Search	Search Type Option
DUID Search	Searches using the DHCP Unique Identifier (DUID) of a device in an IPv6 environment. The accepted format for a DUID is a two-octet type code represented in network byte order, followed by a variable number of octets that make up the identifier; for example, 00:03:00:01:02:03:04:05:07:a0. See Troubleshooting Devices by Device ID, page 26-2, for information on how you can effectively use this search criteria.
FQDN Search	Searches by using the fully qualified domain name (FQDN) associated with the device that is assigned by the DNS Server. This search is especially useful when the device MAC address is unknown. For example, **www.myhost.example.com** is a fully qualified domain name. Where **myhost** identifies the host, **example** identifies the second-level domain, and **.com** identifies the third-level domain. To search for a device with the FQDN IGW-1234.EXAMPLE.COM, you can try: • *.example.com • *.com • *
IP Address Search	Searches by returning all devices on the network that currently have the specified DHCP leased IP address. Wildcard searches are not supported. You must enter the complete IP address. For example, to search for a device with the IP address 10.10.10.10, you must enter 10.10.10.10.
MAC Address Search	Searches by using the precise MAC address for a specific modem or all devices with a specific vendor-prefix that unambiguously identifies the equipment vendor. The vendor-prefix is the first three octets of the MAC address. For example, for MAC address 1,6,aa:bb:cc:dd:ee:ff, the vendor-prefix is “aa:bb:cc”. Therefore, if you perform a MAC address search, you can identify the manufacturer and the type of device. See Troubleshooting Devices by Device ID, page 26-2, for information on how you can effectively use this search criteria.
Group Search	Searches devices that are part of a particular group or group type.

Table 18-1 Searches Supported for Device Management (continued)

Menu Search	Search Type Option
Owner ID Search	Searches by using the owner ID associated with the device. The owner ID may identify the service subscriber's account number, for example. This search function does not support wildcard searching. You must enter the complete owner ID. For example, to search for a device with the owner ID 100000000000xxxxx, you must enter 100000000000xxxxx.
Provisioning Group Search	Searches by using the provisioning group to which the device belongs.
Registered Class of Service Search	Searches by using the Class of Service that a device has been provisioned with.
Registered DHCP Criteria Search	Searches for devices that belong to certain DHCP Criteria.
Related Class of Service Search	Searches by using both the registered and selected Class of Service.
Related DHCP Criteria Search	Searches using both the registered and selected DHCP Criteria.
Selected Class of Service Search	Searches by using the Class of Service selected by the RDU for a device that, for one reason or another, cannot retain its registered Class of Service.
Selected DHCP Criteria Search	Searches using the DHCP Criteria selected by the RDU for a device that, for one reason or another, cannot retain its registered DHCP Criteria.

Note Normally, the Related and Selected Class of Service and the Related and Selected DHCP Criteria are identical. If they are not, you should investigate and modify the Selected Class of Service/DHCP Criteria to match the Related Class of Service/DHCP Criteria.

A Page Size drop-down list on the Manage Devices page lets you limit the number of search results that display per page. You can select 25, 50, or 75 results for display. If the number of results returned for a search exceeds the number selected, a screen prompt appears at the lower left corner of the page. These controls let you scroll backward or forward one page at a time, or to select a specific page.

A maximum of 1,000 results are returned for any query, with a maximum of 75 results appearing per page. To change the default maximum:

1. Change the */adminui/maxReturned* property in the *BPR_HOME/rdu/conf/adminui.properties* file.
2. Restart the Prime Cable Provisioning Tomcat process for the administrator user interface:

```
# /etc/init.d/bprAgent restart tomcat
```

Searching for devices returns results under the following headings or links that appear on the page:

- Identifier—Identifies all devices matching the search criteria. Each of the identifiers that appear links to another page from which you can modify the device.
- Device Type—Displays the available device types. Available selections include:
  - CableHome MAN-Data

  - CableHome MAN-WAN
  - DOCSIS Modem
  - Computer
  - PacketCable Multimedia Terminal Adapter (MTA)
  - Set-top box (STB)
- Status—Identifies whether or not the device is provisioned. A provisioned device is one that has been registered using the application programming interface (API), or the administrator user interface, and has booted on the network.
- Details—Displays all available details for the selected device. For additional information, see Viewing Device Details, page 18-4.

Viewing Device Details

You can view the details of any device identified in the search results. To view any device details, select the device and click **Details**.

Note The information that appears in the View Device Details page largely depends on the type of device you choose. The sample fields listed in Table 18-2 identify the details that typically appear for most devices.

Table 18-2 View Device Details Page

Field or Button	Description
Device Details	
Device Type	Identifies the device type; for example, a DOCSIS modem.
MAC Address	Identifies the MAC address of the device.
DUID	Identifies the DUID of the device.
FQDN	Identifies the fully qualified domain name (FQDN) for the device; for example, IGW-1234.EXAMPLE.COM.
Host Name	Identifies the host. For example, in the FQDN description above, IGW-1234 is the hostname.
Domain Name	Identifies the domain within which the host resides. For example, in the FQDN description above, EXAMPLE.COM is the domain name.
OID	Specifies the Object Identifier, which is the value that identifies a specific SNMP Object in the MIB database.
Revision Number	Identifies the OID revision numbers that are validated before processing.
Behind Device	Identifies the device that is behind this device.
Provisioning Group	Identifies the provisioning group to which the device has been pre-assigned or assigned automatically. This is an active link that, if clicked, displays the Provisioning Group Details page.

Table 18-2 View Device Details Page (continued)

Field or Button	Description
Registered DHCP Criteria	Identifies the DHCP Criteria used. Except in the case of the default DHCP Criteria, this is an active link that, if clicked, displays the appropriate Modify DHCP Criteria page. If you select the default DHCP Criteria, the DHCP Criteria that is configured as the default on the Systems Defaults page is applied.
Security Domain	Identifies the RBAC domain assigned to an entity (such as Device, CoS, DHCP Criteria), in case Instance Level Access control is enabled.
Device Properties	Identifies any properties, other than those that appear on this page, that can be set for this device. This field includes the display of custom properties.
Device Provisioned State	Specifies if the device is provisioned. A device is provisioned only when it is registered and has booted on the network.
Device Registered State	Identifies if the device is registered.
Client Identifier	Identifies the client identification used by the device in its DHCP messages.
Client Request Host Name	Identifies the hostname that the client requests in its DHCP messages.
Registered Class of Service	Identifies the Class of Service assigned to the device. This is an active link that, if clicked, displays the appropriate Modify Class of Service page. If a different Class of Service has been selected for the device by extension, an additional field with Selected Class of Service appears.
Owner Identifier	Identifies the device. This may be a user ID or an account number; the field may also be blank.
Detected Properties	Identifies properties returned by the RDU device-detection extensions when configuration for the device is generated.
Selected Properties	Identifies properties returned by the RDU service-level selection extensions for the detected device type when the configuration for the device is generated.
Is Behind Required Device	Specifies "false" if the *DeviceDetailsKeys.IS_BEHIND_REQUIRED_DEVICE* property has been used to establish a required relay agent device and the service-level selection extension determines that this device did not boot behind the required relay agent.
Is In Required Provisioning Group	Specifies "false" if the *IPDeviceKeys.MUST_BE_IN_PROV_GROUP* property has been used to establish a required provisioning group and the service-level selection extensions determine that this device did not boot in the required provisioning group.

Table 18-2 View Device Details Page (continued)

Field or Button	Description
Selected Access	Identifies the access granted to the device by the service-level selection extensions: • REGISTERED—Indicates that the device was registered and met requirements for access. • PROMISCUOUS—Indicates that the device's provisioning will be based on policies assigned to its relay agent. • DEFAULT—Indicates that the device will be provisioned with default access for its device type. • OTHER—Not used by the default extensions built into Prime Cable Provisioning and is provided for use by custom extensions.
Selected Class of Service	Identifies the name of the Class of Service used to generate the configuration for the device. This is an active link that, if clicked, displays the appropriate Modify Class of Service page.
Selected DHCP Criteria	Identifies the name of the DHCP Criteria used to generate the configuration for the device. This is an active link that, if clicked, displays the appropriate Modify DHCP Criteria page.
Selected Explanation	Provides a textual description of why the service-level selection extensions selected the access they granted the device. For example, the device may have been granted default access because it did not boot in its required provisioning group.
Selected Reason	Identifies why the service-level selection extensions selected the access they granted the device as an enumeration code. The possible values are: • NOT_BEHIND_REQUIRED_DEVICE • NOT_IN_REQUIRED_PROV_GROUP • NOT_REGISTERED • OTHER • PROMISCUOUS_ACCESS_ENABLED • REGISTERED • RELAY_NOT_IN_REQUIRED_PROV_GROUP • RELAY_NOT_REGISTERED Most of these indicate violations of requirements for granting registered or promiscuous access, resulting in default access being granted.
Related Group Name (Group Type)	Identifies the groups to which this device is related. This is an active link that, if clicked, displays the appropriate Modify Group page. See Managing Groups, page 12-2.
DHCPv4 Information **Note** This section does not appear unless the device has discovered DHCPv4 data.	
DHCP Inform Dictionary	Identifies additional information that the Cisco Prime Network Registrar extensions send to the RDU when requesting the generation of a configuration. This is for internal Prime Cable Provisioning use only.

Table 18-2 View Device Details Page (continued)

Field or Button	Description
DHCP Request Dictionary	Identifies the DHCP Discover or DHCP Request packet details sent from the Network Registrar extensions to the RDU when requesting the generation of a configuration.
DHCP Response Dictionary	This field is for internal Prime Cable Provisioning use only; it should always be empty.
DHCP Environment Dictionary	This field is for internal Prime Cable Provisioning use only; it should always be empty.
Lease v4 Information **Note** This section does not appear unless the device has discovered Lease v4 data.	
IP Address	Identifies the IPv4 address of the device.
DHCP Lease Properties	Identifies the lease properties, along with an IPv4 update, that Network Registrar sends to the RDU.

Table 18-2 View Device Details Page (continued)

Field or Button	Description
DHCPv6 Information	
Note This section does not appear unless the device has discovered DHCPv6 data.	
DHCPv6 Inform Dictionary	Identifies additional information that the Cisco Prime Network Registrar extensions send to the RDU when requesting the generation of a configuration. This is for internal Prime Cable Provisioning use only.
DHCPv6 Request Dictionary	Identifies the DHCP Discover or DHCP Request packet details sent from the Network Registrar extensions to the RDU when requesting the generation of a configuration.
DHCPv6 Relay Request Dictionary	Identifies DHCP packet details sent from the Network Registrar extensions to the RDU when requesting the generation of a configuration. This data, however, is derived from the CMTS, and includes information on the CMTS, and the DOCSIS version that the CMTS uses.
DHCPv6 Response Dictionary	This field is for internal Prime Cable Provisioning use only; it should always be empty.
DHCPv6 Environment Dictionary	This field is for internal Prime Cable Provisioning use only; it should always be empty. But if you set a value for the Attributes from Environment Dictionary on the Network Registrar default (**Configuration > Defaults > NR Defaults**) page, that value appears here.
Lease v6 Information	
Note This section does not appear unless the device has discovered Lease v6 data.	
IP Address	Identifies the IPv6 address of the device.
DHCPv6 Lease Properties	Identifies the lease properties, along with an IPv6 update, that Network Registrar sends to the RDU.
Technology-Specific Information	
Note The technology-specific information identifies only data that is relevant for the technologies you are licensed to use.	
XGCP Ports	Identifies the ports on which the Gateway Control Protocol is active.
DOCSIS Version	Identifies the DOCSIS version currently in use.

Managing Devices

The Devices menu lets you add devices to the RDU database and update preprovisioned data. Device management includes:

- Adding, deleting, and modifying RDU devices records
- Regenerating configurations
- Relating devices to management objects, such as Provisioning Group, Class of Service, and Group.

This section describes how to perform various device management functions on new or existing devices. Several information fields appear consistently in all device management pages. These fields include:

- Device Type—When adding a device, this is a drop-down list that identifies the available device types you can create within Prime Cable Provisioning. Available selections, as they appear on screen, include:

  - CableHomeWanData
  - CableHomeWanMan
  - Computer
  - DOCSISModem
  - PacketCableMTA
  - STB

  When modifying a device, the device type cannot be edited or changed.

- MAC Address—Identifies the MAC address of the device.

  Enter the MAC address of the device being added in this field. When doing this, ensure that you enter the commas (,) and colons (:) appropriately. For example, 1,6,00:00:00:00:00:AE.

- DUID—Identifies the DUID of the device.

  Enter the DUID of the device being added in this field. When doing this, ensure that you enter the colons (:) appropriately. For example, 00:03:00:01:02:03:04:05:06:a0.

- Host Name**—**Identifies the device host. For example, from an FQDN of node.example.com, node is the hostname.
- Domain Name**—**Identifies the domain within which the host resides. For example, from an FQDN of node.example.com, example.com is the domain name.
- Owner Identifier**—**Identifies the device by using something other than the hostname. This may be a user ID, or an account number; for example, 1000000000000000000. You can also leave this field blank.
- Registered Class of Service**—**Specifies the Class of Service that the device is provisioned with; for example, the default option or a Class of Service that you defined.
- Registered DHCP Criteria**—**Specifies the DHCP Criteria that the device is provisioned with; for example, the default option or a DHCP Criteria that you defined.

Adding Device Records

To add a device record:

Step 1 From the Manage Devices page, click **Add.**

Step 2 Choose the device type from the options available in the drop-down list.

Step 3 Enter details for the other fields on the page, such as MAC address, DUID, and hostname.

Step 4 Choose the Class of Service, and the DHCP Criteria registered for the device.

Step 5 In addition to the values that you provided for the device earlier, you can optionally add new values for existing property name/value pairs.

- Property Name—Identifies the name of the custom or built-in device property.
- Property Value—Identifies the value of the property.

Step 6 Click **Submit**.

Deleting Devices

Deleting device records is a simple process, but one that you should use carefully. To undo the delete, you must restore a previously backed-up database or re-add the device. If restoration of a backed-up database becomes necessary, see Database Restore, page 5-6.

To delete a device record, locate the device that you want to delete and click **Delete**.

Regenerating Device Configurations

The **Regenerate** button or API operation forces immediate regeneration of configurations for a device that are sent to the DPEs in the device's provisioning group.

Normally, the process of regenerating the configuration is automatically triggered following changes to the device, Class of Service, or other such impacting changes. However, after a change to a Class of Service, the system takes time to regenerate configurations for all devices. You can use the Regenerate button to expedite regeneration of configurations for a given device; this option is especially useful during proactive troubleshooting.

It is sometimes necessary to change many Class of Service or DHCP Criteria parameters. When this happens, existing device configurations become stale and require regeneration of the configuration. To eliminate the need to manually regenerate each configuration, and reduce the potential for introducing errors, Prime Cable Provisioning provides a configuration regeneration service (CRS) that you can use to automatically regenerate all device configurations.

Device configurations are automatically regenerated whenever:

- A file related to a Class of Service, that is, a template or script, is updated.
- The default Class of Service or DHCP Criteria for a device type is changed.
- A DHCP Criteria property is changed.
- The provisioning group object is changed via the administrator user interface or the API.
- The Class of Service object properties are changed.
- The DPE sends a configuration regeneration request to the RDU.
- The device properties or relationship are updated.
- Extended Dynamic TFTP Filename scripts associated to devices are replaced.

Some configurations cannot be automatically regenerated because Prime Cable Provisioning cannot determine if the change impacts device configuration. In such cases, manually regenerate configurations using the *generationConfiguration()* method or from the administrator user interface. Configurations that you must manually regenerate are those that become necessary when:

- A technology default is changed, except for the default Class of Service and the default DHCP Criteria. Changing the technology default properties for the default Class of Service and DHCP Criteria does trigger regeneration of the devices that are given the default DHCP Criteria or default Class of Service.
- The system defaults are changed.
- A file that is included within another DOCSIS template is changed.

Note Regardless of how configurations are regenerated, they are not propagated to the devices until the device configuration is activated, that is, the device contacts the DPE either on schedule or as a result of a connection request initiated from the DPE.

To regenerate a configuration for a device:

Step 1 From the Manage Devices page, locate the device for which you want to regenerate a configuration. You can use one of the search types for this purpose.

Step 2 Check the check box to the left of the device.

Step 3 Click **Regenerate**.

The RDU regenerates a configuration for the specific device.

Relating and Unrelating Devices

The concept of relating devices is similar to that of Class of Service or DHCP Criteria inasmuch as a device is related to a specific Class of Service or to a specific DHCP Criteria. The significant difference is that the Class of Service and DHCP Criteria are considered to be predefined groups and that you use groups to group devices into arbitrary groups that you define.

In this context, the Relate function lets you associate a device, using its MAC address or DUID, to a specific group, which is in turn associated with a specific group type.

By relating a device to a specific group, information indicating that the device is related to a specific group is stored in the database. If you relate the device to the predefined **system-diagnostics (system)** group, you can use available information to troubleshoot potential problems.

Relating a Device to a Group

You can relate and unrelate only one device at a time from the administrator user interface.

To relate a device:

Step 1 From the Manage Devices page, locate the device that you want to relate to a group. You can use one of the search types for this purpose.

Step 2 Check the check box to the left of the device.

Step 3 Click **Relate**. The Relate Device to Group page appears.

Step 4 Select the group type from the drop-down list and the group from the list of defined groups.

Note To select multiple groups from the Groups list, press **Control** or **Shift**.

Step 5 Click **Submit**.

To verify if the device is related to the group you specified, click the View Details icon corresponding to the device. On the Device Details page that appears, check the status against Related Group Name (Group Type).

Unrelating a Device from a Group

To unrelate a device from a group:

Step 1 From the Manage Devices page, locate the device that you want to unrelate from a group.

Step 2 Check the check box corresponding to the device identifier, and click the **Unrelate** button.

Step 3 From the list of defined groups, select the group from which you want to unrelate the device.

Note To select multiple groups from the Groups list, press **Control** or **Shift**.

Step 4 Click **Submit**. The Manage Devices page appears.

Searching Devices in a Group

To search for devices belonging to a particular group:

Step 1 From the Manage Devices page, select the Group Search option from the drop-down list under Search Type.

Step 2 From the Group Name (Group Type) drop-down list, select the name of the group to which the devices are associated.

Step 3 Click **Search**.

The devices related to the group appear.

Resetting Devices

The Reset button lets you reboot any selected device.

To reset a device:

Step 1 From the Manage Devices page, locate the device that you want to reboot. You can use one of the search types for this purpose.

Step 2 Check the check box corresponding to the device.

Step 3 Click **Reset**.

The device reboots.

Managing Dynamic File Configuration

This chapter describes the following features that Cisco Prime Cable Provisioning supports for device configuration and device management:

- Groovy Scripting, page 19-1
- Templates, page 19-15

Groovy Scripting

This section explains the Groovy scripting support that Prime Cable Provisioning provides for device configuration and device management. This section features:

- Overview, page 19-1
- Groovy Script Language, page 19-2
- Adding a Groovy Script to Cisco Prime Cable Provisioning RDU, page 19-3
- Using Configuration File Utility for Groovy, page 19-3
- TFTP File-Naming Convention, page 19-12

Overview

Prime Cable Provisioning uses Groovy scripting, apart from templates, for generating the configuration file, which helps you to deploy dynamic files for any CableLabs standard supported by Prime Cable Provisioning including DOCSIS, PacketCable, CableHome, and OpenCable STB. This scripting interface allows you to access the discovered DHCP data and device properties, which will help in deciding the TFTP file that has to be generated. The Prime Cable Provisioning RDU generates the configuration file using either the template or Groovy scripting. The RDU identifies the Groovy file by the extension, *.groovy*. Groovy sample script files are available in the BPR_HOME/rdu/samples/groovy directory, which can be used for testing.

To create your Groovy script file, you should be familiar with the Groovy scripting language, in addition to the requirements for templates creation.

Groovy Script Language

A Groovy script can include the following options:

Table 19-1 Groovy Script

Option	Description	Example
<comment>	// [ascii-string] /* * Multi-line comments */	// Config File Start/End / * configFile—of type DOCSISTFTPFile * services—of type ExtensionServices * discoveredData—of type DHCPDataAccess * deviceProperties—of type CSRCProperties * option—of type DOCSISoptionfactory * device—of type IPDevice * context—of type ConfigContext */
<option-description>	<option-with-no-suboptions> | <compound-option>	
<option-with-no-suboptions>	configFile.add(option.createOptionValue(<custom-value>,"<option-num>","<option-value>"));	configFile.add(option.createOptionValue("3", "1")); For custom values: configFile.add(option.createOptionValue(OptionSyntax.HEX,"217.53","010868446146484A4737")) ;

<compound-option>	def <variable-name> = option .createOptionValue("<option-num>");
	<variable-name>.add(option .createOptionValue(<custom-value>,"<option-num>","<option-value>"));
	configFile.add(<variable-name>);
		def option24 = option.createOptionValue("24");
		option24.add(option.createOptionValue("24.8", "4194304"));
		configFile.add(option24);

<custom-value> optional	OptionSyntax.HEX | OptionSyntax.ASCII | OptionSyntax.SNMP	
<option-num>	<unsigned-byte>[.<unsigned-byte>]*	"24"
<option-value>	<option-value-string>[,<option-value-string>] *	"1" or [".docsDevNmAccessCommunity.1", "Octet String", "private"] as String[]

Bindings that are visible to the Groovy environment are:

- configFile—of type DOCSISTFTPFile
- services—of type ExtensionServices
- discoveredData—of type DHCPDataAccess
- deviceProperties—of type CSRCProperties
- option—of type DOCSISoptionfactory
- device—of type IPDevice

- context—of type ConfigContext

Note Device object binding is not available while executing the Groovy script from CLI File Utility.

Adding a Groovy Script to Cisco Prime Cable Provisioning RDU

To add a Groovy script file to a Prime Cable Provisioning RDU:

Step 1 Choose **Configuration > Files**. The View Files page appears.

Step 2 Click Add. The Add Files page appears.

Step 3 Choose the CableLabs Configuration Script option from the File Type drop-down list.

Step 4 Browse for the Source File Name

Step 5 Add the <*filename*>.*groovy* file in the File Name field.

Step 6 Click **Submit**.

Using Configuration File Utility for Groovy

Configuration file utility is used to convert groovy file to a binary configuration file and vice versa. It can also be used to view and validate the configuration and groovy files. The configuration file utility is installed in the BPR_HOME/rdu/bin directory. The groovy file and the binary file must be available in the directory from where the configuration file utility is invoked.

Note Since Prime Cable Provisioning uses the configuration utility only to generate binary to groovy file and vice versa, it will not support other scripting languages.

This section discusses the following topics:

- Running the Configuration File Utility, page 19-4
- Validating a Groovy Script Using runCfgUtil, page 19-5
- Converting a Binary File to a Groovy Script File, page 19-6
- Testing Groovy Script Processing for a Local Groovy Script File, page 19-6
- Testing Groovy Script Processing for an External Groovy Script File, page 19-7
- Testing Groovy Script Processing for a Local Groovy Script File and Adding Shared Secret, page 19-7
- Testing Groovy Script Processing for a Local Groovy Script File and Adding EMIC Shared Secret, page 19-8
- Specifying Dynamic Variables at the Command Line, page 19-9
- Specifying a Device for Dynamic Variables, page 19-9
- Specifying Discovered Data at the Command Line, page 19-10
- Specifying a Device for Discovered Data, page 19-11

Running the Configuration File Utility

To run the configuration file utility, run the command from the *BPR_HOME/rdu/bin* directory:

runCfgUtil.sh *options*

The available options include:

- **-?**—Prints this usage message.
- **-e**—Performs encoding of a BACC groovy/template file (default).
- **-d**—Performs decoding of a binary file.
- **-g**—Performs generation of a groovy/template file from a binary file.
- **-c** *shared*—The CMTS shared secret to use when parsing a BACC groovy/template file (the default is cisco).
- **-h** *host:port*—Specifies where the RDU is located (the default is localhost:49187).
- **-i** *device ID*—Specifies the device to use for macro variables substitutions when parsing a groovy/template.
- **-m** *macros*—Specifies the macro variables to be substituted when parsing a groovy/template.
- **-s**—Displays the parsed groovy/template or the contents of the binary file in a human readable format.
- **-o** *filename*—Saves the parsed groovy/template or the human readable output in the specified filename.
- **-l** *filename*—Specifies the input file to be on the local file system.
- **-r** *filename*—Specifies the input file to be remote on the RDU.
- **-pkt**—Specifies the file to be processed as a PacketCable MTA configuration file.
- **-t** *type*—Specifies the PacketCable encoding type (default is secure).
- **-loc** *locale*—Specifies the PacketCable locale such as na, euro, and, ietf (default is na). The default is na. If the MTA is euro-MTA, then the locale should be set to euro.
- **-cablehome**—Specifies the file to be processed as a CableHome configuration file.
- **-docsis**—Specifies the file to be processed as a DOCSIS configuration file (default).
- **-E**—Enable Extended CMTS MIC (EMIC) calculation and identifies the default options for EMIC calculation. The default options are:
  - HMAC type—MMH16
  - EMIC Digest type—Explicit
  - EMIC shared secret as cisco.
- **-Ei**—Specifies [implicit] presentation that will be used for Extended CMTS MIC Digest Subtype.

- **-Eh** *HMACType*—Specifies the hashing algorithm used to compute Extended CMTS MIC. The supported algorithms are MD5 and MMH16 (default is MMH16).
- **-Es** *secret*—The CMTS shared secret to use for Extended CMTS MIC calculation (the default is cisco).
- **-u** *username*—Specifies the username to use when connecting to the RDU.
- **-p** *password*—Specifies the password to use when connecting to the RDU.
- **-v** *version*—Specifies the version of the technology to process the input file.
- **-prop** *filename*—Specifies the property file that has the key and value for the variables used in dynamic script.
- **-dis** *filename*—Specifies the discovered data to be used in the dynamic script in the form key and value pair.
- **-DDv4** *filename*—Specifies the discovered DHCPv4 data to be used in dynamic script in the form key and value pair.
- **-DDv6** *filename*—Specifies the discovered DHCPv6 data to be used in dynamic script in the form key and value pair.
- **-cp** *classpath*—Specifies the path of the extension jars and script files referred in dynamic script.
- **-b**—Specifies bulk processing option for generating multiple output files. All the binary files in the given directory (using -l option) will be processed and the generated files will be available in the output directory indicated in -o option.
- **-ft**—The file type (groovy or tmpl)to be generated. This option will be used when bulk processing is enabled (using -b option) for generation(-g option) operation. (The default file type is tmpl.)

Validating a Groovy Script Using runCfgUtil

To use the configuration file utility to test Prime Cable Provisioning Groovy script:

Step 1 Develop the Groovy script. If the Groovy script extends to other Groovy scripts, make sure all the referenced Groovy scripts are in the same directory.

Step 2 Run the configuration file utility on the local file system. You can check the syntax for the Groovy script, or have the configuration file utility process the Groovy script as CRS would, and return output.

If the Groovy script contains dynamic variables or the discovered data, perform these operations in the order specified:

a. Test with command line substitution with property file.

b. Test with a device that has been added to your RDU.

Step 3 Add the Groovy script (and any extended Groovy scripts that are used) to the RDU.

Step 4 Run the configuration file utility to parse a file. See Testing Groovy Script Processing for an External Groovy Script File.

If the Groovy script contains dynamic variables or the discovered data, perform these operations in the order specified:

a. Test with command line substitution with property file.

b. Test with a device that has been added to your RDU.

Step 5 After all tests succeed, configure a Class of Service to use the Groovy script.

Converting a Binary File to a Groovy Script File

Use the runCfgUtil.sh command to convert binary configuration memory files into Groovy script files. Prime Cable Provisioning dynamic configuration generation is based on Groovy scripts that are created. Automatically converting existing, tested, binary files to Groovy script files speeds the process and reduces the possibility of introducing errors.

Note Using the runCfgUtil.sh tool, you cannot convert a template directly into Groovy scripts and vice versa. You must first convert the template into a binary file, and then convert the binary file into a Groovy script. When you convert a Groovy script to template, you must first convert the Groovy script into a binary file, and then convert the binary file into a template.

Syntax Description

runCfgUtil.sh -g -l *binary_file* **-o** *groovy_file*

- **-g**—Specifies that a Groovy script file needs to be generated from an input binary file
- **-l** *binary_file*—Specifies the local input file, including the pathname. In all cases, the input binary filename will have a *.cm* file extension; *bronze.cm* for example.
- **-o** *groovy_file*—Specifies the output Groovy script file, including the pathname. In all cases, the output Groovy script file will have a *.groovy* file extension; for example, *test.groovy*.

To convert a binary file into a Groovy script file:

Step 1 Change directory to */opt/CSCObac/rdu/samples/docsis*.

Step 2 Select a Groovy script file to use. This example uses an existing binary file called *unprov.cm*.

Step 3 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -g -l unprov.cm -o test.groovy -docsis
```

-docsis—Specifies that the input file is a DOCSIS configuration file.

Testing Groovy Script Processing for a Local Groovy Script File

Use the **runCfgUtil.sh** command to test the processing for Groovy script files stored on the local file system.

Syntax Description

runCfgUtil.sh -pkt -l *file*

- **-pkt**—Identifies the input file as a PacketCable MTA file.
- **-l**—Specifies that the input file is on the local file system.
- *file*—Identifies the input Groovy script file being parsed.

To parse a Groovy script file that is on the local file system:

Step 1 Change directory to */opt/CSCObac/rdu/samples/packet_cable*.

Step 2 Select a Groovy script file to use. This example uses an existing Groovy script file called *unprov_packet_cable.groovy*. The **-pkt** option is used because this is a PacketCable MTA Groovy script.

Step 3 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -pkt -l unprov_packet_cable.groovy
```

unprov_packet_cable.groovy—Identifies the input Groovy script file being parsed.

Testing Groovy Script Processing for an External Groovy Script File

Use the **runCfgUtil.sh** command to test processing of external Groovy script files.

Syntax Description

runCfgUtil.sh -docsis -r *file* **-u** *username* **-p** *password*

- **-r**—Identifies the input file as a file that has been added to the RDU.
- *file*—Identifies the input Groovy script file being parsed.
- **-u** *username*—Specifies the username to use when connecting to the RDU.
- **-p** *password*— Specifies the password to use when connecting to the RDU.
- **-docsis**—Identifies the file as a DOCSIS Groovy script.

To parse a Groovy script file that has been added to the RDU:

Step 1 Select a Groovy script file to use. This example uses an existing Groovy script file called *unprov.groovy*. The **-docsis** option is used because a DOCSIS Groovy script is being used.

Step 2 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -docsis -r unprov.groovy -u admin -p changeme
```

- **unprov.groovy**—Identifies the input file.
- **admin**—Identifies the default username.
- **changeme**—Identifies the default password.

Testing Groovy Script Processing for a Local Groovy Script File and Adding Shared Secret

Use the **runCfgUtil.sh** command to test the processing for a Groovy script file and add a shared secret that you specify.

Syntax Description

runCfgUtil.sh -e -docsis -l *file* **-c** *shared*

- **-e**—Identifies the encode option.
- **-docsis**—Identifies the input file as a DOCSIS Groovy script file.

- **-l**—Specifies that the input file is on the local file system.
- *file*—Identifies the input Groovy script file being parsed.
- **-c**—Specifies the CMTS shared secret when parsing a DOCSIS Groovy script file.
- *shared—*Identifies the shared secret. The default shared secret is **cisco**.

To parse a locally saved Groovy script file, and set a user-specified shared secret:

Step 1 Change directory to */opt/CSCObac/rdu/groovy*.

Step 2 Select a Groovy script file to parse. This example uses an existing Groovy script file called *unprov.groovy*. The **-docsis** option is used because this is a DOCSIS Groovy script.

Step 3 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -e -docsis -l unprov.groovy -c shared
```

- **unprov.groovy**—Identifies the input file on the local file system.
- **shared**—Identifies that shared secret.

Testing Groovy Script Processing for a Local Groovy Script File and Adding EMIC Shared Secret

Use the **runCfgUtil.sh** command to test the processing for a Groovy script file and add a Extended CMTS MIC (EMIC) shared secret that you specify.

Syntax Description

runCfgUtil.sh -E -docsis -l *filename*

- **-E**—Enables EMIC calculation.
- **-docsis**—Identifies the input file as a DOCSIS Groovy script file.
- **-l** *filename*—Specifies the input Groovy script file, including the pathname. In all cases, the input Groovy script file will have a *.groovy* file extension; for example, test.groovy.

To calculate the EMIC with default settings:

Step 1 Select a Groovy script file to use. This example uses an existing Groovy script file called *unprov.groovy*. The **-docsis** option is used because a DOCSIS Groovy script is being used.

Step 2 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -E -l test.groovy
```

Specifying Dynamic Variables at the Command Line

Use the **runCfgUtil.sh** command to specify dynamic variables.

Syntax Description

runCfgUtil.sh -e -l *file* **-prop** *"file"*

- **-e**—Identifies the encode option.
- **-l**—Specifies the input file is on the local file system.
- *file*—Identifies the input Groovy script file being parsed.
- **-prop**—Specifies the property file that has key and value for variables used in dynamic script.
- *"file"*—Identifies the desired dynamic variable. If multiple dynamic variables are required, then each key value pair should be given one after the other.

To specify values for dynamic variables at the command line:

Step 1 Change directory to */opt/CSCObac/rdu/groovy*.

Step 2 Select a Groovy script file to use.

Step 3 Identify the dynamic variables in the Groovy script.

Step 4 Identify the values for the variables.

Step 5 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -e -l macro.groovy -prop prop.properties
```

- **macro.groovy**—Identifies the input file.
- **prop.properties**—Contains key value and pair (eg: MTA_PROP=3)

Specifying a Device for Dynamic Variables

Use the **runCfgUtil.sh** command to specify a device for dynamic variables.

Syntax Description

runCfgUtil.sh -e -r *file* **-i** *MAC* **-u** *username* **-p** *password*

- **-e**—Identifies the encode option. Accepts key and if not mentioned, it takes the default key.
- **-r**—Identifies the input file as a file that has been added to the RDU.
- *file*—Identifies the input Groovy script file being parsed.
- **-i**—Specifies the device to use when parsing dynamic variables.
- *MAC*—Identifies the MAC address of the device.
- **-u** *username*—Specifies the username to use when connecting to the RDU.
- **-p** *password*— Specifies the password to use when connecting to the RDU.

To specify a device to be used for dynamic variable substitution:

Step 1 Select a Groovy script file to use. This example uses the existing Groovy script file, *macro.groovy.*

Step 2 Identify the dynamic variables in the Groovy script.

Step 3 Identify the device to use. This example assumes that the device exists in the RDU and has the dynamic variables set as properties.

Step 4 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -e -r macro.groovy -i "1,6,00:01:02:03:04:05" -u
admin -p changeme
```

- **macro.groovy**—Identifies the input file.
- **1,6,00:01:02:03:04:05**—Identifies the MAC address of the device. The MAC address used here is an example only.
- **admin**—Identifies the default username.
- **changeme**—Identifies the default password.

Specifying Discovered Data at the Command Line

Use the **runCfgUtil.sh** command to specify Discovered Data.

Syntax Description

runCfgUtil.sh -e -l *file* **-dis** *"file"*

- **-e**—Identifies the encode option. Accepts key and if not mentioned, it takes the default key.
- **-l**—Specifies the input file is on the local file system.
- *file*—Identifies the input Groovy script file being parsed.
- **-dis**—Specifies the discovered data to be used in dynamic script in the form key and value pair.
- *"file"*—Identifies the desired discovered data.

To specify values for discovered data at the command line:

Step 1 Select a Groovy script file to use.

Step 2 Identify the discovered data in the Groovy script.

Step 3 Identify the values for the discovered data.

Step 4 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -e -l macro.groovy -dis dis.properties
```

- **macro.groovy**—Identifies the input file.
- **dis.properties**—contains key value and pair (eg: giaddr=10.1.1.9).

Specifying a Device for Discovered Data

Use the **runCfgUtil.sh** command to specify a device and use its discovered data for configuration file generation.

Syntax Description

runCfgUtil.sh -e -r *file* **-i** *MAC* **-u** *username* **-p** *password*

- **-e**—Identifies the encode option. Accepts key and if not mentioned, it takes the default key.
- **-r**—Identifies the input file as a file that has been added to the RDU.
- *file*—Identifies the input Groovy script file being parsed.
- **-i**—Specifies the device to use when parsing discovered data.
- *MAC*—Identifies the MAC address of the device.
- **-u** *username*—Specifies the username to use when connecting to the RDU.
- **-p** *password*— Specifies the password to use when connecting to the RDU.

To specify a device to be used for discovered data substitution:

Step 1 Select a Groovy script file to use. This example uses the existing Groovy script file, *macro.groovy.*

Step 2 Identify the discovered data in the Groovy script.

Step 3 Identify the device to use. This example assumes that the device exists in the RDU and has the discovered data set as properties.

Step 4 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -e -r macro.groovy -i "1,6,00:01:02:03:04:05" -u
admin -p changeme
```

- **macro.groovy**—Identifies the input file.
- **1,6,00:01:02:03:04:05**—Identifies the MAC address of the device. The MAC address used here is an example only.
- **admin**—Identifies the default username.
- **changeme**—Identifies the default password.

Generate Binary File from Groovy

Use the **runCfgUtil.sh** command to specify the output of a parsed Groovy script as a binary file.

Syntax Description

runCfgUtil.sh -l *input_file* **-o** *output_file*

- **-l**—Specifies that the input file is on the local file system.
- *input_file*—Identifies the input Groovy script file being parsed.
- **-o**—Specifies that the parsed Groovy script file is to be saved as a binary file.
- *output_file*—Identifies the name of the file in which the binary contents of the parsed Groovy script file are stored.

To specify the output from parsing a Groovy script to a binary file:

Step 1 Select a Groovy script file to use.

Step 2 Identify the name of the output file. This example uses *unprov.cm*.

Step 3 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -l unprov.groovy -o unprov.cm
```

- **unprov.groovy**—Identifies the existing Groovy script file being parsed into a binary file.
- **unprov.cm**—Identifies the output filename to be used.

Viewing a Local Binary File

See Viewing a Local Binary File, page 19-48 for details.

Viewing an External Binary File

See Viewing an External Binary File, page 19-49 for details.

Activating PacketCable Basic Flow

See Activating PacketCable Basic Flow, page 19-50 for details.

Generating TLV 43s for Multivendor Support

See Generating TLV 43s for Multivendor Support, page 19-52 for details.

TFTP File-Naming Convention

The TFTP File-Naming Convention helps you customize the variable components of the dynamic TFTP filenames, and their order. The Groovy script generates the TFTP filename by using components like the DHCP discovered data as well as the other interfaces that are being exposed to it. The script can include any important information such as, the class of service name, discovered vendor name, downstream speed and so on. You can configure the script either in technology defaults or in system defaults. The default maximum filename length is 127 characters.

If a CableLabs configuration filename script is modified, configuration regeneration is triggered for the list of affected devices to reflect the changes.

You can find Groovy script samples in *BAC_HOME/rdu/samples/groovy*.

Basic Flow of TFTP File-Naming Convention

The following is the sequence of steps that explain the basic flow of TFTP file-naming:

1. A device boots and a request for generating the config file is sent to the RDU.
2. The RDU runs the configuration generation for the device, during which, the generation extension determines that a dynamic TFTP file needs to be assigned to the device. The file could be a template (for example, docsis.tmpl) or a script (for example, docsis.groovy), but cannot be a static file (for example, silver.cm).
3. The generation extension looks for the CableLabs Configuration Filename Script property in the technology defaults. If not found, it looks in the system defaults. The value of the property (CableLabs Configuration Filename Script:) is the name of the script to be executed.
4. The script is executed and it returns the additional strings to be included in the dynamic TFTP file name.
5. The generation extension takes this value and creates a dynamic TFTP filename for the device. After the filename generation is complete, the configuration is sent to the DPEs, which is then cached.

Example 19-1 Sample TFTP Filename Groovy Script

```
/**
 * example_extended_filename.groovy
 *
 * A sample CableLabs Configuration Filename Script that demonstrates how
 * to create an extended filename.  This example includes the following
 * strings in the extended filename: DeviceType, Selected ClassOfService,
 * and provisioning group.  For DOCSIS device types, the default DOCSIS
 * version is included after device type. The resulting extended filename
 * string is:
 *
 * "<device-type>_<default-docsis-version>_<selected-cos>_<pg>"
 * (e.g., "cm_11_goldcos_westpg", "pc_silvercos_eastpg").
 *
 * A CableLabs Configuration Filename Script specifies an extended filename
 * label that is appended to the standard BAC dynamic configuration filename.
 * In BAC 4.2 and later releases, the dynamic configurations have a filename
 * consisting of the fixed/standard prefix.  The script can be configured at
 * System Defaults (preferred) and/or Technology Defaults.
 *
 * BAC properties:
 *     DocsisDefaultKeys.DOCSIS_DEFAULT_VERSION
 *
 * Variable bindings:
 *     configFileName - Extended Filename of type StringBuilder
 *     services - of type ExtensionServices
 *     discoveredData - of type DHCPDataAccess
 *     deviceProperties - of type CSRCProperties
 *     device - of type IPDevice
 *     context - of type ConfigContext
 */

import com.cisco.provisioning.cpe.constants.DocsisDefaultKeys
import com.cisco.provisioning.cpe.extensions.constants.CNRNames
import com.cisco.provisioning.cpe.extensions.services.DeviceType

/*
 * A Groovy list is used to collect the ordered list of string fields
 * that will comprise the extended filename.  Once all the fields have been
 * added to the list, the "join" method is used to concatenate the
 * fields with a underscore ('_') separator character.
 */
def label = []

/*
 * Add Device Type (abbreviated).
 *
 * The device type string is too verbose for a filename component, so an
 * abbreviation is used instead.  For example, the DOCSIS device type value
 * "DOCSISModem" is abbreviated as "cm".  If no abbreviation is defined,
 * a default abbreviation of "xx" is used.
 */
def deviceType = device.getDeviceType().getName()
def deviceTypeMap = [
    (DeviceType.DOCSIS_MODEM) : "cm",
    (DeviceType.PACKET_CABLE_MTA) : "pc",
    (DeviceType.CABLEHOME_WAN_MAN) : "ch",
    (DeviceType.STB) : "st",
    (DeviceType.CUSTOM_CPE): "cu"
    ]
label << deviceTypeMap[deviceType] ?: "xx"

/*
 * Add default DOCSIS version number (exclude embedded "dot").
 *
 * For DOCSIS device types, the default DOCSIS version number specifies the
 * maximum DOCSIS version supported by the CM and CMTS.  This version number
 * indicates the DOCSIS version grammar used when constructing the dynamic
 * configuration file.  The embedded "dot" is stripped from the version number
 * (i.e., "3.0" --> "30").
 */
if (deviceType == DeviceType.DOCSIS_MODEM)
{
    label << deviceProperties.getProperty(
        DocsisDefaultKeys.DOCSIS_DEFAULT_VERSION, "1.0") - "."
}

/*
 * Add Selected Class of Service name.
 */
label << device.getSelectedClassOfService().getClassOfServiceName()

/*
 * Add Provisioning Group name.
 */
label << device.getProvGroup().getProvGroupId()

/*
 * Convert the list of filename components into a string value with underscore
 * ('_') characters separating the filename components.  Add the resulting
 * string to the configFileName StringBuilder binding.
 */
configFileName << label.join("_")
```

Templates

This section details the templates that Prime Cable Provisioning supports for device configuration and device management. This section features:

- Template Files–An Overview, page 19-15
- Template Grammar, page 19-15
  - SNMP VarBind, page 19-20
  - Macro Variables, page 19-22
  - SNMP TLVs, page 19-23
  - Encoding Types for Defined Options, page 19-27
- Using Configuration File Utility for Template, page 19-33

Template Files–An Overview

Prime Cable Provisioning uses templates to help you deploy dynamic PacketCable, DOCSIS, and CableHome files. Using templates, you can create a template file in an easily readable format, and edit it quickly and simply. A template is an ASCII text file that represents the PacketCable, DOCSIS, or CableHome options and values used for generating a valid PacketCable, DOCSIS, or CableHome file. Prime Cable Provisioning uses the *.tmpl* extension to identify template files. You must add template files to the RDU as a file using the administrator user interface or the application programming interface (API), before any Class of Service can reference it.

When installing the Prime Cable Provisioning RDU component, several sample template files are copied to the *BPR_HOME/rdu/templates* directory.

Although all that you need to create or edit a template is a simple text editor, before attempting to create your own template file, you should thoroughly familiarize yourself with this information:

- Prime Cable Provisioning provisioning flows
- DOCSIS 1.0, 1.1, 2.0, and 3.0 RFI specifications
- DOCSIS Layer 2 Virtual Private Networks specification
- PacketCable 1.0, 1.5 and 2.0 specifications
- Multimedia Terminal Adapter (MTA) device provisioning specification
- CableHome 1.0 specification
- SNMP MIBs for cable devices (for example, DOCS-CABLE-DEVICE-MIB)

Template Grammar

A template comprises the following types of statements:

- Comments, page 19-16
- Includes, page 19-16
- Options, page 19-17
- Instance Modifier, page 19-18
- OUI Modifier, page 19-19

Comments allow you to document your templates. Includes allow you to create building block templates to be used in other templates. You use options to specify the PacketCable, DOCSIS, or CableHome type length value (TLV) in a descriptive manner. You can use instance modifiers to group compound options into specific individual TLVs. The OUI modifier allows you to include vendor-specific information. Table 19-2 describes the available template grammar options.

Table 19-2 Template Grammar

Option	Description
<comment>	::= #[ascii-string]
<include>	::= include "<filename.tmpl>"
<option-description>	::= option <option-num> [instance <instance-num>] [oui <oui>] <option-value>
<option-num>	::= <unsigned-byte>[.<unsigned-byte>]*
<option-value>	::= <well-defined-value> | <custom-value>
<well-defined-value>	::= <option-value-string>[,<option-value-string>]*
<custom-value>	::= <ascii-value> | <hex-value> | <ip-value> | <snmp-value>
<ascii-value>	::= ascii <ascii-string>
<hex-value>	::= hex <hex-string>
<ip-value>	::= ip <ip-string>
<instance-num>	::= <unsigned integer>
<template>	::= <template-statement>*
<template-statement>	::= <comment> | <include> | <option-description>
<snmp-value>	::= <snmpvar-oid>,<snmpvar-type>,<snmpvar-value>

Comments

Comments provide information only and are always located between the pound (#) symbol and the end of a line. Example 19-2 shows sample comment usage.

Example 19-2 Sample Comment Usage

```
#
# Template for gold service
#

option 3 1 # enabling network access
```

Includes

Include files let you build a hierarchy of similar, but slightly different, templates. This is very useful for defining options that are common across many service classes without having to duplicate the options in several templates.

You can use multiple include statements in a single template, although the location of the include statement in the template is significant: The contents of the include file are included wherever the include statement is found in the template. The included template must be added as a file to the RDU before it can be used. The included file must not contain any location modifiers such as ../.. because the templates are stored without path information in the RDU database. Example 19-3 and Example 19-4 illustrate both correct and incorrect usage of the include option.

Example 19-3 Correct Include Statement Usage

```
# Valid, including common options
include "common_options.tmpl"
```

Example 19-4 Incorrect Include Statement Usage

```
# Invalid, using location modifier
include "../common_options.tmpl"

# Invalid, using incorrect file suffix
include "common_options.common"

# Invalid, not using double quotes
include common_options.tmpl
```

Options

PacketCable, DOCSIS, and CableHome configuration files consist of properly encoded option ID-value pairs. Two forms of options are supported: defined and custom.

- Well-defined options require the option number and value. The value is encoded based on the encoding type of the option number.
- Custom options require the option number, explicit value encoding type, and the value.

When using compound options, for example, Option 43, you can use the instance modifier to specify the TLV groupings. See Instance Modifier, page 19-18.

When specifying one of these well-defined options in a template, it is not necessary to specify a value encoding for the value. For additional information on these defined encoding types, see Encoding Types for Defined Options, page 19-27, and DOCSIS Option Support, page A-1.

When specifying custom options (for example, Option 43), you must specify the encoding type for the option. The available encoding types are:

- ASCII— ASCII type encodes any given value as an ASCII string without a NULL terminator. If the value contains spaces, they must be enclosed in double quotation marks.
- hex—The value must be valid hexadecimal and there must be exactly 2 characters for each octet. If `01` is specified as the value, then exactly one octet is used in the encoding. If 0001 is specified as the value, then exactly two octets are used in the encoding process.
- IP address—IP address type encodes any given value as 4 octets. For example, the IP address 10.10.10.1 is encoded as 0A0A0A01.
- SNMPVarBind—An SNMP OID string, type, and value. Each of these is comma separated.

Use a comma to separate multivalued options on a given line. Each value is treated separately, so you might have to enclose one of the values in double quotation marks, but not the others. A good example of a multivalued option is Option 11 (SNMP VarBind). See SNMP VarBind, page 19-20, for additional information.

When specifying compound options, there is no need to specify the top-level option (for example Option 4 when specifying Option 4.1). Example 19-5 and Example 19-6 illustrate both correct and incorrect usage of the option statement.

Example 19-5 Correct Option Statement Usage

```
# Valid, specifying the number for well known option 3
option 3 1

# Valid, specifying the number for option 4 sub-option 1
option 4.1 1

# Valid, specifying a vendor option as hex
option 43.200 hex 00000C

# Valid, specifying a vendor option as ascii
option 43.201 ascii "enable log"

# Valid, specifying a vendor option as IP
option 43.202 ip 10.4.2.1
```

Example 19-6 Incorrect Option Statement Usage

```
# Invalid, using hex with incorrect hex separator
option 43.200 hex 00.00.0C

# Invalid, not using double quotes when needed
option 43.201 ascii enable log

# Invalid, not specifying IP address correctly
option 43.202 ip 10-10-10-1

# Invalid, specifying the description for option "Network Access Control"
option "Network Access Control" 1

# Invalid, specifying top level option
option 4
```

Instance Modifier

The instance modifier is used to group compound options into specific individual Type-Length-Values (TLVs). Example 19-7 and Example 19-8 illustrate both correct and incorrect methods of creating separate TLVs. These are required to enable the IOS DOCSIS modem to interpret the IOS commands as two separate commands.

Example 19-7 Correct IOS Command Line Entries

```
# Valid, each IOS command gets its own TLV
option 43.8 instance 1 00-00-0C
option 43.131 instance 1 ascii "login"
option 43.8 instance 2 00-00-0C
option 43.131 instance 2 ascii "password cable"
```

Example 19-8 Incorrect IOS Command Line Entries

```
# Invalid, IOS commands are grouped into one TLV
option 43.8 00-00-0C
option 43.131 ascii "login"
option 43.131 ascii "password cable"

# Invalid, using instance on non-compound options
option 3 instance 1 1
```

Note The encoding type for Option 43.8 is an organizationally unique identifier (OUI). Unlike that shown in Example 19-5, this type only accepts an 00-00-0C format.

OUI Modifier

The OUI modifier enhances multi-vendor support using Option 43 and its suboptions.

In Prime Cable Provisioning, you can use a single template to specify various TLV 43s from many vendors. Example 19-9 specifies the OUI formats as XX-XX-XX, where:

- FF-FF-FF—Identifies the vendor ID to specify encoding for the DOCSIS general extension.
- 00-00-0C—Identifies the Cisco vendor ID that specifies the Cisco-specific cable modem Option 43 and its suboptions.

Example 19-9 illustrates Prime Cable Provisioning support for L2VPN using a cable modem configuration file to classify upstream traffic for L2VPN. Using this template content, you can generate subTLVs:

- 43.5.1 and 43.5.2.2 from the DOCSIS general extension encoding, using OUI=FF-FF-FF.
- 43.1 from the Cisco-specific Option 43, using OUI=00-00-0C.

However, in order to comply with the DOCSIS specification, you must insert as the first subTLV for TLV 43 either:

- 0xFFFFFF when using the DOCSIS extension field to encode general extension information.
- 0x00000C when generating Cisco-specific subTLVs.

Example 19-9 Correct OUI Modifier Usage

```
# Upstream L2VPN Classifier Example

# This example shows how to classify upstream traffic from a specific CPE
# onto an upstream L2VPN service flow, in which other CPE attached to
# the cable modem forward to the non-L2VPN forwarder, as depicted below.

# This example also demonstrates that when using the DOCSIS extension
# field (TLV 43) to encode general extension information (GEI), you do
# not need to specify oui=FF-FF-FF. You only need to specify the OUI tag when
# general extension encoding is not used and vendor-specific encoding is used.

# Upstream L2VPN Classifier Cable Modem Config File

# (43) Per-CM L2VPN Encoding
# GEI (43.8) Vendor ID : 0xFFFFFF for GEI
option 43.8 instance 1 ff-ff-ff

# GEI (43.5) for L2VPN Encoding
# GEI (43.5.1) VPNID Subtype
option 43.5.1  instance 1 0234560003

# GEI (43.5) for L2VPN Encoding
# GEI (43.5.2) IEEE 802.1Q Format Subtype
# VLAN ID 25
option 43.5.2.2 instance 1 25

# Cisco Specific Vendor Option Encodings
# (43.8) Vendor ID : 00-00-0C (Cisco Vendor ID)
option 43.8 instance 2 00-00-0C
```

```
# Cisco Vendor Specific option (43.1)
# Static Downstream Frequency
# Frequency 402750000
option 43.1 instance 2 oui 00-00-0C 402750000

# Cisco Specific Vendor Option Encodings
# (43.8) Vendor ID : 00-00-0C (Cisco Vendor ID)
option 43.8 instance 3 00-00-0C

# Cisco Vendor Specific option (43.3)
# Update Boot Monitor Image
# image name (boot_monitor_image.bin)
option 43.3 instance 3 oui 00-00-0C boot_monitor_image.bin
```

Example 19-10 and Example 19-11 illustrate incorrect usage of the OUI modifier.

Example 19-10 Incorrect OUI Modifier Usage

```
# Invalid, OUI tag needs to be present for each 43 suboption if/when general extension
# encoding is not used and vendor-specific encoding is used.

option 43.8 00-00-0C

option 43.3 boot_monitor_image.bin
```

Example 19-11 Incorrect OUI Modifier Usage

```
# Invalid, when both OUI and instance modifier are used in authoring a template,
# "instance" modifier needs to occur before "oui" modifier.

option 43.8 instance 1 00-00-0C

option 43.3 oui 00-00-0C instance 1 boot_monitor_image.bin
```

SNMP VarBind

You must use an object identifier (OID) when specifying DOCSIS Option 11, PacketCable Option 64, or CableHome Option 28. The MIB that contains the OID must be in one of the following MIBs loaded by the RDU. You must specify as much of the OID as needed to uniquely identify it. You can use the name or the number of the OID. The RDU automatically loads these MIBs:

- SNMPv2-SMI
- SNMPv2-TC
- CISCO-SMI
- CISCO-TC
- SNMPv2-MIB
- RFC1213-MIB
- IANAifType-MIB
- IF-MIB

DOCSIS MIBs

These DOCSIS MIBs are loaded into the RDU:

- DOCS-IF-MIB
- DOCS-BPI-MIB
- CISCO-CABLE-SPECTRUM-MIB
- CISCO-DOCS-EXT-MIB
- SNMP-FRAMEWORK-MIB
- DOCS-CABLE-DEVICE-MIB
- CISCO-CABLE-MODEM-MIB

Note In Cisco BAC 4.1, the DOCSIS MIB, DOCS-CABLE-DEVICE-MIB-OBSOLETE (experimental branch) are removed from the RDU default loaded MIBs list since it predates the DOCS-CABLE-DEVICE-MIB (mib2 branch).

In Cisco BAC 4.2, the CL-SP-MIB-CLABDEF-I02-020920 and DOCS-BPI2-MIB-ipcdn-08 MIBs are removed from the RDU default loaded MIBs list since they are the duplicates of CLAB-DEF-MIB and DOCS-BPI2-MIB, respectively.

The references to any fully qualified MIB OIDs from the above removed MIBs should be replaced with the appropriate OIDs from the new MIBs in customer templates and scripts. However, the custom MIB option can be used to include these experimental OIDs. For more details about adding custom MIBs, see Adding SNMP TLVs With Vendor-Specific MIBs, page 19-24.

PacketCable MIBs

These PacketCable (North American) MIBs are loaded into the RDU:

- CLAB-DEF-MIB
- PKTC-MTA-MIB
- PKTC-SIG-MIB
- PKTC-EVENT-MIB

CableHome MIBs

These CableHome MIBs are loaded into the RDU:

- CABH-CAP-MIB
- CABH-CDP-MIB
- CABH-CTP-MIB
- CABH-PS-DEV-MIB
- CABH-QOS-MIB
- CABH-SEC-MIB

These additional MIBs are needed but are not part of the Prime Cable Provisioning product:

- CABH-CTP-MIB needs RMON2-MIB, TOKEN-RING-RMON-MIB
- CABH-SEC-MIB needs DOCS-BPI2-MIB.

Macro Variables

Macro variables are specified as values in templates that let you specify device-specific option values. When a macro variable is encountered in the template, the properties hierarchy is searched for the macro variable name and the value of the variable is then substituted. The variable name is a custom property, which is predefined in the RDU. It must not contain any spaces.

After the custom property is defined, it can be used in this property hierarchy:

- Device properties
- Provisioning Group properties
- Class of Service properties
- DHCP Criteria properties
- Technology defaults, such as PacketCable, DOCSIS, or CableHome
- System defaults

The template parser works bottom up when locating properties in the hierarchy (device first, then the Class of Service, and so on) and converts the template option syntax. The following syntax is supported for macro variables:

- ${var-name}—This syntax is a straight substitution. If the variable is not found, the parser will generate an error.
- ${var-name, ignore}—This syntax lets the template parser ignore this option if the variable value is not found in the properties hierarchy.
- ${var-name, default-value}—This syntax provides a default value if the variable is not found in the properties hierarchy.

Example 19-12 and Example 19-13 illustrate correct and incorrect usage of Option 11.

Example 19-12 Correct Macro Variables Usage

```
# Valid, using macro variable for max CPE's, straight substitution
option 18 ${MAX_CPES}

# Valid, using macro variable for max CPE's, ignore option if variable not found
# option 18 will not be defined in the DOCSIS configuration file if MAX_CPES
# is not found in the properties hierarchy
option 18 ${MAX_CPES, ignore}

# Valid, using macro variable for max CPE's with a default value
option 18 ${MAX_CPES, 1}


# Valid, using macro variable for vendor option
option 43.200 hex ${MACRO_VAR_HEX}

# Valid, using macro variable for vendor option
option 43.201 ascii ${MACRO_VAR_ASCII}

# Valid, using macro variable for vendor option
option 43.202 ip ${MACRO_VAR_IP}

# Valid, using macro variable in double quotes
option 18 "${MAX_CPES}"

# Valid, using macro variable within a value
option 43.131 ascii "hostname ${HOSTNAME}"
```

```
# Valid, using macro variables in multi-valued options
option 11 ${ACCESS_CONTROL_MIB,
.mib-2.docsDev.docsDevMIBObjects.docsDevNmAccessTable.docsDevNmAccessEntry.docsDevNmAccess
Control.1}, Integer, ${ACCESS_CONTROL_VAL, 3}

# Valid, using macro variable in an include statement
include "${EXTRA_TEMPLATE}"
# Valid, using macro variable in an include statement with a default value
include "${EXTRA_TEMPLATE, modem_reset.tmpl}"

# Valid, using macro variable in an include statement with a default value
include "${EXTRA_TEMPLATE, modem_reset}.tmpl"

# Valid, using macro variable in an include statement with an ignore clause
include "${MY_TEMPLATE, ignore}"
```

Example 19-13 Incorrect Macro Variables Usage

```
# Invalid, using macro variable as the option number
option ${MAX_CPES} 1

# Invalid, using macro variable with space in name
option 18 ${MAX CPES}
```

SNMP TLVs

Prime Cable Provisioning supports SNMP TLVs in dynamic template files, using Option 11 and 64, for:

- DOCSIS—From Cisco Prime Cable Provisioning for Cable version 2.0 onwards.
- PacketCable—From Cisco Prime Cable Provisioning version 2.5 onwards.
- CableHome—From Cisco Prime Cable Provisioning version 2.6 onwards.

To validate the syntax of the SNMP TLVs in these template files, Cisco Prime Cable Provisioning requires a MIB file containing the corresponding SNMP OID that is referenced in the SNMP TLV. If a template contains an SNMP TLV with an SNMP OID that cannot be found in a MIB, the SNMP TLV generates a syntax error.

The following sections describe how you can add SNMP TLVs without a MIB or with a vendor-specific MIB.

Adding SNMP TLVs Without a MIB

You can add SNMP TLVs in dynamic configuration files (DOCSIS, PacketCable, CableHome) without requiring the MIB be loaded by the RDU. From within RDU configuration extensions, the functionality can be accessed with the DOCSISOptionFactory interface, using the following method:

```
public OptionValue createOptionValue(OptionSyntax syntax, String optionNumStr, String[]
optionValueList)
```

The public OptionSyntax.SNMP enumerated value can be used in the above method, in conjunction with the optionValueList containing the tuple: OID, Type, Value.

From RDU dynamic configuration templates, the following syntax is used to specify SNMP TLVs that are not validated against the RDU MIBs:

```
option option-number snmp OID, Type, Value
```

Examples:

```
# DOCS-CABLE-DEVICE-MIB:
option 11 snmp .docsDevNmAccessIp.1,IPADDRESS,192.168.1.1

# Arris vendor specific SNMP TLV (OID numbers only, mix names/numbers)
option 11 snmp .1.3.6.1.4.1.4115.1.3.1.1.2.3.2.0, INTEGER, 6
option 11 snmp .enterprises.4115.1.3.1.1.2.3.2.0, INTEGER, 6

# NOTE: trailing colon required for single octet
option 11 snmp .1.3.6.1.2.1.69.1.2.1.6.3, STRING, 'c0:'
```

Table 19-3 describes the allowed SNMP variable type names.

Table 19-3 SNMP Variable Types

IETF standard SMI Data Type	SNMP API name
Integer32	INTEGER
Integer (Enumerated)	INTEGER
Unsigned32	UNSIGNED32
Gauge32	GAUGE
Counter32	COUNTER
Counter64	COUNTER64
Timeticks	TIMETICKS
OCTET STRING	STRING
OBJECT IDENTIFIER	OBJID
IpAddress	IPADDRESS
BITS	STRING

For example, to specify an SMI Integer32 type, the following types are accepted (regardless of case sensitivity): Integer32, INTEGER.

For OCTET STRING type, all of the following types are accepted: OCTET STRING, OCTETSTRING, or STRING.

The custom SNMP TLV template option can be used to specify any SNMP TLV, including those that are present in the RDU MIBs. The custom SNMP TLV error checking is less stringent, and does not detect incorrect scalar/columnar references (for example, .0 versus .n in OID names).

Adding SNMP TLVs With Vendor-Specific MIBs

Adding a MIB to the RDU enables templates to use the human-readable SNMP OID while also permitting macro variables to be used with the SNMP TLV value.

Cisco BACC 2.7 or later

Note The */docsis/mibs/custom/mibList* property is renamed */snmp/mibs/mibList* from Cisco BACC 2.7 onwards.

If you have the MIB corresponding to the SNMP OID that you want to use, you can add the MIB file to the Prime Cable Provisioning RDU. After you add the MIB, any SNMP TLV using an SNMP OID referenced in the new MIB is recognized.

To add a new MIB to the Prime Cable Provisioning RDU:

Step 1 Launch the Prime Cable Provisioning administrator user interface.

Step 2 On the navigation bar, click **Configuration > Defaults**.

Step 3 On the Configure Defaults page that appears, click the System Defaults link on the left pane.

Step 4 In the MIB List field, paste the content of the new MIB at the end.

Step 5 Click **Submit**.

Note In version 2.7 and later, the MIB parsing tool has been enhanced; subsequently, the tool sometimes returns errors on MIB versions that parsed without error previously. If you encounter any errors that you are unable to resolve by editing the new MIB, contact the Cisco Technical Assistance Center.

Debugging the MIB Load Order

Typically, vendors provide several MIBs requiring a specific load order to satisfy inter-MIB dependencies. But because the vendor frequently does not provide the correct load order, you must determine the correct load order yourself. This section describes how you can use Prime Cable Provisioning debugging information to resolve MIB load-order issues.

Note The MIB load order in Prime Cable Provisioning is set by the order in which the MIBs are listed in the:

- *\/docsis/mibs/custom/mibList* property, if you are using Cisco BACC 2.6.x or earlier releases.
- */snmp/mibs/MibList* property, if you are using Cisco BACC 2.7.x or later releases.

You can use the runCfgUtil.sh tool to determine the correct load order for the property specified in the *api.properties* file. The runCfgUtil.sh tool resides in the *BPR_HOME/rdu/bin* directory.

Note This procedure references the */snmp/mibs/MibList* property that Cisco BACC 2.7.x or later releases use. If you are running 2.6.x or earlier releases, ensure that you use the */docsis/mibs/custom/mibList* property.

Step 1 Configure runCfgUtil.sh via the *api.properties* file using configuration content similar to that described in this step. The *api.properties* file enables Prime Cable Provisioning tracing to direct MIB debugging information to the user console.

```
#
# Enable logging to the console
#
/server/log/1/level=Info
/server/log/1/properties=level
/server/log/1/service=com.cisco.csrc.logging.SystemLogService
/server/log/1/name=Console
```

```
#
# Enable trace categories
#
/server/log/trace/rduserver/enable=enabled
#
# The list of MIBs to be added.
#
/snmp/mibs/MibList=arrishdr.mib,arris_cm_capability.mib,arris_mta_device.mib,arris_sip.mib
,arris_cm.mib,pp.mib,blp2.mib,dev0.mib,docs_evnt.mib,qos.mib,test.mib,usb.mib,snmpv2_conf.
mib,rfc1493.mib,rfc1907.mib,rfc2011.mib,rfc2013.mib,rfc2233.mib,rfc2571.mib,rfc2572.mib,rf
c2573.mib,rfc2574.mib,rfc2575.mib,rfc2576.mib,rfc2665.mib,rfc2669.mib,rfc2670.mib,rfc2786.
mib,rfc2851.mib,rfc2933.mib,rfc 3083.mib
```

Step 2 With runCfgUtil.sh so configured, run the tool to encode any template containing an Option 11 or Option 64 (SNMP encoding). The tool attempts to load the MIBs specified within `/snmp/mibs/MibList`, and directs the complete debugging information, along with any MIB load errors, to the user console.

Step 3 Use the error information to massage the MIB order specified within `/snmp/mibs/MibList` until the complete set of MIBs loads without error and the file encode succeeds.

Step 4 Once you determine a successful load order, complete the procedure described in this step based on the Prime Cable Provisioning version you are using:

Cisco BACC 2.7 or later

a. From the administrator user interface, click **Configuration > Defaults**, then the System Defaults link.

b. In the MIB List field, copy the load order information.

The RDU is now configured to encode templates using the vendor-supplied MIBs.

Note You do not need to restart the RDU.

Ensure that you use the */snmp/mibs/mibList* string in the *api.properties* file and the MIB List field.

Cisco BACC 2.6 or earlier

a. Copy the load order information to the */docsis/mibs/custom/mibList* property in the *rdu.properties* file. This file resides in the *BPR_HOME/rdu/conf* directory.

b. Restart the RDU via the Prime Cable Provisioning process watchdog, using the **/etc/init.d/bprAgent restart rdu** command.

The RDU is now configured to encode templates using the vendor-supplied MIBs.

Encoding Types for Defined Options

Table 19-4 identifies the options with defined encoding types.

Table 19-4 Defined Option Encoding Types

Encoding	Input	Examples

Authorization Action	Unsigned 8-bit integer or description string.
	To allow authorization, the values are:
	• 0
	• permit
	To deny authorization, the values are:
	• 1
	• deny
		`0`
		`1`
		`permit`
		`deny`

Access View Control	Unsigned 8-bit integer or description string.
	To include the SNMPv3 Access View subtree from access view, the values are:
	• 1
	• included
	To exclude the SNMPv3 Access View subtree from access view, the values are:
	• 2
	• excluded
		`1`
		`2`
		`included`
		`excluded`

Access View Type	Unsigned 8-bit integer or description string.
	To enable read-only access, the values are:
	• 1
	• Read-only
	To enable read-write access, the values are:
	• 2
	• Read-write
		`1`
		`2`
		`Read-only`
		`Read-write`

Table 19-4 Defined Option Encoding Types (continued)

Encoding	Input	Examples

ActInact	Unsigned 8-bit integer or description string.
	To disable the TLV, the values are:
	• 0
	• Inactive
	To enable the TLV, the values are:
	• 0
	• Active
		`0`
		`1`
		
		`Inactive`
		`Active`

BitFlag8	Unsigned 8-bit integer. The output is a hexadecimal string representation of the value.	`0xFE`
BitFlag32	Unsigned 32-bit integer. The output is a hexadecimal string representation of the value.	`0xFFFF0000`

Boolean	0 for false and 1 for true.	`0`
		`1`

Byte16	16 bytes specified as a hexadecimal string of 32 characters. Is typically used to represent the MIC option of the cable modem and the CMTS. No 0x prefix is allowed.	None. Prime Cable Provisioning automatically calculates the hash for the cable modem and CMTS MIC option.
Bytes	A series of hexadecimal octets. Each octet must be 2 characters.	`000102030405060708`

CPE Access Control	Unsigned 8-bit integer or description string.
	To disable device access control, the values are:
	• 0
	• Disabled
	To enable device access control, the values are:
	• 1
	• Enabled
		`0`
		`1`
		
		`Disabled`
		`Enabled`

DSCClassifier	Unsigned 8-bit integer or DSCClassifier string name. The unsigned integers include: • 0—DSC Add Classifier • 1—DSC Replace Classifier • 2—DSC Delete Classifier	`0`

Table 19-4 Defined Option Encoding Types (continued)

Encoding	Input	Examples

EnableDisable	Unsigned 8-bit integer or description string.
	To disable, the values are:
	• 0
	• Disabled
	To enable, the values are:
	• 1
	• Enabled
		`0`
		`1`
		
		`Disabled`
		`Enabled`

Inet Address Peer	A one-byte InetAddressTypeCode:
	• 1 for IPv4
	• 2 for IPv6
	This value is followed by an IPv4 or an IPv6 internet address.
	As a result, this length is 5 bytes (1+4) for IPv4 and 17 bytes (1+16) for IPv6.
		`1,10.112.125.111`
		`2,0:0:0:0:0:ffff:8190:3426`

IP address	Four unsigned integer 8, dot (`.`) separated.	`10.10.10.1`
IPv6 address	A string representation of an IPv6 address *x:x:x:x:x:x:x:x,* where the *x*s are one to four hexadecimal digits of the eight 16-bit pieces of the address.	`2001:db8:0:0:8:800:200c:417a`

IPv4 or IPv6 address	A string representation of an IPv4 or IPv6 address.	`10.112.125.111`
		`0:0:0:0:0:ffff:8190:3426`

IP Mode	Unsigned 8-bit integer or description string.
	For IPv4 mode, the values are:
	• 0
	• IPv4
	For IPv6 mode, the values are:
	• 1
	• IPv6
		`0`
		`1`
		
		`IPv4`
		`IPv6`

Multiple IP addresses	Comma-separated list of IP addresses.	`10.11.12.13,10.11.12.14`
Multiple IPv6 addresses	Comma-separated list of IPv6 addresses.	`2001:db8:0:0:8:800:200c:417a,ff01:0:0:0:0:0:0:101`

MAC address	Six hexadecimal octets, colon (`:`) or dash (`-`) separated. Each octet must be exactly 2 characters. Colons and dashes must not be mixed.	`00:01:02:03:04:05`
		`00-01-02-03-04-05`

Table 19-4 Defined Option Encoding Types (continued)

Encoding	Input	Examples

MAC address and mask	Twelve octets, colon (:) or dash (–) separated. Each octet must be 2 characters. Colons and dashes must not be mixed. The first six octets represent the MAC address; the last six represent the mask for the MAC address.	`00:01:02:03:04:05:06:07:08:09:0A:0B`
		`00-01-02-03-04-05-06-07-08-09-0A-0B`

NoLV	Type only. Does not include value or length.	`null`
NVTASCII	An ASCII string. The encoded string will not be NULL terminated.	`This is an ASCII string`
OID	An SNMP OID string.	`sysinfo.0`
OIDCF	An SNMP OID string and an unsigned integer (0 or 1), comma separated.	`sysinfo.0,1`

OnOff	Unsigned 8-bit integer.
	To switch on the TLV, the values are:
	• 0
	• On
	To switch off the TLV, the values are:
	• 1
	• Off
		`0`
		`1`
		`On`
		`Off`

OUI	Three hexadecimal octets, colon (:) or dash (–) separated. Each octet must be 2 characters.	`00-00-0C`

RFC868Time	Unsigned 32-bit integer representing the RFC868 time. The output is a date-time string that uses this format: *MM/dd/yyyy HH:mm:ss*.	`0`
		(representing "12/31/1899 19:00:00")
		`4294967295`
		(representing "02/07/2036 01:28:15")

ServiceFlow	Unsigned 8-bit integer or a service flow description string. The output is a service flow that indicates: • 0—Reserved • 1—Undefined (Dependent on CMTS implementation) • 2—Best Effort • 3—Non-real-time polling service • 4—Real-time polling service • 5—Unsolicited grant service with activity detection • 6—Unsolicited grant service	`0`

Table 19-4 Defined Option Encoding Types (continued)

Encoding	Input	Examples
SNMPVarBind	An SNMP OID string, type, and value. Each of these is comma separated. Valid types are: • BITS • Counter • Counter32 • Counter64 • Gauge • Gauge32 • INTEGER • Integer32 • IpAddress • OCTETSTRING • OBJECTIDENTIFIER • Opaque • TimeTicks • Unsigned32 **Note** The OCTETSTRING can be a string that will be converted to a hexadecimal notation without a trailing NULL, octet string for example, or hexadecimal notation contained in single quotation marks, 'aa:bb:cc' for example.	`.experimental.docsDev.docsDevMIBObjects.docsDevNmAccessTable.docsDevNmAccessEntry.docsDevNmAccessStatus.1, INTEGER, 4`
SrvChangeAct	Unsigned 8-bit integer that is restricted to a range from 0 to 3, or a SrvChangeAct description. The output for the description string is: • 0—Add PHS Rule • 1—Set PHS Rule • 2—Delete PHS Rule • 3—Delete all PHS Rules	`0`

Subtype	One or two comma-separated unsigned integer 8.	`12`
		`12,14`

Table 19-4 Defined Option Encoding Types (continued)

Encoding	Input	Examples

Transport address and mask	For IPv4, four-octet IP address in dotted notation followed by the port number, separated by a comma (,).
	For IPv6, in dotted notation or string:
	• Valid IPv6 address in dotted notation followed by the port number, separated by comma (,).
	• A string representation of IPv6 address, followed by the port number, separated by comma (,). For example: *x:x:x:x:x:x:x:x*,1234, where the *x*s are one to four hexadecimal digits of the eight 16-bit pieces of the address.
		IPv4
		`10.112.125.111,5678`
		IPv6
		`2001.db8.0.0.8.800.200c.417a,5678`
		`2001:db8:0:0:8:800:200c:417a,5678`

Unsigned integer 8	0 to 255	`14`
Unsigned integer 16	0 to 65535	`1244`
Unsigned integer 32	0 to 4294967295	`3455335`
Unsigned integer 8 and unsigned integer 16	One unsigned integer 8 and one unsigned integer 16, comma separated.	`3,12324`
Unsigned integer 8 pair	Two unsigned integer 8, comma separated.	`1,3`
Unsigned integer 8 triplet	Three unsigned integer 8, comma separated.	`1,2,3`

Verify	Unsigned 8-bit integer
	To enable verification, the values are:
	• 0
	• Verify
	To disable verification, the values are:
	• 1
	• Don't Verify
	Note The definitions of true and false for the Verify TLV are in line with the DOCSIS 1.1 specification (Option 26.11).
		`0 = verify`
		`1 = don’t verify`

ZTASCII	An ASCII string. The encoded string will be NULL terminated.	`This is an ASCII string`

BITS Value Syntax

When using the BITS type, you must specify either the labels (“interval1 interval2 interval3”) or numeric bit location (“0 1 2”). Note that label values are 1-based and bit values are 0-based.

This is the syntax that uses the bit numbers:

```
option 11 .pktcSigDevR0Cadence.0,STRING,"0 1 2 3 4 5 6 7 8 9 10 11 12 13 14"
```

This is the syntax for the customer octet string (FFFE000000000000) that uses the labels:

```
option 11 .pktcSigDevR0Cadence.0,STRING,"interval1 interval2 interval3
interval4 interval5 interval6 interval7 interval8 interval9 interval10
interval11 interval12 interval13 interval14 interval15"
```

OCTETSTRING Syntax

The OCTETSTRING can be either a string that is converted to hexadecimal notation without a trailing NULL (for example, octet string), or hexadecimal notation contained within single quotation marks (for example, 'aa:bb:cc').

Using Configuration File Utility for Template

You use the configuration file utility to test, validate, and view PacketCable 1.0/1.5/2.0, DOCSIS 1.0/1.1/2.0/3.0, and CableHome template and configuration files. These activities are critical to successfully deploy individualized configuration files. See Template Files–An Overview, page 19-15, for more information on templates.

The configuration file utility is available only when the RDU is installed; the utility is installed in the *BPR_HOME/rdu/bin* directory.

Both the template file being encoded and the binary file being decoded must reside in the directory from which the configuration file utility is invoked.

All examples in this section assume that the RDU is operating and that these conditions apply:

- The Prime Cable Provisioning application is installed in the default home directory (*/opt/CSCObac*).
- The RDU login name is **admin**.
- The RDU login password is **changeme**.

Note Some of the examples in this section were trimmed whenever the omitted information is of no consequence to the example of its outcome. Instances where this occurs are identified by an ellipses (...) that precedes the example summary.

This section discusses these topics:

- Running the Configuration File Utility, page 19-34
- Adding a Template to Cisco Prime Cable Provisioning, page 19-35
- Converting a Binary File to a Template File, page 19-36
- Testing Template Processing for a Local Template File, page 19-37
- Testing Template Processing for an External Template File, page 19-38
- Specifying Macro Variables at the Command Line, page 19-45

Running the Configuration File Utility

In subsequent procedures and examples, the phrase "run the configuration file utility" means to enter the **runCfgUtil.sh** command from the directory specified. To run the configuration file utility, run this command from the *BPR_HOME/rdu/bin* directory:

runCfgUtil.sh *options*

The available *options* include:

- **-c** *shared*—Specifies the CMTS shared secret when parsing a DOCSIS template file. To specify the default shared secret, enter **-c cisco**.
- **-cablehome**—Identifies the input file as a CableHome portal service configuration file. Do not use this with either the **-docsis** or **-pkt** options.
- **-d**—Decodes the binary input file. Do not use this with the **-e** option.
- **-docsis**—Specifies the input file as a DOCSIS configuration file. Do not use this default with the **-pkt** option.
- **-v** *version*—Specifies the DOCSIS version being used. For example, if you are using DOCSIS 1.1, enter **-v 1.1**. If you do not specify the version number, the command defaults to use DOCSIS 3.0. The values that Prime Cable Provisioning supports are 1.0, 1.1, 2.0, and 3.0.
- **-e**—Encodes the template input file. Do not use this default with the **-d** option.
- **-g**—Generates a template file from either a DOCSIS, PacketCable, or CableHome binary file.
- **-h** *host:port*—Specifies the host and port. The default port number is 49187.
- **-i** *device-id*—Identifies the device to use when substituting macro variables during template parsing. For example, if the device MAC address is 1,6,00:00:00:00:00:01, enter **-i 1,6,00:00:00:00:00:01**, or if the device DUID is 00:03:00:01:00:18:68:52:75:c0, enter **-i 00:03:00:01:00:18:68:52:75:c0**. When using this option, you must also use the **-u** and **-p** options, respectively, to specify the username and password. Do not use this with the **-m** option.
- **-l** *filename*—Identifies the input file as being on the local file system. For example, if your input file is called *any_file*, enter **-l any_file**. Do not use this with the **-r** option.
- **-loc** *locale*—Specifies the PacketCable locale such as na, euro, and, ietf (default is na). The default is na. If the MTA is euro-MTA, then the locale should be set to euro.
- **-m** *macros*—Specifies key value pairs for macro variables. The format is key=value. If you require multiple macro variables, use a double comma separator between the key value pairs; for example, key_1=value_1,,key_2=value_2. Do not use this with the **-i** option.
- **-p** *password*—Specifies the password to use when connecting to the RDU. For example, if your password is 123456, enter **-p 123456**.
- **-o** *filename*—Saves a parsed template file as a binary file. For example, if you want the output to be found in a file called *op_file,* enter **-o op_file**.

- **-pkt**—Identifies the input file as a PacketCable MTA configuration file. Do not use this with the **-docsis** option.
- **-r** *filename*—Identifies the input file as a remote file that has been added to the RDU. For example, if your file is called *file25*, enter **-r file25**. When using this option you must also use the **-u** and **-p** options, to specify the username and password, respectively. Do not use this with the **-l** option.
- **-s**—Displays the parsed template or the contents of the binary file in a human-readable format.
- **-t**—Specifies the PacketCable encoding type: **Secure** or **Basic** (the default is Secure).
- **-u** *username*—Specifies the username to use when connecting to the RDU. For example, if your username is admin, enter **-u admin**.
- **-E**—Enables Extended CMTS MIC (EMIC) calculation and identifies the default options for EMIC calculation. The default options are:
  - HMAC type—MMH16
  - EMIC Digest type—Explicit
  - EMIC shared secret as **cisco**.
- **-Ei**—Identifies the EMIC Digest type as implicit for EMIC calculation.
- **-Eh**—Specifies the HMAC type: MD5 or MMH16 (the default is MMH16).
- **-Es** *secret*—Specifies the EMIC shared secret when parsing a DOCSIS template file.

Note The configuration file utility does not include Option 19 (TFTP server timestamp) and Option 20 (TFTP server provisioned modem address) in the template file; the Prime Cable Provisioning TFTP mixing, however, does. Also, options 6 (CM MIC) and 7 (CMTS MIC) are both automatically inserted into the encoded template file. Therefore, you do not have to specify these message integrity checks (MICs).

Adding a Template to Cisco Prime Cable Provisioning

To use the configuration file utility to test Prime Cable Provisioning templates:

Step 1 Develop the template as described in Template Files–An Overview, page 19-15. If the template includes other templates, make sure all the referenced templates are in the same directory.

Step 2 Run the configuration file utility on the local file system. You can check the syntax for the template, or have the configuration file utility process the template as CRS would, and return output.

If the template contains macro variables, perform these operations in the order specified:

a. Test with command line substitution.

b. Test with a device that has been added to your RDU.

Step 3 Add the template (and any included templates that are used) to the RDU.

Step 4 Run the configuration file utility to parse a file. See Testing Template Processing for an External Template File, page 19-38.

If the template contains macro variables, perform these operations in the order specified:

a. Test with command-line substitution.

b. Test with a device that has been added to your RDU.

Step 5 After all tests succeed, configure a Class of Service to use the template.

Converting a Binary File to a Template File

Use the **runCfgUtil.sh** command to convert binary configuration memory files into template files. Prime Cable Provisioning dynamic configuration generation is based on templates that are created. Automatically converting existing, tested, binary files to template files speeds the process and reduces the possibility of introducing errors.

Syntax Description

runCfgUtil.sh -g -l *binary_file* **-o** *template_file*

- **-g**—Specifies that a template file needs to be generated from an input binary file
- **-l** *binary_file*—Specifies the local input file, including the pathname. In all cases, the input binary filename will have a *.cm* file extension; *bronze.cm* for example.
- **-o** *template_file*—Specifies the output template file, including the pathname. In all cases, the output template file will have a *.tmpl* file extension; for example*, test.tmpl*.

To convert a binary file into a template file:

Step 1 Change directory to */opt/CSCObac/rdu/samples/docsis*.

Step 2 Select a template file to use. This example uses an existing binary file called *unprov.cm*.

Step 3 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -g -l unprov.cm -o test.tmpl -docsis
```

-docsis—Specifies the input file to be a DOCSIS configuration file.

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 4.2, Revision: 1.26

#################################################################
## Template File Generator
## Generated on Fri Oct 12 16:12:51 EST 2007
#################################################################

#################################################################
## Each generated option will be represented by the following:
## The first line will represent a description of the
## generated option
## The second line will represent the generated option
## The third line will represent the custom version
## of the generated option
#################################################################

# (3) Network Access Control
```

```
Option 3  01
# Option 3  hex 01

# (4.1) Class ID
Option 4.1 1
# Option 4.1 hex 01

# (4.2) Maximum Downstream Rate
Option 4.2 128000
# Option 4.2 hex 0001F400

# (4.3) Maximum Upstream Rate
Option 4.3 64000
# Option 4.3 hex 0000FA00

# (4.4) Upstream Channel Priority
Option 4.4 1
# Option 4.4 hex 01

# (4.5) Guaranteed Minimum Upstream Channel Data Rate
Option 4.5 0
# Option 4.5 hex 00000000

# (4.6) Maximum Upstream Channel Transmit Burst
Option 4.6 1600
# Option 4.6 hex 0640

# (4.7) Class-of-Service Privacy Enable
Option 4.7 00
# Option 4.7 hex 00

# (11) SNMP MIB Object
Option 11
.iso.org.dod.internet.experimental.docsDev.docsDevMIBObjects.docsDevNmAccessTable.docsDevN
mAccessEntry.docsDevNmAccessStatus.1,INTEGER,createAndGo
# Option 11  hex 3082000F060A2B060103530102010701020104

...

# (18) Maximum Number of CPEs
Option 18  1
# Option 18  hex 01
```

Testing Template Processing for a Local Template File

Use the **runCfgUtil.sh** command to test processing for template files stored on the local file system.

Syntax Description

runCfgUtil.sh -pkt -l *file*

- **-pkt**—Identifies the input file as a PacketCable MTA file.
- **-l**—Specifies that the input file is on the local file system.
- *file*—Identifies the input template file being parsed.

To parse a template file that is on the local file system:

Step 1 Change directory to */opt/CSCObac/rdu/samples/packet_cable*.

Step 2 Select a template file to use. This example uses an existing template file called *unprov_packet_cable.tmpl*. The **-pkt** option is used because this is a PacketCable MTA template.

Step 3 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -pkt -l unprov_packet_cable.tmpl
```

unprov_packet_cable.tmpl—Identifies the input template file being parsed.

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 4.2, Revision: 1.26

Off     File Bytes      Option    Description                       Value

0       FE0101          254       Telephony Config File Start/End   1

3       0B153013060E    11        SNMP MIB Object                   .iso.org.dod.internet.
        2B06010401A30B                                              private.enterprises.ca
        0202010101                                                  bleLabs.clabProject.cl
        0700020102                                                  abProjPacketCable.pktc
                                                                    MtaMib.pktcMtaMibObjec
                                                                    ts .pktcMtaDevBase.
                                                                    pktcMtaDevEnabled.0,IN
                                                                    TEGER,false(2)
...

0 error(s), 0 warning(s) detected. Parsing of unprov_packet_cable.tmpl was successful.
The file unprov_packet_cable.tmpl was parsed successfully in 434 ms.
The parser initialization time was 92 ms.
The parser parse time was 342 ms.
```

Testing Template Processing for an External Template File

Use the **runCfgUtil.sh** command to test processing of external template files.

Syntax Description

runCfgUtil.sh -docsis -r *file* **-u** *username* **-p** *password*

- **-r**—Identifies the input file as a file that has been added to the RDU.
- *file*—Identifies the input template file being parsed.
- **-u** *username*—Specifies the username to use when connecting to the RDU.
- **-p** *password*— Specifies the password to use when connecting to the RDU.
- **-docsis**—Identifies the file as a DOCSIS template.

To parse a template file that has been added to the RDU:

Step 1 Select a template file to use. This example uses an existing template file called *unprov.tmpl*. The **-docsis** option is used because a DOCSIS template is being used.

Step 2 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -docsis -r unprov.tmpl -u admin -p changeme
```

- **unprov.tmpl**—Identifies the input file.
- **admin**—Identifies the default username.
- **changeme**—Identifies the default password.

After running the utility, results similar to these should appear:

Note The results shown here are for illustration only and have been trimmed for brevity.

```
Broadband Access Center Configuration Utility
Version: 4.2, Revision: 1.26

Off    File Bytes      Option   Description                       Value

0      030101          3        Network Access Control            On

3      041F            4        Class of Service

5      010101          4.1      Class ID                          1

8      02040000FA00    4.2      Maximum Downstream Rate           128000 bits/sec

14     03040000FA00    4.3      Maximum Upstream Rate             64000 bits/sec

20     040101          4.4      Upstream Channel Priority         1

...

252    06108506547F    6        CM MIC Configuration Setting      8506547FC9152B44
       C9152B44DB95                                               DB955420843EF6FE
       5420843EF6FE

270    0710644B675B    7        CMTS MIC Configuration Setting    644B675B70B7BD3E
       70B7BD3E09AC                                               09AC210F794A1E8F
       210F794A1E8F

288    FF              255      End-of-Data Marker

289    00              0        PAD

290    00              0        PAD

291    00              0        PAD

0 error(s), 0 warning(s) detected. Parsing of unprov.tmpl was successful.
The file unprov.tmpl was parsed successfully in 375 ms.
The parser initialization time was 63 ms.
The parser parse time was 312 ms.
```

Testing Template Processing for a Local Template File and Adding Shared Secret

Use the **runCfgUtil.sh** command to test processing for a template file and add a shared secret that you specify.

Syntax Description

runCfgUtil.sh -e -docsis -l *file* **-c** *shared*

- **-e**—Identifies the encode option.
- **-docsis**—Identifies the input file as a DOCSIS template file.
- **-l**—Specifies that the input file is on the local file system.
- *file*—Identifies the input template file being parsed.
- **-c**—Specifies the CMTS shared secret when parsing a DOCSIS template file.
- *shared*—Identifies the new shared secret. The default shared secret is **cisco**.

To parse a locally saved template file, and set a user-specified shared secret:

Step 1 Change directory to */opt/CSCObac/rdu/templates*.

Step 2 Select a template file to parse. This example uses an existing template file called *unprov.tmpl*. The **-docsis** option is used because this is a DOCSIS template.

Step 3 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -e -docsis -l unprov.tmpl -c shared
```

- **unprov.tmpl**—Identifies the input file on the local file system.
- **shared**—Identifies that new shared secret.

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 4.2, Revision: 1.26

Off    File Bytes     Option     Description                   Value

0      030100         3         Network Access Control        Off

3      041F           4         Class of Service

5      010101         4.1       Class ID                      1

8      02040001F400   4.2       Maximum Downstream Rate       128000 bits/sec

14     03040000FA00   4.3       Maximum Upstream Rate         64000 bits/sec

20     040101         4.4       Upstream Channel Priority     1

...

252    06108506547F   6         CM MIC Configuration Setting  8506547FC9152B44
       C9152B44DB95                                           DB955420843EF6FE
       5420843EF6FE

270    0710644B675B   7         CMTS MIC Configuration Setting 644B675B70B7BD3E
       70B7BD3E09AC                                           09AC210F794A1E8F
       210F794A1E8F

288    FF             255       End-of-Data Marker

289    00             0         PAD

290    00             0         PAD
```

```
291    00              0         PAD

0 error(s), 0 warning(s) detected. Parsing of unprov.tmpl was successful.
The file unprov.tmpl was parsed successfully in 375 ms.
The parser initialization time was 63 ms.
The parser parse time was 312 ms.
```

Testing Template Processing for a Local Template File and Adding EMIC Shared Secret

Use the **runCfgUtil.sh** command to test the processing for a template file and add a EMIC shared secret that you specify.

Example 1:

This example describes how you can use runCfgUtil.sh command to enable EMIC, and set:

- MMH16 as the HMAC type.
- EMIC Digest Explicit option.
- cisco as the EMIC shared secret for EMIC calculation.

Syntax Description

runCfgUtil.sh -E -docsis -l *filename*

- **-E**—Enables EMIC calculation.
- **-docsis**—Identifies the input file as a DOCSIS template file.
- **-l** *filename*—Specifies the input template file, including the pathname. In all cases, the input template file will have a *.tmpl* file extension; for example, test.tmpl.

To perform EMIC calculation with default settings:

Step 1 Select a template file to use. This example uses an existing template file called *unprov.tmpl*. The **-docsis** option is used because a DOCSIS template is being used.

Step 2 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -E -l test.tmpl
```

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 4.2, Revision: 1.26
Off.   File bytes.     Option. Description.                      Value.
0      030101          3       Network Access Control            On
3      041F            4       Class of Service
5      010101          4.1     Class ID                          1
8      0204001F4000    4.2     Maximum Downstream Rate           2048000 bits/sec
14     03040007D000    4.3     Maximum Upstream Rate             512000 bits/sec
20     040106          4.4     Upstream Channel Priority         6
23     050400000000    4.5     Guaranteed Minimum Upstream       0 bits/sec
                               Channel Data Rate
29     06020640        4.6     Maximum Upstream Channel          1600 bytes
                               Transmit Burst
33     070100          4.7     Class-of-Service Privacy Enable   Disabled
249    120103          18      Maximum Number of CPEs            3
252    2B1C            43      DOCSIS Extension Field
254    0803FFFFFF      43.8    Vendor ID                         FF-FF-FF
259    0615            43.6    Extended CMTS MIC Configuration
                               Setting
```

261	010102	43.6.1	Extended CMTS MIC HMAC type	2
264	020678007BEC 1C80	43.6.2	Extended CMTS MIC Bitmap	78007BEC1C80
272	03081605487D 3D7A9403	43.6.3	Explicit Extended CMTS MIC Digest Subtype	1605487D3D7A9403
282	06108BFC8016 39BE8D7F396E E49D402832FC	6	CM MIC Configuration Setting	8BFC801639BE8D7F 396EE49D402832FC
300	071053F94674 11C355EF01D0 104995AB9797	7	CMTS MIC Configuration Setting	53F9467411C355EF 01D0104995AB9797
318	FF	255	End-of-Data Marker	
319	00	0	PAD	

Syntax Description

Example 2:

This example describes how you can use **runCfgUtil.sh** command to enable EMIC and to change the default settings:

runCfgUtil.sh -E -Ei -Eh *MD5* **-Es** *secret* **-l** *filename*

- **-E**—Enables EMIC calculation.
- **-Ei**—Specifies EMIC Digest type as implicit for EMIC calculation.
- **-l** *filename*—Specifies the input template file, including the pathname. In all cases, the input template file will have a *.tmpl* file extension; for example, test.tmpl.
- **-Eh**—Specifies the HMAC type: MD5 or MMH16 (the default is MMH16).
- **-Es** *secret*—Specifies the EMIC shared secret when parsing a DOCSIS template file.

To perform EMIC calculation using the options which are not included as defaults:

Step 1 Select a template file to use. This example uses an existing template file called *unprov.tmpl*. The **-docsis** option is used because a DOCSIS template is being used.

Step 2 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -E -Ei -Eh MD5 -Es secret -l test.tmpl
```

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 4.2, Revision: 1.26
Off.   File bytes.     Option.   Description.                      Value.
0       030101         3         Network Access Control            On
3      041F            4         Class of Service
5      010101          4.1       Class ID                          1
8      0204001F4000    4.2       Maximum Downstream Rate           2048000 bits/sec
14     03040007D000    4.3       Maximum Upstream Rate             512000 bits/sec
20     040106          4.4       Upstream Channel Priority         6
23     050400000000    4.5       Guaranteed Minimum Upstream       0 bits/sec
                                 ChannelData Rate
29     06020640        4.6       Maximum Upstream Channel          1600 bytes
                                 Transmit Burst
33     070100          4.7       Class-of-Service Privacy Enable   Disabled
249    120103          18        Maximum Number of CPEs            3
252    2B12            43        DOCSIS Extension Field
254    0803FFFFFF      43.8      Vendor ID                         FF-FF-FF
259    060B            43.6      Extended CMTS MIC Configuration
                                 Setting

261    010101          43.6.1    Extended CMTS MIC HMAC type       1
264    020678007BEC    43.6.2    Extended CMTS MIC Bitmap          78007BEC1C80
       1C80
272    06107E6CF875    6         CM MIC Configuration Setting      7E6CF87532C55179
       32C551791CCF                                                1CCF34770C94FA03
       34770C94FA03
290    0710C9F8007A    7         CMTS MIC Configuration Setting    C9F8007A40E49137
       40E4913779D3                                                79D3C1C53599141B
       C1C53599141B
308    FF              255       End-of-Data Marker
309    00              0         PAD
310    00              0         PAD
311    00              0         PAD
```

Specifying Macro Variables at the Command Line

Use the **runCfgUtil.sh** command to specify macro variables.

Syntax Description

runCfgUtil.sh -e -l *file* **-m** *"macros"*

- **-e**—Identifies the encode option.
- **-l**—Specifies the input file is on the local file system.
- *file*—Identifies the input template file being parsed.
- **-m**—Specifies the macro variables to be substituted when parsing a template.
- *"macros"*—Identifies the desired macros. When multiple macro variables are required, insert a double comma separator between each macro.

To specify values for macro variables at the command line:

Step 1 Change directory to */opt/CSCObac/rdu/templates*.

Step 2 Select a template file to use.

Step 3 Identify the macro variables in the template. In this example, the macro variables are macro1 (option 3) and macro11 (option 4.2).

Step 4 Identify the values for the macro variables. The value for macro1 will be set to 1, and the value for macro11 to 64000.

Step 5 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -e -l macro.tmpl -m "macro1=1,,macro11=64000"
```

- **macro.tmpl**—Identifies the input file.
- **macro1=1,,macro11=64000**—Identifies the key value pairs for macro variables. Because multiple macro variables are necessary, a double comma separator is inserted between the key value pairs.

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 4.2, Revision: 1.26

Off     File Bytes      Option    Description                     Value

0       030101          3         Network Access Control          On

3       041F            4         Class of Service

5       010101          4.1       Class ID                        1

8       02040000FA00    4.2       Maximum Downstream Rate         64000 bits/sec

14      03040000FA00    4.3       Maximum Upstream Rate           64000 bits/sec

20      040101          4.4       Upstream Channel Priority       1

...

0 error(s), 0 warning(s) detected. Parsing of macro.tmpl was successful.
The file macro.tmpl was parsed successfully in 854 ms.
The parser initialization time was 76 ms.
The parser parse time was 778 ms.
```

Specifying a Device for Macro Variables

Use the **runCfgUtil.sh** command to specify a device for macro variables.

Syntax Description

runCfgUtil.sh -e -r *file* **-i** *MAC* **-u** *username* **-p** *password*

- **-e**—Identifies the encode option.
- **-r**—Identifies the input file as a file that has been added to the RDU.
- *file*—Identifies the input template file being parsed.
- **-i**—Specifies the device to use when parsing macro variables.
- *MAC*—Identifies the MAC address of the device.
- **-u** *username*—Specifies the username to use when connecting to the RDU.
- **-p** *password*— Specifies the password to use when connecting to the RDU.

To specify a device to be used for macro variable substitution:

Step 1 Select a template file to use. This example uses the existing template file, *macro.tmpl*.

Step 2 Identify the macro variables in the template. In this example, the macro variables are macro1 (option 3) and macro11 (option 4.2).

Step 3 Identify the device to use. This example assumes that the device exists in the RDU and has the macro variables set as properties. The value for macro1 will be set to 1, and the value for macro11 to 64000.

Step 4 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -e -r macro.tmpl -i "1,6,00:01:02:03:04:05" -u admin
-p changeme
```

- **macro.tmpl**—Identifies the input file.
- **1,6,00:01:02:03:04:05**—Identifies the MAC address of the device. The MAC address used here is for example purposes only.
- **admin**—Identifies the default username.
- **changeme**—Identifies the default password.

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 4.2, Revision: 1.26

Off    File Bytes      Option     Description                       Value

0      030101          3         Network Access Control            On

3      041F            4         Class of Service

5      010101          4.1       Class ID                          1

8      02040000FA00    4.2       Maximum Downstream Rate           64000 bits/sec

14     03040000FA00    4.3       Maximum Upstream Rate             64000 bits/sec

20     040101          4.4       Upstream Channel Priority         1

...

0 error(s), 0 warning(s) detected. Parsing of macro.tmpl was successful.
The file macro.tmpl was parsed successfully in 159 ms.
The parser initialization time was 42 ms.
The parser parse time was 117 ms.
```

Specifying Output to a Binary File

Use the **runCfgUtil.sh** command to specify the output of a parsed template as a binary file.

Syntax Description

runCfgUtil.sh -l *input_file* **-o** *output_file*

- **-l**—Specifies that the input file is on the local file system.
- *input_file*—Identifies the input template file being parsed.
- **-o**—Specifies that the parsed template file is to be saved as a binary file.
- *output_file*—Identifies the name of the file in which the binary contents of the parsed template file are stored.

To specify the output from parsing a template to a binary file:

Step 1 Change directory to */opt/CSCObac/rdu/templates*.

Step 2 Select a template file to use.

Step 3 Identify the name of the output file. This example uses *unprov.cm*.

Step 4 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -l unprov.tmpl -o unprov.cm
```

- **unprov.tmpl**—Identifies the existing template file being parsed into a binary file.
- **unprov.cm**—Identifies the output filename to be used.

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 4.2

0 error(s), 0 warning(s) detected. Parsing of unprov.tmpl was successful.
The file unprov.tmpl was parsed successfully in 595 ms.
The parser initialization time was 262 ms.
The parser parse time was 333 ms.
```

Viewing a Local Binary File

Use the **runCfgUtil.sh** command to view a binary file stored in the local system.

Syntax Description

runCfgUtil.sh -d -l *file*

- **-d**—Specifies that the command is going to decode a binary input file for viewing.
- **-l**—Identifies that the input file resides on the local file system.
- *file*—Identifies the existing binary input file to be viewed.

To view a binary file that is on the local file system:

Step 1 Change directory to */opt/CSCObac/rdu/samples/packet_cable*.

Step 2 Select a binary file to view.

Step 3 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -d -l unprov_packet_cable.bin
```

unprov_packet_cable.bin—Identifies the existing binary input file to be viewed.

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 4.2, Revision: 1.26
Warning: Expecting config file of type docsis, but input file is of type pktc1.0.
Decoding as pktc1.0
Off   File Bytes    Optio  Description              Value
                    n
0     FE0101        254    Telephony Config File    1
                           Start/End
      0B153013060E  11     SNMP MIB Object          .iso.org.dod.internet.private.enterpri
3     2B06010401A3                                  ses.cableLabs.clabProject.clabProjPack
      0B0202010101                                  etCable.pktcMtaMib.pktcMtaMibObjects.p
      0700020102                                    ktcMtaDevBase.pktcMtaDevEnabled.0,INTE
                                                    GER,fals e(2)
...
```

Note The warning in this example appears because the default input file is DOCSIS, and this example uses a binary PacketCable file. If you use the **-pkt** option to specify the input file as a PacketCable file, the warning does not appear. For example:

```
/opt/CSCObac/rdu/bin/# runCfgUtil.sh -d -pkt -l unprov_packet_cable.bin
```

Viewing an External Binary File

Use the **runCfgUtil.sh** command to view an external binary file.

Syntax Description

runCfgUtil.sh -d -r *file* **-u** *username* **-p** *password*

- **-d**—Specifies that the command is going to decode a binary input file for viewing.
- **-r**—Identifies the input file as a file that has been added to the RDU.
- *file*—Identifies the existing binary file in the RDU.
- **-u** *username*—Specifies the username to use when connecting to the RDU.
- **-p** *password*— Specifies the password to use when connecting to the RDU.

To view a binary file that has been added to the RDU:

Step 1 Select a binary file to view. This example uses the existing binary file *unprov.cm*, and assumes that the RDU is localhost:49187.

Step 2 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -d -r unprov.cm -u admin -p changeme
```

- **unprov.cm**—Identifies the existing binary file in the RDU.
- **admin**—Identifies the default username.
- **changeme**—Identifies the default password.

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 4.2, Revision: 1.26
Off    File Bytes     Option    Description                     Value
0      030100         3        Network Access Control           Off
3      041F           4        Class of Service
5      010101         4.1      Class ID                         1
8      02040001F400   4.2      Maximum Downstream Rate          128000 bits/sec
14     03040000FA00   4.3      Maximum Upstream Rate            64000 bits/sec
20     040101         4.4      Upstream Channel Priority        1
...
252    06108506547F   6        CM MIC Configuration Setting     8506547FC9152B44
       C9152B44DB95                                             DB955420843EF6FE
       5420843EF6FE
270    0710644B675B   7        CMTS MIC Configuration Setting   644B675B70B7BD3E
       70B7BD3E09AC                                             09AC210F794A1E8F
       210F794A1E8F
288    FF             255      End-of-Data Marker
289    00             0        PAD
290    00             0        PAD
291    00             0        PAD
0 error(s), 0 warning(s) detected. Parsing of unprov.tmpl was successful.
The file unprov.tmpl was parsed successfully in 375 ms.
The parser initialization time was 63 ms.
The parser parse time was 312 ms.
```

Activating PacketCable Basic Flow

Use the **runCfgUtil.sh** command to support the generation and insertion of the PacketCable Basic Flow integrity hash into a Basic Flow static configuration file.

Syntax Description

runCfgUtil.sh -t {**basic** | **secure**} **-pkt -r** *filename* **-u** *username* **-p** *password*

- **basic**—Calculates and inserts a PacketCable Basic Flow integrity hash into an MTA static configuration file.
- **secure**—Stops the insertion of the PacketCable Basic Flow integrity hash into an MTA static configuration file. This is the default setting.
- **-r**—Identifies the input file as a file that has been added to the RDU.
- *filename*—Identifies the input file.
- **-u** *username*—Specifies the username to use when connecting to the RDU.
- **-p** *password*—Specifies the password to use when connecting to the RDU.
- **-pkt**—Identifies the input file as a PacketCable MTA configuration file.

To support the generation and insertion of the PacketCable Basic Flow integrity hash into a Basic flow static configuration file:

Step 1 Select the Basic Flow static configuration file into which you want to insert the PacketCable Basic Flow integrity hash. This example uses the *example_mta_config.tmpl*.

Step 2 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -t basic -pkt -r example_mta_config.tmpl -u admin
-p changeme
```

- **example_mta_config.tmpl**—Identifies the Basic Flow static configuration file.
- **admin**—Identifies the default username.
- **changeme**—Identifies the default password.

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 4.2, Revision: 1.26

Off    File Bytes      Option   Description                      Value

0      FE0101          254      Telephony Config File Start/End  1

3      0B153013060E    11       SNMP MIB Object                  .iso.org.dod.internet.
       2B06010401A3                                              private.enterprises.ca
       0B0202010101                                              bleLabs.clabProject.cl
       0700020101                                                abProjPacketCable.pktc
                                                                 MtaMib.pktcMtaMibObjec
                                                                 ts.pktcMtaDevBase.pktc
                                                                 MtaDevEnabled.0,INTEGE
                                                                 R,true(1)

26     0B2530230610    11       SNMP MIB Object                  .iso.org.dod.internet.
       2B06010401A3                                              private.enterprises.ca
       0B0202020102                                              bleLabs.clabProject.cl
       01010109040F                                              abProjPacketCable.pktc
       434D532E4950                                              SigMib.pktcSigMibObjec
       464F4E49582E                                              ts.pktcNcsEndPntConfig
       434F4D                                                    Objects.pktcNcsEndPntC
                                                                 onfigTable.pktcNcsEndP
                                                                 ntConfigEntry.pktcNcsE
                                                                 ndPntConfigCallAgentId
                                                                 .9,STRING,CMS.IPFONIX.
                                                                 COM

...

371    FE01FF          254      Telephony Config File Start/End  255

0 error(s), 0 warning(s) detected. Parsing of example_mta_config.tmpl was successful.
The file example_mta_config.tmpl was parsed successfully in 100 ms.
The parser initialization time was 44 ms.
The parser parse time was 56 ms.
```

A file with a *.tmpl* extension is assumed to be a dynamic configuration template, for which the Basic hash calculation and insertion occur transparently during template processing; as a result, you can use the same template for provisioning in the Secure and Basic modes.

However, if you want to convert a Secure static binary configuration file to a Basic static configuration file before inserting the hash, follow this procedure:

a. Convert the Secure static file to a template, by using:

# **runCfgUtil -l** *input_static_filename* **-pkt -g -o** *output_template_filename*

b. Convert the Secure static template into a Basic static configuration file, by using:

# **runCfgUtil -t basic -l** *input_template_name* **-pkt -o** *output_Basic_static_filename*

This command calculates and inserts the Basic integrity hash into the Basic static configuration file.

Generating TLV 43s for Multivendor Support

Use the **runCfgUtil.sh** command to generate TLV 43s in order to provide multivendor support.

Syntax Description

runCfgUtil.sh -docsis -r *filename* **-u** *username* **-u** *password*

- **-docsis**—Identifies the input file as a DOCSIS template file.
- *filename*—Identifies the input template file being parsed.
- **-r**—Identifies the input file as a file that has been added to the RDU.
- **-u** *username*—Specifies the username to use when connecting to the RDU.
- **-p** *password*— Specifies the password to use when connecting to the RDU.

To generate TLV 43s using a template file that has been added to the RDU:

Step 1 Select a template file to use. This example uses an existing template file called *test.tmpl*. The **-docsis** option is used because a DOCSIS template is being used.

Step 2 Run the configuration file utility using this command:

```
/opt/CSCObac/rdu/bin# runCfgUtil.sh -docsis -r test.tmpl -u admin -p changeme
```

- **test.tmpl**—Identifies the DOCSIS configuration file.
- **admin**—Identifies the default username.
- **changeme**—Identifies the default password.

After running the utility, results similar to these should appear:

```
Broadband Access Center Configuration Utility
Version: 4.2, Revision: 1.26

Off     File Bytes      Option    Description                       Value

0       2B14            43        DOCSIS Extension Field

2       0803FFFFFF      43.8      Vendor ID                         FF-FF-FF

7       050D            43.5      L2VPN Encoding

9       010502345600    43.5.1    VPNID Subtype                     0234560003
        03

16      0204            43.5.2    NSI Encapsulation Subtype

18      02020019        43.5.2.2  IEEE 802.1Q Format Subtype        25

22      2B0B            43        DOCSIS Extension Field
```

```
24      080300000C      43.8     Vendor ID                        00-00-0C (CISCO
                                                                  SYSTEMS, INC.)
29      010418017A30    43.1     Static Downstream Frequency      402750000
35      2B1D            43       DOCSIS Extension Field
37      080300000C      43.8     Vendor ID                        00-00-0C (CISCO
                                                                  SYSTEMS, INC.)
42      0316626F6F74    43.3     Update Boot Monitor Image        boot_monitor_image.bin
        5F6D6F6E6974
        6F725F696D61
        67652E62696E
66      061071E79068    6        CM MIC Configuration Setting     71E790683DE8B995
        3DE8B9950536                                              05368936F4C5312F
        8936F4C5312F
84      0710DB0EED14    7        CMTS MIC Configuration Setting   DB0EED14B5B3428D
        B5B3428D2B15                                              2B150DA582B41A54
        0DA582B41A54
102     FF              255      End-of-Data Marker
103     00              0        PAD
0 error(s), 0 warning(s) detected. Parsing of test.tmpl was successful.
The file test.tmpl was parsed successfully in 250 ms.
The parser initialization time was 109 ms.
The parser parse time was 141 ms.
```

Using MIBs with Dynamic DOCSIS Templates

For a full list of MIBs that Prime Cable Provisioning ships with, see SNMP VarBind, page 19-20.

You can add MIBs using an application programming interface (API) call or by modifying *rdu.properties*. For more details, see Configuring Euro-PacketCable MIBs, page 7-19.

You can add SNMP TLVs to a template:

- When no MIB is available. See Adding SNMP TLVs Without a MIB, page 19-23.
- With vendor-specific MIBs. See Adding SNMP TLVs With Vendor-Specific MIBs, page 19-24.

CHAPTER 20

CPE Provisioning Overview

This chapter describes the management of customer premises equipment (CPE) using the technologies that the CPE supports for the Cisco 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, and computers.

Prime Cable Provisioning provisions the following device types:

- Cable modems and STBs compliant with DOCSIS 1.0, 1.1, and 2.0
- Embedded Multimedia Terminal Adapters (eMTAs) compliant with PacketCable versions 1.x
- Devices compliant with CableHome 1.0
- Computers

Prime Cable Provisioning supports provisioning and managing of:

- IPv6 devices, which include:
  - Cable modems compliant with DOCSIS 3.0
  - Computers
  - Set-top boxes (STBs)
- 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.

Figure 20-1 illustrates the interaction among the various objects in the device data model.

Figure 20-1 Device Object Model

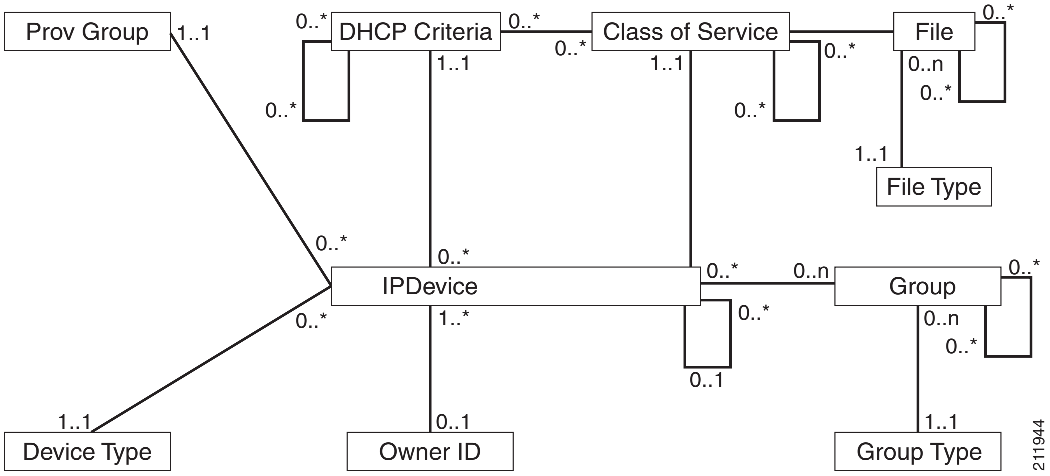


Table 20-1 describes the attributes and relationships unique to each object in the data model.

Table 20-1 Device Object Relationships

Object	Related to...
IPDevice • Could be preprovisioned or self-provisioned (See Device Deployment in Cisco Prime Cable Provisioning, page 20-8). • 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, page 20-3.)	• 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 Prime Cable Provisioning provisioning.

Table 20-2 lists the data that Prime Cable Provisioning discovers for IPv4 devices.

Table 20-2 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
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

Table 20-2 Data Discovered from IPv4 Devices (continued)

Option	Description
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

Table 20-3 lists the data that Prime Cable Provisioning discovers for IPv6 devices.

Table 20-3 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.
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, page 18-4.

For a list of properties that Prime Cable Provisioning extensions use to discover data for DHCPv4 and DHCPv6, see Configuring Cisco Prime Network Registrar Extension, page 6-11.

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, page 20-7
- Property Hierarchy, page 20-7
- Templates and Property Hierarchy, page 20-8
- Custom Properties, page 20-8

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, page 19-33. Template files are identified by a *.tmpl* extension.

For a summary of static provisioning versus dynamic provisioning, see Table 3-6.

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, page 11-3.

Device Deployment in Cisco 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, page 3-37).

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, page 20-10
- Configuration Update Workflow, page 20-14

Initial Configuration Workflows

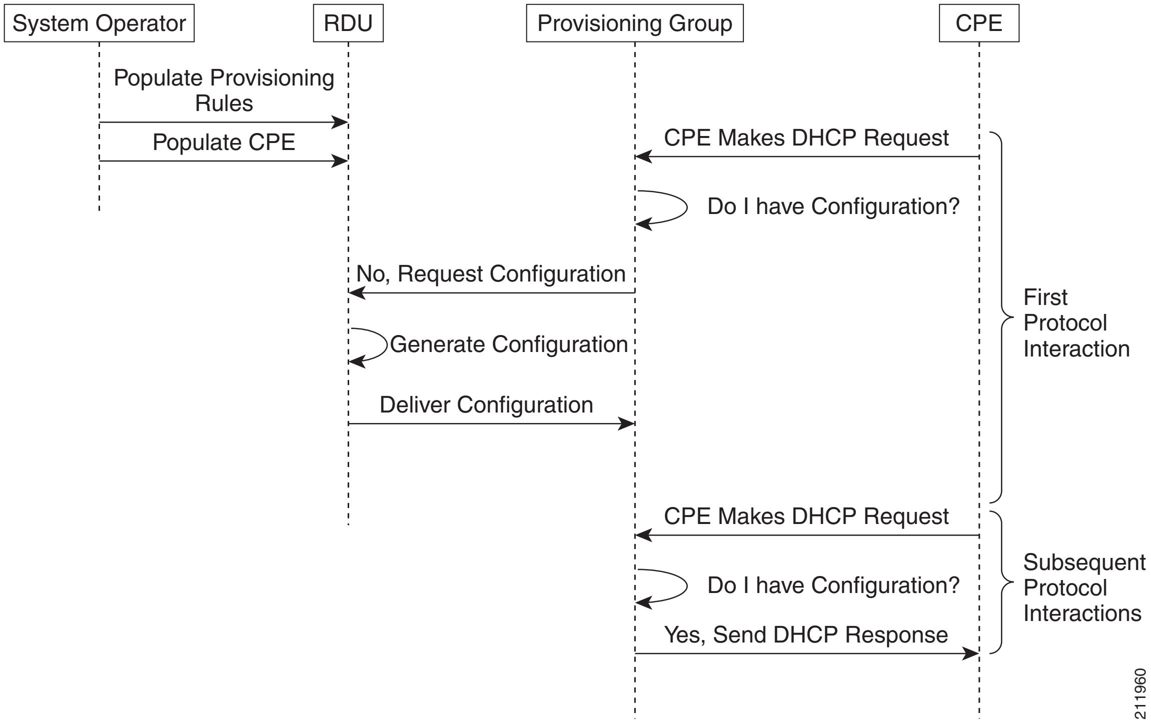
This section describes the configuration workflow when a device is initially installed and booted. The workflows differ based on deployment and registration mode and include:

- Preprovisioned Device Workflow, page 20-10
- Self-Provisioned Device Workflow, page 20-12

Preprovisioned Device Workflow

This section describes the workflow for a preprovisioned device. Figure 20-2 shows a common initial configuration workflow.

Figure 20-2 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.

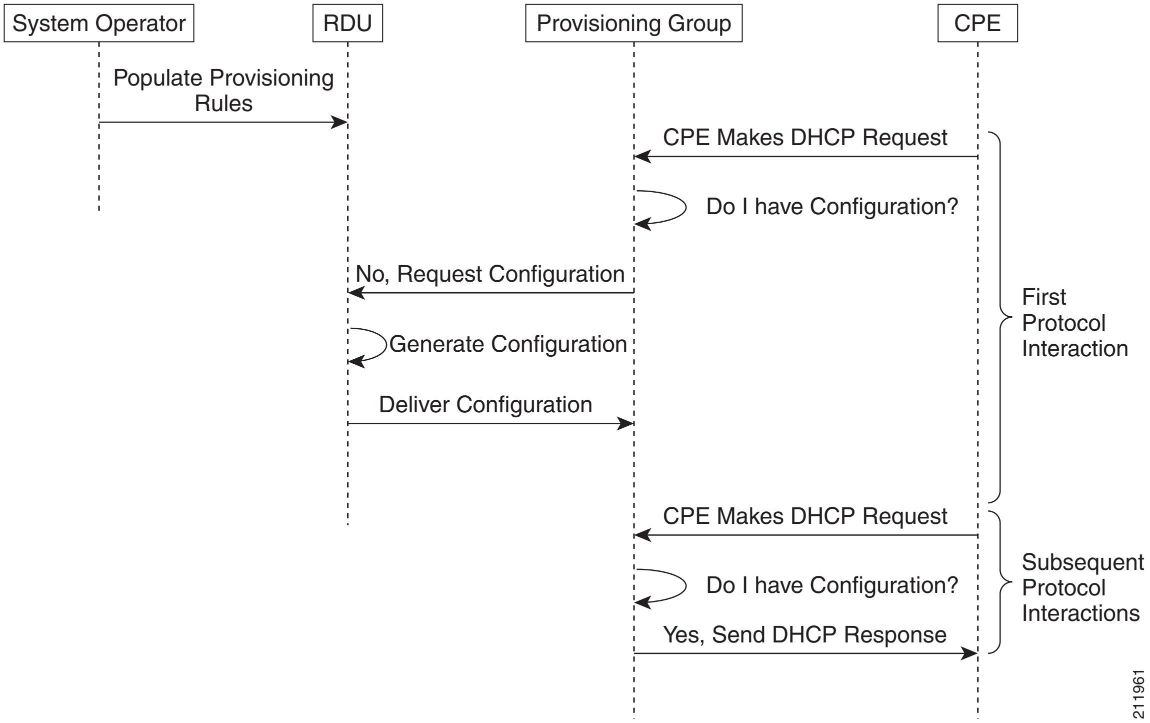
3. The DPE, on receiving the device request, looks up its cache for a configuration for the device. Because the device has never previously contacted the provisioning group, no configuration is found. The Network Registrar extensions in the provisioning group then request the RDU to generate a configuration 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. Figure 20-3 shows a common initial configuration workflow.

Figure 20-3 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.

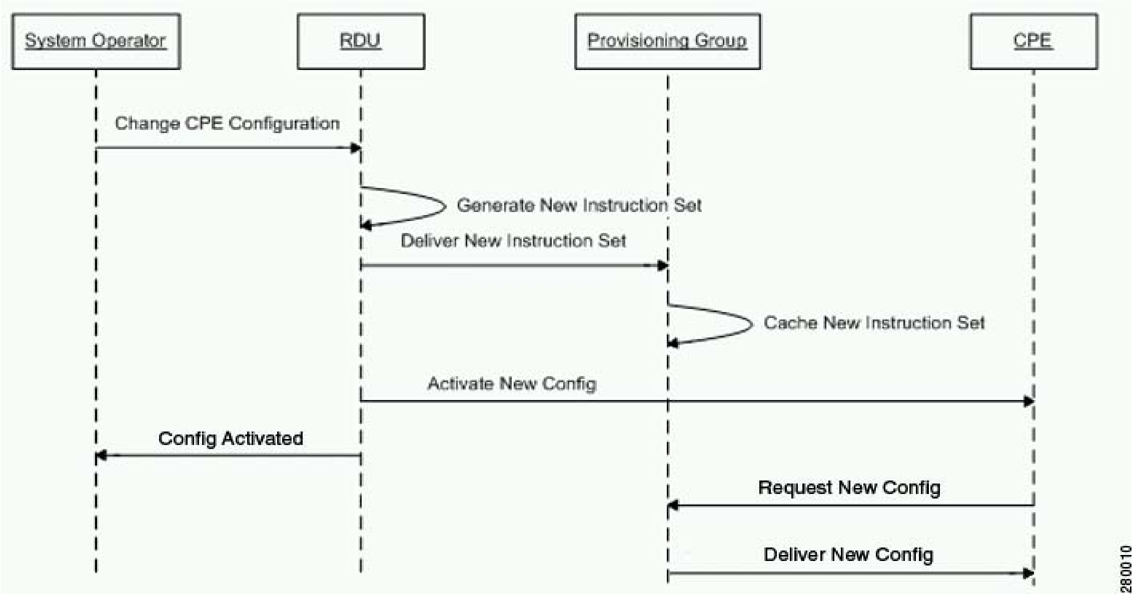
4. The RDU generates a configuration appropriate for the device. The resulting configuration directs DPE responses to various CPE protocol events, such as a DHCP Discover.
5. The device configuration is forwarded to the DPE and cached there. Now, the DPE is programmed to handle subsequent CPE protocol 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.

Figure 20-4 shows the common configuration workflow when you change the configuration of a device that was previously configured.

Figure 20-4 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.

PART 4

Monitoring Prime Cable Provisioning

This part contains the following sections:

- Monitoring Servers Using Admin UI
- Monitoring Servers Using SNMP
- Monitoring Component Logs
- Prime Cable Provisioning Process Watchdog
- Alert and Error Messages

CHAPTER 21

Monitoring Servers Using Admin UI

This chapter describes how you can monitor the performance of the RDU and DPE servers in a Prime Cable Provisioning deployment. These servers are the central RDU server and the DPE servers.

You can check server statistics from the:

- Admin UI
- DPE CLI
- RDU and DPE log files using the Admin UI or the DPE CLI.

Using Admin UI

To view server statistics available on the Admin UI:

1. Click **Server**.

   You get the options: Device Provisioning Engine, Network Registrar DHCP, Provisioning Groups, and Regional Distribution Unit.

2. Click the:
   - Device Provisioning Engine to monitor all DPEs currently registered in the Prime Cable Provisioning.
   - Network Registrar DHCP to monitor all the Network Registrar extension points that have been registered with the RDU.
   - Provisioning Groups to monitor all current provisioning groups.
   - Regional Distribution Unit tab to display Regional Distribution Unit (RDU) status and statistics.

This section describes the Prime Cable Provisioning server pages:

- Monitoring RDU, page 21-2
- Monitoring Provisioning Groups, page 21-3
- Monitoring DPE, page 21-5
- Monitoring CPNR Extension Points, page 21-9

Monitoring RDU

The RDU option, from the **Servers** menu (**Servers > Regional Distribution Unit**), displays details of the RDU as described in Table 21-1.

Table 21-1 View Regional Distribution Unit Details Page

Field or Button	Description
Regional Distribution Unit Details	
Host Name	Identifies the hostname of the system that is running the RDU.
Port	Identifies the RDU listening port number for connections from DPEs. The default port number is 49187, but you can select a different port number during RDU installation.
SSL Port	Identifies the port number used for Secure Socket Layer (SSL) communication.
Secure Communication	Identifies if the communication between the RDU and other servers uses any shared secret for encryption and decryption.
Unsecure Communication	Identifies if the communication between the RDU and other servers uses any shared secret for encryption and decryption.
IP Address	Identifies the IP address assigned to the RDU.
Properties	Identifies the properties configured for the RDU.
Version	Specifies the version of RDU software currently in use.
Up Time	Specifies the total time that the RDU has been operational since its last period of downtime.
State	Identifies whether the RDU is ready to respond to requests. The only state visible on the administrator user interface is Ready.
PACE Statistics	
Batches Processed	Identifies how many individual batches have been processed since the last RDU startup.
Batches Succeeded	Identifies how many individual batches have been successfully processed since the last RDU startup.
Batches Dropped	Identifies how many batches have been dropped since the last RDU startup.
Batches Failed	Identifies how many batches have failed processing since the last RDU startup.
Average Processing Time	Identifies the average time, in milliseconds, that it takes to process the batch excluding the time it spends in the queue if the RDU is too busy.
Average Batch Processing Time	Identifies the average time, in milliseconds, that it takes to process the batch including the time it spends in the queue if the RDU is too busy.
Configuration Regeneration Statistics	

Table 21-1 View Regional Distribution Unit Details Page (continued)

Field or Button	Description
State	Identifies the operational state of the configuration generation service. This could be: • Idle—Specifies that the CRS is not processing regeneration requests. • Regeneration—Specifies that the CRS is processing regeneration requests. • Waiting Regeneration—Specifies that the CRS is unable to regenerate configurations for a device. When the CRS is stuck in this state, check the *rdu.log* file for details.
Requests Processed	Identifies the number of configuration regeneration requests processed since the last RDU startup.
Log Files	
RDU Log File	Features the View Details icon, that, if clicked, displays the View Log File Contents page, which provides details of the *rdu.log* file.
Audit Log File	Features the View Details icon, that, if clicked, displays the View Log File Contents page, which provides details of the *audit.log* file.
Device Statistics	
Note The Device Statistics section appears only when the appropriate devices are present.	
	Identifies the number of devices in the RDU database. The information presented in this area depends on the technologies licensed and configured. These devices may include: • DOCSIS Modems • Computers • PacketCable MTAs • CableHome WAN-Data/WAN-MAN devices • STBs
Note If you have installed JAR files, information on the installed extension JAR files and the loaded extension class files appears after the Device Statistics section.	

Monitoring Provisioning Groups

The Manage Provisioning Groups page (**Servers > Provisioning Groups**) lets you monitor all current provisioning groups. Each provisioning group appearing in this list is a link to its own details page. Click this link to display the details page, which displays details as described in Table 21-2.

Table 21-2 View Provisioning Groups Details Page

Field or Button	Description
Provisioning Group Details	
Name	Identifies the provisioning group name selected from the Manage Provisioning Groups page.

Table 21-2 View Provisioning Groups Details Page (continued)

Field or Button	Description
Primary Device Provisioning Engine	Identifies the hostnames of the DPEs that are primary for this provisioning group. This is an active link that, if clicked, displays the View Device Provisioning Engine Details page.
Secondary Device Provisioning Engine	Identifies the hostnames of the DPEs that are secondary for this provisioning group. This is an active link that, if clicked, displays the View Device Provisioning Engine Details page.
Network Registrar Extension Points	Identifies the hostname of the Network Registrar server assigned to this provisioning group. This is an active link that, if clicked, displays the View Network Registrar Extension Point Details page.
Number of Devices	Specifies the number of devices that belong to this provisioning group.
Lease Query Management	
LeaseQuery AutoConfig	Enables or disables autoconfiguration of lease query addresses. This feature is enabled by default. If you enable this feature, the RDU adjusts its lease query configuration to set both IPv4 and IPv6 address lists from the Network Registrar servers in the provisioning group. If you disable this feature, the RDU does not change its lease query configuration upon registering with the Network Registrar server. **Note** Only if this feature is disabled do subsequent fields in this section appear.
Configured IP Address List (IPv4)	Displays the list of IPv4 addresses on the Network Registrar extensions that the RDU is configured to use for sending DHCPv4 lease query requests.
Configured IP Address List (IPv6)	Displays the list of IPv6 addresses on the Network Registrar extensions that the RDU is configured to use for sending DHCPv6 lease query requests.
Capabilities Management Using these fields, you manually enable or disable capabilities that the provisioning group can provide. If a field is disabled, it means that the provisioning group is not capable of supporting a given device type or feature. Device capabilities are registered with the RDU when the devices startup. See Provisioning Group Capabilities, page 3-39. The values for these fields include: • Enabled—The server is enabled and configured for use. • Disabled—The server supports the feature but is not configured for use. • Not Capable—The server does not support the feature. You must upgrade to Prime Cable Provisioning to enable support for the feature.	
IPv4 - DOCSIS 1.0/1.1	Enables or disables support for DOCSIS 1.0 and 1.1 modems and the computers behind them in the IPv4 mode. To support this feature, you must also enable TFTPv4 on the DPEs in the provisioning group and the Network Registrar DHCP server that supports DHCPv4.
IPv4 - DOCSIS 2.0	Enables or disables support for all DOCSIS 1.0 and 1.1 devices and DOCSIS 2.0 modems in the IPv4 mode.

Table 21-2 View Provisioning Groups Details Page (continued)

Field or Button	Description
IPv4 - DOCSIS 3.0	Enables or disables support for DOCSIS 1.0, 1.1, 2.0, and 3.0 modems in the IPv4 mode and the set-top boxes behind these modems. To support this feature, ensure that all DPEs in the provisioning group run Cisco BAC 4.2 or above.
IPv4 - PacketCable	Enables or disables support for PacketCable MTAs in the IPv4 mode. To support this feature, you must enable PacketCable on all your DPEs in the provisioning group.
IPv4 - CableHome	Enables or disables support for home networking devices in the IPv4 mode.
IPv6 - DOCSIS 3.0	Enables or disables support for DOCSIS 3.0 modems in the IPv6 mode and the set-top boxes behind these modems. To support this feature, you must enable TFTPv6 on the DPEs in the provisioning group and the Network Registrar DHCP server that supports DHCPv6.
Dynamic TFTP Compression	Enables or disables dynamic TFTP compression for DPEs in this provisioning group. If you enable this feature, the dynamic TFTP files that a DPE caches are compressed, thus enhancing DPE performance. Enable dynamic TFTP compression if most of the devices in your network use large files. To use this feature, ensure that all DPEs in the provisioning group run at least Cisco BAC 4.1.
Extended TFTP Config Filename	Enables or disables Extended TFTP Config filename for DPEs in this provisioning group. If you enable this feature, the dynamic TFTP file names can be labeled with dynamic content (for example, COS, vendor-make/model, CPE and so on). This gives flexibility to write your own scripts to define the file names. Enable this feature if you want to customize the Dynamic TFTP filenames. To use this feature, ensure that all DPEs in the provisioning group run at least Prime Cable Provisioning. For more information about Extended TFTP filename. refer to TFTP File-Naming Convention, page 19-12.
Security Domain	Identifies the RBAC domain assigned to the provisioning group, in case Instance Level Access control is enabled.

Monitoring DPE

The Manage Device Provisioning Engines page (**Servers > Device Provisioning Engine**) lets you monitor the list of all DPEs currently registered with the Prime Cable Provisioning database. Each DPE name that appears on this page is a link to another page that displays the details for that DPE. Click the DPE link to display the details page, whose content is similar to the details described in Table 21-3.

Note The RDU determines the names of the Network Registrar extensions and DPEs by performing a reverse DNS lookup on the DPE interfaces through which the DPE contacts the RDU.

Table 21-3 View Device Provisioning Engines Details Page

Field or Button	Description
Device Provisioning Engine Details	
Host Name	Identifies the DPE hostname.
Port	Identifies the DPE port number from which DPE established connection to the RDU.
IP Address	Identifies the IP address of the DPE.
Primary Provisioning Group(s)	Identifies the primary provisioning groups that the selected DPE belongs to. This is an active link that, if clicked, displays the Provisioning Group Details page for that provisioning group.
Secondary Provisioning Group(s)	Identifies the secondary provisioning group (provided that this DPE belongs to a secondary provisioning group) that the selected DPE belongs to. This is an active link that, if clicked, displays the Provisioning Group Details page for that provisioning group.
Properties	Identifies the properties configured for the DPE.
Version	Identifies the version of DPE software currently in use.
Up Time	Specifies the total duration that the DPE has been operational since its last startup.
State	Identifies whether the DPE is ready for operations. These states include: • Registering • Initializing • Synchronizing • Ready • Offline For details on each state, see DPE-RDU Synchronization, page 3-31. **Note** If this field reads Offline, details from the Uptime field onwards do not appear. The DPE is prepared to service client requests in any state except Offline.
Secure Communication with RDU	Identifies if secure communication with the RDU is enabled or disabled.
Security Domain	Identifies the RBAC domain assigned to the DPE, in case Instance Level Access control is enabled.
Protocol Services This section specifies the status of the TFTP and ToD protocols on the DPE.	
TFTPv4	Specifies if TFTPv4 is enabled or disabled on the DPE.
TFTPv6	Specifies if TFTPv6 is enabled or disabled on the DPE.
ToDv4	Specifies if ToDv4 is enabled or disabled on the DPE.
ToDv6	Specifies if ToDv6 is enabled or disabled on the DPE.

Table 21-3 View Device Provisioning Engines Details Page (continued)

Field or Button	Description
Registered Capabilities	
This section specifies the capabilities that all DPEs in this provisioning group registered with the RDU.	
IPv4 - DOCSIS 1.0/1.1	Identifies whether the DOCSIS 1.0 and 1.1 versions are enabled on this DPE in the IPv4 mode.
IPv4 - DOCSIS 2.0	Identifies whether the DOCSIS 2.0 version is enabled on this DPE in the IPv4 mode.
IPv4 - DOCSIS 3.0	Identifies whether the DOCSIS 3.0 version is enabled on this DPE in the IPv4 mode.
IPv4 - PacketCable	Identifies whether the PacketCable voice technology is enabled on this DPE in IPv4 mode.
IPv4 - CableHome	Identifies whether the home networking technology is enabled on this DPE in IPv4 mode.
IPv6 - DOCSIS 3.0	Identifies whether the DOCSIS 3.0 version is enabled on this DPE in the IPv6 mode.
Dynamic TFTP Compression	Identifies whether dynamic TFTP compression is enabled on this DPE. By enabling this feature, you can compress the size of dynamic configurations that are stored at the DPE. When used with dynamic TFTP configuration, this feature dramatically reduces the size of the DPE cache. **Note** You can enable this feature from the **Servers > Provisioning Groups** page but only when all DPEs in the provisioning group support it. For details, see Provisioning Group Capabilities, page 3-39.
Extended TFTP Config Filename	Identifies whether Extended TFTP Config Filename is enabled on this DPE. If you enable this feature, the dynamic TFTP file names can be labeled with dynamic content (for example, COS, vendor-make/model, CPE and so on).
Log File	
DPE Log File	Features the View Details icon that if clicked displays the View Log File Contents page, which provides details of *dpe.log*.
Cache Statistics	
Hits	Identifies the number of cache hits that occurred since the last time the DPE was started.
Misses	Identifies the number of cache misses that occurred since the last time the DPE was started.
Lease Updates	Identifies the number of IPv4 and IPv6 leases that were updated.
Files	Identifies the number of cache files that are currently stored in the DPE.
Configurations	Identifies how many device configuration files are saved in cache.
TFTP Statistics v4	
Packets Received	Identifies the number of TFTPv4 packets that were received by the selected DPE.

Table 21-3 View Device Provisioning Engines Details Page (continued)

Field or Button	Description
Packets Dropped	Identifies the number of TFTPv4 packets that were dropped because of an overloaded DPE.
Packets Successful	Identifies the number of TFTPv4 packets that were transmitted successfully.
Packets Failed	Identifies the number of TFTPv4 packets that failed during transmission.
TFTP Statistics v6	
Packets Received	Identifies the number of TFTPv6 packets that were received by the selected DPE.
Packets Dropped	Identifies the number of TFTPv6 packets that were dropped because of an overloaded DPE.
Packets Successful	Identifies the number of TFTPv6 packets that were transmitted successfully.
Packets Failed	Identifies the number of TFTPv6 packets that failed during transmission.
Time of Day Statistics v4	
Packets Received	Identifies the number of Time of Day v4 packets that were received by the selected DPE.
Packets Dropped	Identifies the number of Time of Day v4 packets that were dropped because of an overloaded DPE.
Packets Successful	Identifies the number of Time of Day v4 packets that were transmitted successfully.
Packets Failed	Identifies the number of Time of Day v4 packets that failed during transmission.
Time of Day Statistics v6	
Packets Received	Identifies the number of Time of Day v6 packets that were received by the selected DPE.
Packets Dropped	Identifies the number of Time of Day v6 packets that were dropped because of an overloaded DPE.
Packets Successful	Identifies the number of Time of Day v6 packets that were transmitted successfully.
Packets Failed	Identifies the number of Time of Day v6 packets that failed during transmission.
PacketCable SNMP Statistics	
SNMP Informs Successful	Identifies the number of inform requests that were successfully sent.
SNMP Sets Successful	Identifies the number of successful SNMP sets.
SNMP Configuration Informs Successful	Identifies the number of SNMP informs received from PacketCable MTAs indicating that they were successfully provisioned.
SNMP Configuration Informs Failed	Identifies the number of SNMP informs received from PacketCable MTAs indicating that they failed to be provisioned.
PacketCable MTA Statistics	

Table 21-3 View Device Provisioning Engines Details Page (continued)

Field or Button	Description
MTA AP Requests Received	Specifies the number of AP-REQ messages received by the DPE from the MTA.
MTA AP Responses Sent	Specifies the number of AP-REP messages sent by the DPE to the MTA.
PacketCable KDC Statistics	
KDC FQDN Requests Received	Specifies the number of FQDN-REQ messages sent by the KDC to the DPE.
KDC FQDN Responses Sent	Specifies the number of FQDN-REP messages sent by the DPE to the KDC.
Configured Network Interfaces	
Provisioning Group Communication	Specifies details related to the provisioning group to which the DPE belongs.
IPv4 Provisioning	Specifies details of the DPE interface that is configured for IPv4 provisioning. These details are: • IPv4 address • Port number • FQDN **Note** This section appears only if the DPE interface is configured for IPv4 provisioning.
IPv6 Provisioning	Specifies details of the DPE interface that is configured for IPv6 provisioning. These details are: • IPv6 address • Port number • FQDN **Note** This section appears only if the DPE interface is configured for IPv6 provisioning.

Monitoring CPNR Extension Points

The Manage Network Registrar Extension Points page (**Servers > Network Registrar DHCP**) lists the extension points for all Network Registrar servers that have been registered with the RDU, and are configured for use with Prime Cable Provisioning. Network Registrar servers automatically register with the RDU when those servers are started.

Each Network Registrar extension point that appears on this page is a link to a secondary page that displays details of that extension point. Click the Network Registrar extension point link to display the details page, which displays details as described in Table 21-4.

Table 21-4 View Network Registrar Extension Point Details Page

Field or Button	Description
Network Registrar Extension Point Details	
Host Name	Displays the hostname of the system running Network Registrar.
IP Address	Identifies the IP address of the Network Registrar server.
Provisioning Group	Identifies the provisioning group for the Network Registrar server. This is an active link that, if clicked, displays the Provisioning Group Details page for that provisioning group.
Properties	Identifies the properties that are applied to the Network Registrar server.
Version	Identifies the extension point software currently in use.
Up Time	Specifies the total time that the Network Registrar extension point has been operational since its last startup. This time is indicated in hours, minutes, and seconds.
State	Identifies whether the DPE is ready for operations. These states include: • Registering • Initializing • Synchronizing • Ready • Offline For details on each state, see DPE-RDU Synchronization, page 3-31. **Note** If this field reads Offline, the options from the Uptime field onwards do not appear. The DPE is prepared to service client requests in any state except Offline.
Secure Communication with RDU	Identifies if secure communication with the RDU is enabled or disabled.
Security Domain	Identifies the RBAC domain assigned to the Network Registrar, in case Instance Level Access control is enabled.
Protocol Services	
DHCPv4	Identifies if DHCPv4 is enabled or disabled.
DHCPv6	Identifies if DHCPv6 is enabled or disabled.
Registered Capabilities	
IPv4 - DOCSIS 1.0/1.1	Identifies whether the DOCSIS 1.0 and 1.1 versions are enabled in the IPv4 mode on the DPE that connects to the Network Registrar server.
IPv4 - DOCSIS 2.0	Identifies whether the DOCSIS 2.0 version is enabled in the IPv4 mode on the the DPE that connects to the Network Registrar server.
IPv4 - DOCSIS 3.0	Identifies whether the DOCSIS 3.0 version is enabled in the IPv4 mode on the DPE that connects to the Network Registrar server.

Table 21-4 View Network Registrar Extension Point Details Page (continued)

Field or Button	Description
IPv4 - PacketCable	Identifies whether the PacketCable voice technology is enabled in the IPv4 mode on the DPE that connects to the Network Registrar server.
IPv4 - CableHome	Identifies whether the home networking technology is enabled in IPv4 mode on the DPE that connects to the Network Registrar server.
IPv6 - DOCSIS 3.0	Identifies whether the DOCSIS 3.0 version is enabled in the IPv6 mode on the DPE that connects to the Network Registrar server.
Network Registrar Extension Point Statistics	
DHCPv4 Packets Received	Identifies the number of DHCPv4 packets that were received.
DHCPv4 Packets Ignored	Identifies the number of DHCPv4 packets that were ignored.
DHCPv4 Packets Dropped	Identifies the number of DHCPv4 packets that were dropped.
DHCPv4 Packets Successful	Identifies the number of DHCPv4 packets that transferred successfully.
DHCPv4 Packets Failed	Identifies the number of DHCPv4 packets that failed to be transferred.
DHCPv6 Packets Received	Identifies the number of DHCPv6 packets that were received.
DHCPv6 Packets Ignored	Identifies the number of DHCPv6 packets that were ignored.
DHCPv6 Packets Dropped	Identifies the number of DHCPv6 packets that were dropped.
DHCPv6 Packets Successful	Identifies the number of DHCPv6 packets that transferred successfully.
DHCPv6 Packets Failed	Identifies the number of DHCPv6 packets that failed to be transferred.
Device Provisioning Engine Details **Note** The following fields appear for each DPE that connects with the Network Registrar server.	
DPE	Identifies the IP address of the DPE.
Port	Identifies the port number from which the DPE established a connection to the RDU.
Type	Identifies whether this DPE is a primary or secondary DPE.
Status	Identifies whether the DPE is operational.

Using DPE CLI

To monitor the status of the DPE server, run the **show dpe** command to check if the DPE is running and displays the state of the process and, if running, its operational statistics. See Example 21-1.

Note This command does not indicate if the DPE is running successfully, only that the process itself is currently executing. However, when the DPE is running, you can use statistics that this command prints to determine if the DPE is successfully servicing requests.

Example 21-1 show dpe Output

This result occurs when the DPE is running.

```
dpe# show dpe
BAC Agent is running
Process dpe is running
Version BAC 4.2 (SOL_CBAC4_0_L_000000000000).
Caching 1 device configs and 1 external files.
0 sessions succeed and 0 sessions failed.
0 file requests succeed and 0 file requests failed.
0 immediate proxy operations received: 0 succeed, and 0 failed.
Connection status is Ready.
Running for 4 hours 30 mins 16 secs.
```

This result occurs when the DPE is not running.

```
dpe_host# show dpe
BAC Agent is running
Process dpe is not running
```

Note For more information, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.

CHAPTER 22

Monitoring Servers Using SNMP

Prime Cable Provisioning supports management of servers via SNMP. Specifically, an SNMP-based management system can be used to monitor Prime Cable Provisioning server state, license utilization information, server connections, and server-specific statistics.

SNMP Agent

Prime Cable Provisioning provides basic SNMP v2-based monitoring of the RDU and DPE servers. The Prime Cable Provisioning SNMP agents support SNMP informs and traps, collectively called notifications. You can configure the SNMP agent on the DPE using snmp-server CLI commands, and on the RDU using the SNMP configuration command-line tool.

For additional information on the SNMP configuration command-line tool, see Using snmpAgentCfgUtil.sh Tool, page 22-3. For additional information on the DPE CLI, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.

MIB Support

Prime Cable Provisioning supports several different MIBs. Table 22-1 summarizes MIB support for each Prime Cable Provisioning component.

Table 22-1 Prime Cable Provisioning-Supported MIBs

Component	MIBs Supported
DPE	CISCO-BACC-SERVER-MIB
	CISCO-BACC-DPE-MIB
RDU	CISCO-BACC-SERVER-MIB
	CISCO-BACC-RDU-MIB

The SNMP agent supports the CISCO-BACC-SERVER-MIB. This MIB defines the managed objects that are common to all servers on Prime Cable Provisioning. This MIB supports the monitoring of multiple Prime Cable Provisioning servers when they are installed on the same device. The `ciscoBaccServerStateChanged` notification is generated every time a server state change occurs.

The RDU SNMP agent supports the CISCO-BACC-RDU-MIB, which defines managed objects for the RDU. This MIB defines statistics related to the state of the RDU and the statistics on the communication interface between the RDU and DPE and between the RDU and Network Registrar.

The SNMP agent generates a `cnaHealthNotif` trap that announces that the RDU server has started, shut down, or failed, or there is a change in the exit status.

The DPE SNMP agent supports the CISCO-BACC-DPE-MIB, which defines managed objects for the components installed on a DPE. The DPE manages local caching of device configurations and configuration files used by all supported devices. This MIB provides some basic DPE configuration and statistics information, including entries for TFTP and ToD servers.

The SNMP agent also supports the CISCO-NMS-APPL-HEALTH-MIB, which defines the Cisco NMS application health status notifications and related objects. These notifications are sent to the OSS/NMS to inform them about the NMS application status, including: started, stopped, failed, busy, or any abnormal exit of applications. The default MIB is MIB-II.

Note For a description of all objects, see the corresponding MIB files in the *BPR_HOME/rdu/mibs* directory.

Table 22-2 lists the Prime Cable Provisioning RDU SNMP Traps

Table 22-2 Cisco Prime Cable Provisioning RDU SNMP Traps

MIB	Trap Name	Trap OID	Sub Type Varbind OID	Sub Type Varbind Value	Sub Type Varbind Value Description	Trap Description
CISCO-BACC-RDU-MIB	ciscoBaccRduLicenseLimit	.1.3.6.1.4.1.9.9.353.0.0	.1.3.6.1.4.1.9.9.353.1.1.2.1.2.(1-n) (LicenseName)	*[Technology name]*	Indicates the corresponding technology name of the license. For example, DOCSIS, PacketCable and so on.	The notification appears when the number of devices exceeds the limit allowed by the license for a specific technology.
			.1.3.6.1.4.1.9.9.353.1.1.2.1.3.(1 - n) (LicenseMaxAllowed)	0..4294967295	Indicates the total number of devices or server components allowed for the technology.	
			.1.3.6.1.4.1.9.9.353.1.1.2.1.4.(1 - n) (cbrLicenseUsage)	0..4294967295	Indicates the total number of licenses of specific technology type already in use.	

MIB	Trap Name	Trap OID	Sub Type Varbind OID	Sub Type Varbind Value	Sub Type Varbind Value Description	Trap Description
CISCO-BACC-SERVER-MIB	ciscoBaccServerStateChanged	.1.3.6.1.4.1.9.9.349.0.0	.1.3.6.1.4.1.9.9.349.1.1.1.3.(1 - n) (cbsState)	1..8	Indicates the status of the server.	This notification appears when the status of the server is changed: • Unknown (1) • initializing (2) • disconnected (3) • shuttingDown(4) • readyOverloaded (5) • ready (6) • offline (7) • unlicensed (8)
			.1.3.6.1.4.1.9.9.349.1.1.1.6.(1 - n) (cbsServerType)	*[ServerType]* RDU,DPE,etc.	A unique name identifying the type of the server. For example: RDU, DPE and so on.	
			.1.3.6.1.2.1.1.5.0 (sysName)	DisplayString (SIZE (0..255))	An administratively-assigned name for the managed node. By convention, this is the fully-qualified domain name of the node. If the name is unknown, the value is a zero-length string.	

Using snmpAgentCfgUtil.sh Tool

You can use the **snmpAgentCfgUtil.sh** tool to manage the SNMP agent installed on a Solaris or Linux computer. Using this tool, which resides in the *BPR_HOME/snmp/bin* directory, you can add (or remove) your host to a list of other hosts that receive SNMP notifications, and start and stop the SNMP agent process. This tool should be run from the local directory.

Note The default port number of an SNMP agent running on a Solaris or Linux computer is 8001.

You can use the RDU SNMP agent for:

- Adding a Host, page 22-4
- Deleting a Host, page 22-4
- Adding an SNMP Agent Community, page 22-5
- Deleting an SNMP Agent Community, page 22-5
- Starting the SNMP Agent, page 22-6
- Stopping the SNMP Agent, page 22-6
- Configuring an SNMP Agent Listening Port, page 22-6

- Changing the SNMP Agent Location, page 22-7
- Setting Up SNMP Contacts, page 22-7
- Displaying SNMP Agent Settings, page 22-8
- Specifying SNMP Notification Types, page 22-8

Adding a Host

You use this command to add the host address to the list of hosts that receive SNMP notifications from the SNMP agent.

Syntax Description

snmpAgentCfgUtil.sh add host *ip-addr* **community** *community* [**udp-port** *port*]

- *ip-addr*—Specifies the IP address of the host to which notifications are sent.
- *community*—Specifies the community (read or write) to be used while sending SNMP notifications.
- *port*—Identifies the UDP port used for sending the SNMP notifications.

Examples

```
# ./snmpAgentCfgUtil.sh add host 10.10.10.5 community trapCommunity udp-port 162
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see Chapter 23, "Prime Cable Provisioning Process Watchdog."

Deleting a Host

You use this command to remove a host from the list of those receiving SNMP notifications from the SNMP agent.

Syntax Description

snmpAgentCfgUtil.sh delete host *ip-addr*

ip-addr—Specifies the IP address of the host that you want to delete from the list of hosts.

Examples

```
# ./snmpAgentCfgUtil.sh delete host 10.10.10.5
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see Chapter 23, "Prime Cable Provisioning Process Watchdog."

Adding an SNMP Agent Community

You use this command to add an SNMP community string to allow access to the SNMP agent.

Syntax Description

snmpAgentCfgUtil.sh add community *string* [**ro** | **rw**]

- *string*—Identifies the SNMP community.
- **ro**—Assigns a read-only (**ro**) community string. Only *get* requests (queries) can be performed. The ro community string allows *get* requests, but no *set* operations. The NMS and the managed device must reference the same community string.
- **rw**—Assigns a read-write (**rw**) community string. SNMP applications require read-write access for *set* operations. The **rw** community string enables write access to OID values.

Note The default **ro** and **rw** community strings are `baccread` and `baccwrite`, respectively. We recommend that you change these values before deploying Prime Cable Provisioning.

Examples

```
# ./snmpAgentCfgUtil.sh add community fsda54 ro
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see Chapter 23, "Prime Cable Provisioning Process Watchdog".

Deleting an SNMP Agent Community

You use this command to delete an SNMP community string to prevent access to the SNMP agent.

Syntax Description

snmpAgentCfgUtil.sh delete community *string* [**ro** | **rw**]

- *string*—Identifies the SNMP community.
- **ro**—Identifies the specified community as a read-only one.
- **rw**—Identifies the specified community as a read-write one.

Note See Adding an SNMP Agent Community, page 22-5, for additional information on the **ro** and **rw** community strings.

Examples

```
# ./snmpAgentCfgUtil.sh delete community fsda54 ro
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see Chapter 23, "Prime Cable Provisioning Process Watchdog.".

Starting the SNMP Agent

You use this command to start the SNMP agent process on a Solaris or Linux computer on which Prime Cable Provisioning is installed.

Note You can also start the SNMP agent by invoking the Prime Cable Provisioning process watchdog using the **/etc/init.d/bprAgent start snmpAgent** command. For more information, see Using Prime Cable Provisioning Process Watchdog from CLI, page 23-2.

Examples

```
# ./snmpAgentCfgUtil.sh start
Process snmpAgent has been started
```

Stopping the SNMP Agent

You use this command to stop the SNMP agent process on a Solaris or Linux computer on which Prime Cable Provisioning is installed.

Note You can also stop the SNMP agent by invoking the Prime Cable Provisioning process watchdog using the **/etc/init.d/bprAgent stop snmpAgent** command. For more information, see Using Prime Cable Provisioning Process Watchdog from CLI, page 23-2.

Examples

```
# ./snmpAgentCfgUtil.sh stop
Process snmpAgent has stopped
```

Configuring an SNMP Agent Listening Port

You use this command to specify the port number that the SNMP agent will listen to. The default port number used by RDU SNMP agent is 8001.

Syntax Description

snmpAgentCfgUtil.sh udp-port *port*

port identifies the port number that the SNMP agent will listen to.

Examples

```
# ./snmpAgentCfgUtil.sh udp-port 8001
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see Chapter 23, "Prime Cable Provisioning Process Watchdog.".

Changing the SNMP Agent Location

You use this command to enter a string of text that indicates the location of the device running the SNMP agent. This could, for example, be used to identify the physical location of the device. You can enter any character string that is fewer than 255 characters.

Syntax Description

snmpAgentCfgUtil.sh location *location*

location is the character string identifying the agent's location.

Examples

In this example, the physical location of the SNMP agent is in an equipment rack identified as rack 5D:

```
# ./snmpAgentCfgUtil.sh location "equipmentrack5D"
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see Chapter 23, "Prime Cable Provisioning Process Watchdog.".

Setting Up SNMP Contacts

You can use this command to enter a string of text that identifies the contact person for the SNMP agent, together with information on how to contact this person. This could, for example, be used to identify a specific person including that person's telephone number. You can enter any character string that is fewer than 255 characters.

Syntax Description

snmpAgentCfgUtil.sh contact *contact-info*

contact-info is the character string identifying the individual to contact concerning the SNMP agent.

Examples

In this example, the contact name is Terry and the telephone extension is 1234:

```
# ./snmpAgentCfgUtil.sh contact "Terry-ext1234"
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see Chapter 23, "Prime Cable Provisioning Process Watchdog.".

Displaying SNMP Agent Settings

You use this command to display all current SNMP settings.

Examples

```
# ./snmpAgentCfgUtil.sh show
Location                        : equipmentrack5D
Contact                         : Terry-ext1234
Port Number                     : 8001
Notification Type               : trap
Notification Recipient Table   :
        [ Host IP address, Community, UDP Port ]
        [ 10.10.10.5 , trapCommunity , 162 ]
Access Control Table            :
        Read Only Communities
                baccread
        Read Write Communities
                baccwrite
```

Specifying SNMP Notification Types

You use this command to specify the types of notifications (traps or informs) that will be sent from the SNMP agent. By default, traps are sent, though you can set the agent to send SNMP informs instead.

Note For the SNMP trap feature to work, you must enable the notification flag. In other words, the value for the MIB variable 0cbsNotifEnableFlags (OID = .1.3.6.1.4.1.9.9.349.1.1.1.1.5.1) must be set to 1.

Syntax Description

snmpAgentCfgUtil.sh inform [**retries timeout**] | **trap**

Where the parameter is the backoff timeout between retries.

Examples

```
# ./snmpAgentCfgUtil.sh inform retries 3 timeout 1000
OK
Please restart [stop and start] SNMP agent.
```

Note The changes that you introduce through this command do not take effect until you restart the SNMP agent by using the **/etc/init.d/bprAgent restart snmpAgent** command. For detailed information, see Chapter 23, "Prime Cable Provisioning Process Watchdog."

Use the **snmpAgentCfgUtil.sh show** command to verify your configuration settings.

```
# ./snmpAgentCfgUtil.sh show
Location                      : equipmentrack5D
Contact                       : Terry-ext1234
Port Number                   : 8001
Notification Type             : inform
Notification Retries          : 3
Notification Timeout          : 1000
Notification Recipient Table  :
        [ Host IP address, Community, UDP Port ]
        [ 10.10.10.5 , trapCommunity , 162 ]
Access Control Table          :
        Read Only Communities
                baccread
        Read Write Communities
                baccwrite
```

CHAPTER 23

Prime Cable Provisioning Process Watchdog

The Prime Cable Provisioning process watchdog is an administrative agent that monitors the runtime health of all Prime Cable Provisioning processes. This process watchdog ensures that if a process stops unexpectedly, it is automatically restarted. One instance of the Prime Cable Provisioning process watchdog runs on every system that runs Prime Cable Provisioning components.

You can use the Prime Cable Provisioning process watchdog as a command-line tool to start, stop, restart, and determine the status of any monitored processes.

If a monitored application fails, it is restarted automatically. If, for any reason, the restart process also fails, the Prime Cable Provisioning process watchdog server waits a prescribed length of time before trying to restart.

Note You do not have to use the Prime Cable Provisioning process watchdog and the SNMP agent to monitor the extensions that are installed on Cisco Prime Network Registrar.

The period between restart attempts starts at 1 second and increases exponentially with every subsequent attempt until it reaches a length of 5 minutes. After that, the process restart is attempted at 5-minute intervals until successful. Five minutes after a successful restart, the period is automatically reset to 1 second again.

For example:

1. Process A fails.
2. The Prime Cable Provisioning process watchdog server attempts to restart it and the first restart fails.
3. The Prime Cable Provisioning process watchdog server waits 2 seconds and attempts to restart the process and the second restart fails.
4. The Prime Cable Provisioning process watchdog server waits 4 seconds and attempts to restart the process and the third restart fails.
5. The Prime Cable Provisioning process watchdog server waits 16 seconds and attempts to restart the process.

Using Prime Cable Provisioning Process Watchdog from CLI

The Prime Cable Provisioning process watchdog automatically starts whenever the system boots up. Consequently, this watchdog also starts those Prime Cable Provisioning system components installed on the same system. You can control the Prime Cable Provisioning watchdog through a simple command-line utility by running the **/etc/init.d/bprAgent** command.

Table 23-1 describes the command-line interface (CLI) commands available for use with the Prime Cable Provisioning process watchdog.

Table 23-1 Prime Cable Provisioning CLI Commands

Command	Description
bprAgent start	Starts the Prime Cable Provisioning process watchdog, including all monitored processes.
bprAgent stop	Stops the Prime Cable Provisioning process watchdog, including all monitored processes.
bprAgent restart	Restarts the Prime Cable Provisioning process watchdog, including all monitored processes.
bprAgent status	Gets the status of the Prime Cable Provisioning process watchdog, including all monitored processes.
bprAgent start *process-name*	Starts one particular monitored process. The value *process-name* identifies that process.
bprAgent stop *process-name*	Stops one particular monitored process. The value *process-name* identifies that process.
bprAgent restart *process-name*	Restarts one particular monitored process. The value *process-name* identifies that process.
bprAgent status *process-name*	Gets the status of one particular monitored process. The value *process-name* identifies that process. The *process-name* mentioned in this table can be: rdu, pws, dpe, kdc, snmpAgent, tomcat or cli(for DPE CII).

When the operating system (Solaris and Linux) is rebooted, the Prime Cable Provisioning process watchdog is first stopped, allowing Prime Cable Provisioning servers to shut down properly. To shut down or reboot the operating system gracefully, use the **init 6** command.

The **reboot** command does not execute application shutdown hooks and kills Prime Cable Provisioning processes rather than shutting them down. While this action is not harmful to Prime Cable Provisioning, it may delay server start-up and skew certain statistics and performance counters.

The events that trigger an action in the Prime Cable Provisioning watchdog daemon, including process crashes and restarts, are logged in a log file, *BPR_DATA/agent/logs/agent.log*. The watchdog daemon also logs important events to syslog under the standard `local6` facility.

CHAPTER 24

Alert and Error Messages

This chapter identifies all alert and error messages that Prime Cable Provisioning generates, specifically:

- Regional Distribution Unit Alerts, page 24-2
- Device Provisioning Engines Alerts, page 24-3
- Watchdog Alerts, page 24-4
- Network Prime Registrar Extension Point Alerts, page 24-6

Prime Cable Provisioning generates alerts through the Syslog service. Syslog is a client-server protocol that manages the logging of information. Prime Cable Provisioning syslog alerts are not a logging service; they provide a notification that a problem exists, but do not necessarily define the specific cause of the problem. You might find this information in the appropriate Prime Cable Provisioning log files.

Message Format

When Prime Cable Provisioning generates an alert message, the format is:

XXX-#-####: Message

- *XXX*—Identifies the facility code, which can include:
  - RDU (Regional Distribution Unit)
  - DPE (Device Provisioning Engine)
  - AGENT (rduSnmpAgent or dpeSnmpAgent)
  - NR_EP (Cisco Prime Network Registrar extension points)
  - KDC (Key Distribution Center)
- *#*—Identifies the severity level in use. Table 24-1 describes the different levels.

Table 24-1 Severity Levels for Alert Messages

Severity Level	Description
1	Identifies an alert
2	Identifies a critical alert
3	Identifies an error
6	Identifies an informational message

- *###*—Identifies the numeric error code.
- *Message*—Provides the alert text or message.

Regional Distribution Unit Alerts

Table 24-2 identifies the RDU alerts.

Table 24-2 RDU Alerts

Alert	Description
RDU-1-101: RDU ran out of disk space	Indicates that the storage partition of the RDU server ran out of space. After encountering this error, the RDU attempts to restart automatically, but will typically encounter the same error again until more storage space is available. You can remove or compress some of the log files. See Chapter 28, "Prime Cable Provisioning Support Tools" for additional information.
RDU-1-103: RDU ran out of memory	Indicates that the RDU ran out of memory. After encountering this error, the RDU server restarts automatically.
RDU-1-111: Evaluation key for technology *[technology_name]* expired	Indicates that an evaluation key for the technology specified expired. You must contact Cisco sales or TAC for a new license key.
RDU-1-115: You have used *[]* percent of available *[technology_name]* licenses.	Identifies, in percentage, the quantity of licenses used out of the total number of allowable licenses. Appears when you reach 80 percent of the license capacity.
RDU-1-122: DNS took *[]* seconds for lookup of address *[ip/hostname]*. Check DNS configuration and health of servers	Indicates that Prime Cable Provisioning performance may be slow due to delayed response from the DNS. The alert is generated whenever IP address lookup takes more than 60 seconds.
RDU-2-119: Directory *[]* that contains the RDU database has a filesystem block size of *[]* bytes that does not match the required size of *[]* bytes. Corruption may occur.	Indicates that the Prime Cable Provisioning database may not be reliable because the file system that contains the database files is not configured to support an 8-KB or greater block size. For details on configuring the file-system block size, see the *Cisco Prime Cable Provisioning 5.0 Quick Start Guide*.

Table 24-2 RDU Alerts (continued)

Alert	Description
RDU-2-200: Directory *[]* that contains the RDU database transaction logs has a filesystem block size of *[]* bytes that does not match the required size of *[]* bytes. Corruption may occur.	Indicates that the Prime Cable Provisioning database may not be reliable because the file system that contains the database log files is not configured to support an 8-KB or greater block size. For details on configuring the file system block size, see the *Cisco Prime Cable Provisioning 5.0 Quick Start Guide*.

Note Whenever an RDU syslog alert is sent, additional details (if any) can be found in the log file *BPR_DATA/rdu/logs/rdu.log*.

Device Provisioning Engines Alerts

Whenever a DPE syslog alert is sent, you can find additional details in the DPE logs.

You can use the **show log** command to access the DPE logs. For additional information, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.

Some DPE errors are also propagated to the RDU server log files. You can find these in the *BPR_DATA/rdu/logs/rdu.log* file.

Table 24-3 identifies the DPE alerts.

Table 24-3 DPE Alerts

Alert	Description
DPE-1-102: DPE ran out of disk space	The storage partition that the DPE server uses ran out of space. You have three options: **a.** Clear out any excess support bundles that may reside on the disk. You can do this by moving those support bundles to another machine and then running the **clear bundles** command from the DPE command-line interface (CLI). **b.** Run the **clear logs** command from the DPE CLI to clear more disk space. **c.** As a last resort, run the **clear cache** command from the DPE CLI to remove any cache files and force the DPE to resynchronize with the RDU server.
DPE-1-104: DPE ran out of memory	The DPE process ran out of memory. After encountering this error condition, the DPE restarts automatically. Determine how many device configurations are on the DPE; the larger the number of device configurations, the more memory is used. To reduce the number of device configurations, limit the number of devices in the provisioning groups, either primary or secondary, that the DPE serves.

Table 24-3 DPE Alerts (continued)

Alert	Description
DPE-1-109: Failed to connect to RDU	The RDU cannot be contacted. You must: **a.** Verify that the DPE network is configured and connected correctly. **b.** Check that the DPE is configured to connect to the proper RDU, and that the connecting port is configured properly by using the **dpe rdu-server** command. **c.** Check that the RDU process is running on the correct server and listening on the correct port. The DPE attempts to reconnect to the RDU process every few seconds until a connection is established.
DPE-1-117: DPE license nodes have been exceeded or there is no valid DPE license	Indicates that the Prime Cable Provisioning process watchdog, which starts the DPE, did not detect a license for the DPE. Enter the license key for the DPE using the administrator user interface. If you do not have a license, contact your Cisco representative.
DPE-1-116: DPE evaluation license has expired. Dropping DPE connections and deleting DPEs from database	Indicates that an evaluation license key for the DPE expired. You must contact Cisco sales or TAC for a new license key.
DPE-2-118: Directory *[]* that contains the DPE's cache has a filesystem block size of *[]* bytes that does not match the required size of *[]* bytes. Corruption may occur.	Indicates that the DPE cache may not be reliable because the file system is not configured to support an 8-KB or greater block size. For details on configuring the file system block size, see the *Cisco Prime Cable Provisioning 5.0 Quick Start Guide*.
DPE-1-121: Cannot start the server due to an invalid encryption key.	Indicates that the DPE could not be started because of an invalid encryption key.

Watchdog Alerts

Whenever the process watchdog sends a syslog alert, you can find error details (if any) in the *BPR_DATA/agent/logs/agent_console.log* file and the log files corresponding to the specific component mentioned in the alert (if any). For example, if you receive an alert similar to *The rdu unexpectedly terminated*, you would check the RDU server log file (*BPR_DATA/rdu/logs/rdu.log*) for additional information. Table 24-4 identifies the process watchdog alerts.

Table 24-4 Process Watchdog Alerts

Alert	Description
AGENT-3-9001: Failed to start the *[component]*	Indicates that the watchdog has failed to start the specified component.
AGENT-3-9002: The *[component]* unexpectedly terminated	Indicates that the specified component, monitored by the process watchdog, has unexpectedly failed.

Table 24-4 Process Watchdog Alerts (continued)

Alert	Description
AGENT-6-9004: The *[component]* has started	Generated any time a component is successfully started by the process watchdog. This message is for informational purposes only.
AGENT-6-9005: The *[component]* has stopped	Generated any time a component is successfully stopped through the process watchdog. This message is for informational purposes only.
AGENT-3-9003: Failed to stop the *[component]*	Indicates that a component did not stop when the process watchdog attempted to stop it.
AGENT-3-9003: Failed to create listener thread; *[error no]*Failed to close listen socket; *[error no]* Failed to cancel listen thread, and so on	Indicates errors that are not defined in other alert messages.

The *[component]* variable presented in the process watchdog alerts list shown in Table 24-4 represents any of these component values:

- rdu
- pws
- dpe
- tomcat
- cli
- snmpAgent
- kdc

Network Prime Registrar Extension Point Alerts

Whenever a Prime Cable Provisioning Network Registrar extension point syslog alert is sent, you can find additional details in the Network Registrar log file.

Table 24-5 identifies the process watchdog alerts.

Table 24-5 Network Registrar Extension Alerts

Alert	Description
NR_EP-1-106: Failed to connect to RDU	The Network Registrar server cannot connect to the RDU. You should verify that the RDU process is running and, if it is not already running, start the RDU. If the RDU is running, use the Network Registrar computer to ping the RDU. If you are unable to ping the RDU, fix the routing tables or other communication parameters, between the two devices. If this alert is frequently repeated, you may have an unstable connection between the two hosts. Use generally accepted network troubleshooting techniques to improve the connectivity between the two hosts.
NR_EP-1-107: Failed to connect to any DPEs	The Network Registrar extension cannot connect to the DPEs. Check that there are DPEs in the provisioning group for each Network Registrar extension. If not, change the Network Registrar provisioning group to one that has DPEs available. If DPEs are in the provisioning group, ensure that the Network Registrar extension has registered with the RDU; if it has not, it will not recognize any of the DPEs. If, after completing the check, the alert continues, check that there is network connectivity between the Network Registrar extension and the DPEs in the provisioning group. If this alert is frequently repeated, you may have an unstable connection between the two hosts. Use generally accepted network troubleshooting techniques to improve the connectivity between the two hosts.
NR_EP-6-108: The Prime Cable Provisioning NR extensions have started	The Network Registrar extensions have been started.
NR_EP-6-109: The Prime Cable Provisioning NR extensions have stopped	The Network Registrar extensions have been stopped.
NR_EP-6-110: Registered with RDU *[address and port]*	The Network Registrar extensions have been registered with the RDU. The *address and port* identifies the address of the RDU that has registered the Network Registrar extensions.
NR_EP-1-111: Failed to find usable (best) DPEs	The Network Registrar extensions are unable to find a usable DPE.

Monitoring Component Logs

This chapter describes how you can monitor the Prime Cable Provisioning components.

Logging of events is performed at component level, and in some unique situations, DPE events are additionally logged at the RDU to give them higher visibility.

Log files are stored in their own log directories and can be examined by using any text processor. You can compress the files for easier e-mailing to the Cisco Technical Assistance Center or system integrators for troubleshooting and fault resolution.

You can also access the RDU and the DPE logs from the Admin UI.

Log Levels and Structures

The log file structure, illustrated in Example 25-1, includes:

- Domain Name—This is the name of the computer generating the log files.
- Date and Time—This is the date on which a message is logged. This information also identifies the applicable time zone.
- Facility—This identifies the system, which (in this case) is Prime Cable Provisioning.
- Sub-facility—This identifies the Prime Cable Provisioning subsystem or component.
- Severity Level—The logging system defines seven levels of severity (as described in Table 25-1) that are used to identify the urgency with which you might want to address log issues. The process of configuring these severity levels is described in Command DefaultConfiguring Severity Levels, page 25-3.

Table 25-1 Severity Levels

Log Level	Description
0-Emergency	System unstable. Sets the logging function to save all emergency messages.
1-Alert	Immediate action needed. Sets the logging function to save all activities that need immediate action and those of a more severe nature.
2-Critical	Critical conditions exist. Sets the logging function to save all error messages and those of a more severe nature.
3-Error	Error conditions exist. Sets the logging function to save all error messages and those of a more severe nature.

Table 25-1 Severity Levels (continued)

Log Level	Description
4-Warning	Warning conditions exist. Sets the logging function to save all warning messages and those of a more severe nature.
5-Notification	A normal, but significant, condition exists. Sets the logging function to save all notification messages and those of a more severe nature.
6-Information	Informational messages. Sets the logging function to save all logging messages available.

Note Another level known as 7-Debug is used exclusively by Cisco for debugging purposes. Do not use this level except at the direction of the Cisco Technical Assistance Center.

- Msg ID—This is a unique identifier for the message text.
- Message—This is the actual log message.

Example 25-1 Sample Log File

Domain Name	Data and Time	Facility	Sub-facility	Severity Level	Msg ID	Message
bac.example.com:	2013-02-08 22:43:23,341 EST:	%CPCP-	RDU-	5	0236:	Prime Cable Provisioning Regional Distribution Unit starting up
bac.example.com:	2013-02-08 22:43:23,711 EST:	%CPCP-	RDU-	5	0566:	API defaults
bac.example.com:	2013-02-08 22:43:25,211 EST:	%CPCP-	RDU-	5	0567:	Network Registrar defaults
bac.example.com:	2013-02-08 22:43:25,321 EST:	%CPCP-	RDU-	5	0568:	Server defaults
bac.example.com:	2013-02-08 22:43:26,721 EST:	%CPCP-	RDU-	5	0570:	DOCSIS defaults
bac.example.com:	2013-02-08 22:43:26,911 EST:	%CPCP-	RDU-	5	0571:	Computer defaults
bac.example.com:	2013-02-08 22:43:27,221 EST:	%CPCP-	RDU-	5	1018:	CableHome WAN-MAN defaults
bac.example.com:	2013-02-08 22:43:27,321 EST:	%CPCP-	RDU-	5	1019:	CableHome WAN-Data defaults
bac.example.com:	2013-02-08 22:43:27,711 EST:	%CPCP-	RDU-	5	0707:	PacketCable defaults
bac.example.com:	2013-02-08 22:43:28,561 EST:	%CPCP-	RDU-	5	0569:	Created default admin user
bac.example.com:	2013-02-08 22:43:28,711 EST:	%CPCP-	RDU-	5	0575:	Database initialization completed in [471] msec

Command Default Configuring Severity Levels

You can configure the severity levels of logging for all components to suit your specific requirements. For example, the severity level for the RDU could be set to Warning, and the level for the DPE could be set to Alert.

Log messages are written based on certain events taking place. Whenever an event takes place, the appropriate log message and severity level are assigned and, if that level is less than or equal to the configured level, the message is written to the log. The message is not written to the log if the level is higher than the configured value.

For example, assume that the log level is set to 4-Warning. All events generating messages with a log level of 4 or less are written into the log file. If the log level is set to 6-Information, the log file will receive all messages. Consequently, configuring a higher log level results in a larger log file size.

Note The KDC is not considered in this log file.

To configure the severity level on the DPE, use the **log level** command from the DPE command line. For detailed information, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.

To configure the log level tool on the RDU, see Using the RDU Log Level Tool, page 25-5.

Rotating Log Files

All log files are numbered and rolled over based on a configured maximum file size. The default maximum file size is 25 MB. (To configure the maximum file size from the application programming interface [API], use the *ServerDefaultsKeys.SERVER_LOG_MAXSIZE* property.) Once a log file touches the configured limit, the data is rolled over to another file. This file is renamed in the *XXX.N*.log format, where:

- *XXX*—Specifies the name of the log file.
- *N*—Specifies any value between 1 and 200.

Note The RDU and DPE servers store up to 200 log files at a given time. For a list of log files in these servers, see subsequent sections.

For example, once *rdu.log* reaches the 25-MB limit, it is renamed as *rdu.1.log*. With every 25-MB increase in file size, the latest file is renamed as *rdu.2.log*, *rdu.3.log*, and so on. So, the *rdu.4.log* file will contain data more recent than *rdu.7.log*. The latest log information, however, is always stored in *rdu.log*.

Note For rdu_auth.log, the default maximum size is 10MB and the RDU server stores up to 100 log files at any given point of time. However, these values can be configured using the file log4j.xml located at the directory BPR_HOME/rdu/conf. You must restart the RDU for every change made to the file.

Regional Distribution Unit Logs

The RDU has two logs that it maintains in the *BPR_DATA/rdu/logs* directory:

- *rdu.log*—Records RDU processing according to the configured default severity level. (For instructions on setting the default log levels, see Setting the RDU Log Level, page 25-6.)
- *audit.log*—Records high-level changes to the Prime Cable Provisioning configuration or functionality including the user who made the change.
- *rdu_auth.log*—When a user tries to authenticate itself to RDU, authentication related information gets captured in this log.

When you enable logging of informational messages (6-Information), the RDU logs additional messages that expose batch-processing operations. These messages also contain information on elapsed time and rate.

Viewing the *rdu.log* File

You can use any text processor to view the *rdu.log* file. In addition, you can view the log file from the Admin UI.

To view the file:

Step 1 Choose the Regional Distribution Unit tab under **Servers**.

The View Regional Distribution Unit Details page appears.

Step 2 Click the View Details icon corresponding to RDU Log File.

The View Log File Contents page appears, displaying data from *rdu.log*.

Viewing the *audit.log* File

You can use any text processor to view the *audit.log* file. In addition, you can view the log file from the Admin UI.

To view the file:

Step 1 Choose the Regional Distribution Unit tab under **Servers**.

The View Regional Distribution Unit Details page appears.

Step 2 Click the View Details icon corresponding to Audit Log File.

The View Log File Contents page appears, displaying data from *audit.log*.

Viewing the *rdu_auth.log* File

You can use any text processor to view the *rdu_auth.log* file. You cannot view this log file from the Admin UI.

Setting the Log Level for *rdu_auth.log*

The logging level for rdu_auth.log can be set using the file log4j.xml located in the directory BPR_HOME/rdu/conf. By default, only informational messages are logged. You must restart the RDU, after making any changes in the log level.

Using the RDU Log Level Tool

Use the RDU log level tool to change the current log level of the RDU from the command line, using the **setLogLevel.sh** command. This tool resides in the *BPR_HOME/rdu/bin* directory.

Table 25-2 identifies the available severity levels and the types of messages written to the log file when enabled.

Table 25-2 RDU Logging Levels

Log Level	Description
0-Emergency	System unstable. Sets the logging function to save all emergency messages.
1-Alert	Immediate action needed. Sets the logging function to save all activities that need immediate action and those of a more severe nature.
2-Critical	Critical conditions exist. Sets the logging function to save all error messages and those of a more severe nature
3-Error	Error conditions exist. Sets the logging function to save all error messages and those of a more severe nature.
4-Warning	Warning conditions exist. Sets the logging function to save all warning messages and those of a more severe nature.
5-Notification	A normal, but significant, condition exists. Sets the logging function to save all notification messages and those of a more severe nature.
6-Information	Informational messages. Sets the logging function to save all logging messages available.
Note	Another level known as 7-Debug is used exclusively by Cisco for debugging purposes. Do not use this level except at the direction of the Cisco TAC.

We recommend that you keep the RDU severity level at the Warning level to help maintain a steady operations state. The Information level is recommended to be used with caution if you need to maintain steady state performance during debug operations. You should exercise caution when running with the Information level because this creates a great number of log entries, which in itself can adversely impact performance.

Note The RDU process has to be up to execute the log level tool. Also, you must be a privileged user to run this tool by using the **setLogLevel.sh** command.

Syntax Description

setLogLevel.sh -[*0..6*] [**-help**] [**-show**] [**-default**] [**-debug**]

- **-**[*0..6*]—Identifies the severity level to be used. For a list of available levels, see Table 25-2.
- **-help**—Displays help for the tool.
- **-show**—Displays the current severity level set for the RDU server.
- **-default**—Sets the RDU to the installation default level 5 (notification).
- **-debug**— Sets an interactive mode to enable or disable tracing categories for the RDU server.

Note You should only enable the debug settings that the Cisco support staff recommends.

You can also use this tool to perform these functions:

- Setting the RDU Log Level, page 25-6
- Viewing the Current Log Level of RDU, page 25-7

Setting the RDU Log Level

You can use this tool to change the logging level from one value to another value. The following example illustrates how to set the RDU logging level to the warning level, as indicated by the number 4 in the **setLogLevel.sh** command. The actual log level set is not important for the procedure; it can be interchanged as required.

The example described in this section assumes that the RDU server is up, the username for the RDU is admin, and the password is changeme**.**

To set the RDU logging level:

Step 1 Change directory to *BPR_HOME/rdu/bin*.

Step 2 Run the RDU log level tool using this command:

```
# setLogLevel.sh 4
```

This prompt appears:

```
Please type RDU username:
```

Step 3 Enter the RDU username. In this example, the default username (**admin**) is used.

```
Please type RDU username: admin
```

This prompt appears:

```
Please type RDU password:
```

Step 4 Enter the RDU password for the RDU. In this example, the default password (**changeme**) is used.

```
Please type RDU password: changeme
```

This message appears to notify you that the log level has been changed. In this example, the level was 5, for notification, and is now 4, for warning.

```
RDU Log level was changed from 5 (notification) to 4 (warning).
```

Viewing the Current Log Level of RDU

You can use this tool to view the RDU log and determine which logging level is configured before attempting to change the value.

The example described in this section assumes that the:

- RDU server is up.
- Username for the RDU is **admin.**
- Password is **changeme.**

To view the current logging level of the RDU:

Step 1 Change directory to *BPR_HOME/rdu/bin*.

Step 2 Run this command:

```
# setLogLevel.sh -show
```

This prompt appears:

```
Please type RDU username:
```

Step 3 Enter the RDU username (**admin**) and press **Enter**.

```
Please type RDU username: admin
```

This prompt appears:

```
Please type RDU password:
```

Step 4 Enter the RDU password (**changeme**) and press **Enter**.

```
Please type RDU password: changeme
```

This message appears:

```
The logging is currently set at level: 4 (warning)
All tracing is currently disabled.
```

Provisioning Web Services Log

The PWS maintains a pws.log file in the *BPR_DATA/pws/logs* directory. The file contains log information of the PWS component. The other log file called pws_console.log is the Tomcat console log and is located at *BPR_DATA/agent/logs* directory. By default, the PWS log level is set to INFO.

Using the PWS Log Level Tool in CLI

Use the PWS log level tool (**ws-cli.sh**) to change the current log level of the PWS. This tool resides in the *BPR_HOME/pws/bin* directory.

Table 25-3 identifies the available severity levels and the types of messages written to the pws.log file. These log levels can be altered during run time.

Table 25-3 PWS Logging Levels

Log Level	Description
ERROR	Records all PWS error messages.
WARN	Records all PWS warning messages.
INFO	Records information regarding PWS operations.
DEBUG	Records debug information that helps you to dress any error.

It is recommended that you set the PWS log level to the default INFO level. Using ws-cli.sh, you can change the log level to DEBUG if you are planning to troubleshoot the system. You should exercise caution when running with the DEBUG level because this creates a great number of log entries, which in itself can adversely impact performance.

Note Loggers are applicable only at runtime and are set to the default values every time you restart the PWS. To retain the runtime log details, use the command -sap that will save the modifications being made to the logger. These changes will not change even after you restart the PWS.

Syntax Description

ws-cli.sh

- **-ll**—Lists all the loggers and it's severity level. You can also use **--listlog.**
- **-sl**—Sets the log level. You can also use **--setlog**.

You can also use this tool to perform these functions:

- Setting the PWS Log Level, page 25-9
- Viewing the Current Log Level of PWS, page 25-9

PWS Loggers

For PWS, Prime Cable Provisioning provides loggers for every operation type. The different logger names are:

- api_group—Provides log details about groups.
- api_device_type—Provides log details about device type.
- api_file—Provides log details about files.
- api_cos—Provides log details about class of service.
- api_device—Provides log details about devices.
- api_search—Provides log details about search operation.
- api_dhcpcriteria—Provides log details about DHCP criteria.
- api_session—Provides log details about sessions.
- root—Can be set to provides generic log details.

Setting the PWS Log Level

You can use this tool to change the logging level from one value to another value. The following example illustrates how to set the PWS logging level to the DEBUG level, as indicated by the number 4 in the **ws-cli.sh** command. The actual log level is not important for the procedure; it can be interchanged as required. The example described in this section assumes that the PWS server is up**.**

To set the PWS logging level:

Step 1 Change directory to *BPR_HOME/pws/bin*.

Step 2 Run the PWS log level tool using this command. This will change the log level from INFO to DEBUG.

```
# ws-cli.sh -sl <root=DEBUG>
```

Step 3 Save the changes by using the command:

```
# ws-cli.sh -sap
```

Viewing the Current Log Level of PWS

You can use this tool to view the PWS log and determine which logging level is configured before attempting to change the value. The example described in this section assumes that the PWS server is up.

To view the current logging level of the PWS:

Step 1 Change directory to *BPR_HOME/pws/bin*.

Step 2 Run this command:

```
# ws-cli.sh -ll <logger>
```

In case you have not included logger in the CLI command, it lists all the loggers and it's log levels.

Device Provisioning Engines Log

The DPE maintains a *dpe.log* file in the *BPR_DATA/dpe/logs* directory. The file contains records of all events having the configured default level. In situations where the DPE undergoes catastrophic failure, such as engaging in a series of system crashes, the catastrophic errors are also logged into the *rdu.log* file.

The *SNMPService.logyyy.log* log file is used by the DPE, when PacketCable is enabled on the DPE server, to provide detailed debugging information. You use the **service packetcable** *1..1* **show snmp log** command from the DPE command-line interface (CLI) to view this file, which resides in the *BPR_DATA/dpe/logs* directory. For PacketCable command usage, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.

Note PacketCable logging messages are sent to the *dpe.log* file and the detailed SNMP debugging is sent to the *SNMPService.logyyy.log* file.

You can use any text viewer to view the *dpe.log* file. In addition, you can use the **show log** command from the DPE CLI. For additional information, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.

You can also view the DPE log file using the Prime Cable Provisioning Admin UI.

To view the file:

Step 1 Choose **Servers > Device Provisioning Engines**.

Step 2 Click the link of the DPE whose log file you want to view.

The View Device Provisioning Engines Details page appears.

Cisco Prime Network Registrar Logs

Prime Cable Provisioning generates log messages from Cisco Prime Network Registrar DHCP server extensions. The DHCP server log resides in the *cnr-install-path/name_dhcp_1_log* directory; *cnr-install-path* is a variable and is specific to the value that you enter. The default location for the DHCP server log file is */var/nwreg2/local/logs/name_dhcp_1_log*.

The log messages emitted via the DHCP server extensions are based on the extension trace level setting. You can set values (described in Table 25-4) at the trace level; the number you set makes that number the current setting of the **extension-trace-level** attribute for all extensions.

Table 25-4 DHCP Server Extension Trace Levels

Level	Description
0	Logs error and warning conditions. Sets the extensions to emit all error and warning messages and those of a more severe nature.
1	Logs server interactions, which include configuration instructions obtained from the DPE and configuration generation requests that are forwarded to the RDU.
2	Logs processing details, which include individual configuration commands and attribute values forwarded in instruction generation requests.
3	Logs internal processing for extensions debugging, which includes hexadecimal dumps of messages.
4	Logs debugging of extension background operations, which include polling of DPE status.

You can change the extension trace level by using the Network Registrar web UI. To change the level:

Step 1 Open the Network Registrar local web UI.

Step 2 From the menu, click **DHCP**, then **DHCP Server**.

Step 3 Click the Local DHCP Server link.

Step 4 On the Edit DHCP Server page, expand the Extensions attribute category.

Step 5 Set the **extension-trace-level** value, then click **Modify Server**.

Step 6 Reload the DHCP server.

Note For detailed information on logging performed by the DHCP server, see the *Cisco Prime Network Registrar 8.1 User Guide*.

Admin UI Log

The Admin UI maintains a admiui.log file in the BPR_DATA/adminui/logs directory where all the Admin UI related logs are stored. The other log file called tomcat_console.log is the Admin UI Tomcat console log and is located at BPR_DATA/agent/logs directory. By default, the Admin UI log is set to the INFO level.

Using the Admin UI Log Level Tool

Table 25-5 identifies the available severity levels and the types of messages written to the adminui.log file.

Table 25-5 Admin UI Logging Levels

Log Level	Description
ERROR	Records all Admin UI error messages.
WARN	Records all Admin UI warning messages.
INFO	Records information regarding Admin UI operations.
DEBUG	Records debug information that helps you to dress any error.
TRACE	Records details about any server traces.

These log levels can be altered but Admin UI(Tomcat) needs to be restarted for the log level to take effect. It is recommended that you set the Admin UI log severity level to the INFO level. You can set the log level to DEBUG if you are planning to troubleshoot the system. You should exercise caution when running with the DEBUG level because this creates a great number of log entries, which in itself can adversely impact performance.

Setting the Admin UI Log Level

By default, the Admin UI log level is set to INFO. To change the log level, you need to edit the file logback located at: */opt/CSCObac/rdu/tomcat/webapps/adminui/WEB-INF/classes*. The file contains multiple loggers and to change the log level to say DEBUG, edit the file using a text editor as following:

```
logger name="com.cisco" level="DEBUG" additivity="false”
```

You must restart the tomcat server for the new log level to take effect.

PART 5

Troubleshooting Prime Cable Provisioning

This part contains the following sections:

- Troubleshooting Prime Cable Provisioning
- Frequently Asked Questions
- Prime Cable Provisioning Support Tools

CHAPTER 26

Troubleshooting Prime Cable Provisioning

This chapter provides details on how to troubleshoot with Cisco Prime Cable Provisioning. This chapter describes:

- Troubleshooting Checklist, page 26-1
- Troubleshooting Devices by Device ID, page 26-2
- Troubleshooting Using Diagnostics Tool, page 26-5
- Bundling Server State for Support, page 26-10
- Troubleshooting DOCSIS Networks, page 26-11
- Troubleshooting PacketCable eMTA Provisioning, page 26-11

For a list of FAQs related to Prime Cable Provisioning provisioning, see Chapter 27, "Frequently Asked Questions."

You can also refer to Technical Notes posted at the Support Site for customer specific issues and solutions.

Troubleshooting Checklist

While troubleshooting with Prime Cable Provisioning, use the checklist described in Table 26-1.

Table 26-1 Troubleshooting Checklist

Procedure	Refer to...	Check Off
1. Check if the Prime Cable Provisioning processes are up on all systems on which Prime Cable Provisioning components are installed.	Using Prime Cable Provisioning Process Watchdog from CLI, page 23-2	□
2. Check the Prime Cable Provisioning component logs for indications of high-severity errors. These include the information logged for:		□
– RDU	Regional Distribution Unit Logs, page 25-4	
– DPE	Device Provisioning Engines Log, page 25-9	

Table 26-1 Troubleshooting Checklist (continued)

Procedure	Refer to...	Check Off
3. View server uptime from the administrator user interface to confirm that the servers are not bouncing.	Monitoring Servers Using Admin UI, page 21-1	❑
4. View the RDU and DPE service performance statistics from the administrator user interface. Observe any abnormal numbers, such as extended transaction times.	Monitoring Servers Using Admin UI, page 21-1	❑
5. Check the syslog alerts log.	Chapter 24, "Alert and Error Messages"	❑
6. Check the operating system and hardware resources, such as: – Disk space – CPU time – Memory	Solaris documentation for specific commands.	❑
7. If troubleshooting a specific device, view the device instructions that are cached at the DPE.	The **show device-config** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.	❑
8. Configure individual device troubleshooting from the administrator user interface and, after a period of time, inspect the troubleshooting log.	Configuring Devices for Troubleshooting, page 26-3	❑
9. Configure a higher level of logging on the RDU or the appropriate DPE for detailed logging information.	Using the RDU Log Level Tool, page 25-5 The **log level** command described in the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.	❑

Troubleshooting Devices by Device ID

You can use this feature to collect detailed diagnostics about one or more specific devices. Troubleshooting information includes all server interactions related to a given device or a group of devices. This information includes administrator user interface operations, RDU application programming interface (API) operations, DPE interactions with devices, and interserver DPE-to-RDU interactions.

You can enable or disable diagnostics via group management for one or more specific devices without turning logging on, and without searching through log files for specific device information.

Prime Cable Provisioning maintains a list of devices, based on device identifiers (MAC addresses and DUIDs), for which detailed diagnostics are collected. Troubleshooting information is stored centrally at the RDU and is maintained on a per-device basis. Neither DPEs nor Cisco Prime Network Registrar extensions store this data. Rather, they forward this information to the RDU, which, upon receiving information, writes it to the *troubleshooting.log* file in the *BPR_DATA/rdu/logs* directory.

The *troubleshooting.log* file is different from other log files such as *rdu.log*, *dpe.log*, and *audit.log*. It only logs detailed troubleshooting information relating to a specific set of devices in the diagnostics mode.

If the connection from the DPE or Network Registrar extension to the RDU is lost, any new troubleshooting events occurring on the DPE or Network Registrar extension are discarded. The logging of troubleshooting information resumes only after the connection to the RDU is restored.

The DPE maps MAC addresses and DUIDs of a specific device being diagnosed to the IP address for that device. The DPE receives IP updates from the Network Registrar extensions for the devices being diagnosed.

Any modifications to the device tracking list, such as the addition of a new device or a group, take place immediately at all servers; you do not have to reboot the RDU or the DPE. The log files on the respective servers list the current list of devices in the diagnostics mode.

⚠

Caution Additional memory and disk space is required whenever the device troubleshooting feature is used. As the number of tracked devices increases, so does the amount of memory and disk space that is required to support the number of logs that are created.

Configuring Devices for Troubleshooting

Device diagnostics is disabled until one or more devices are set in diagnostics mode.

To enable diagnostics for a device, the device must be preregistered in the Prime Cable Provisioning RDU. If the device is not yet preregistered, add the device from the Manage Devices page by clicking the Add button. For information on adding devices, see Adding Device Records, page 18-9.

You can configure a maximum number of devices in diagnostics mode to prevent inadvertently putting too many devices in this mode and thus diminishing server performance. By default, the maximum number of devices allowed is 25. To configure the maximum number of devices from the Admin UI, click the Systems Defaults page via the **Configuration > Defaults**. Enter a value in the Maximum Diagnostics Device Count field.

Relating a Device to a Group

You can troubleshoot a device by relating it to a specific group. Use the Relate function to associate a device, using its MAC address or its DUID, to a specific group, which is in turn associated with a specific group type. (See Relating and Unrelating Devices, page 18-11.) Doing so records an extraordinarily large volume of information for a device; you can then use the information to troubleshoot potential problems.

Table 26-2 identifies a possible workflow using the Relate and Unrelate functions.

Table 26-2 Sample Relate/Unrelate Process

Step	Action
1.	Determine whether or not a problem exists and identify which devices are affected.
2.	Relate the devices to a group.
3.	Wait a few minutes to ensure that device traffic is passing, or perform a hard boot of the device.

Table 26-2 Sample Relate/Unrelate Process (continued)

Step	Action
4.	Open the *BPR_DATA/rdu/logs/troubleshooting.log* file in a word processing application and locate the entries for the MAC address or the DUID of the specific device.
5.	Identify, correct, test, and verify the problem.
6.	Unrelate the device from the group.

Viewing List of Devices in Diagnostics Mode

When you enable troubleshooting for a device, the device is automatically added to a special device group that contains a list of devices in troubleshooting mode. The group type is **system** and the group name is **system-diagnostics**. You can access the list of devices in this group from the API or the administrator user interface.

To view a list of devices currently enabled for diagnostics:

Step 1 From the Manage Devices page, click the Search Type drop-down list and select Group Search.

Step 2 From the Group Name (Group Type) drop-down list, select the **system-diagnostics (system)** option to view all the devices in diagnostics mode.

Step 3 Click **Search**.

Note An alternative way to view the list of devices in diagnostics mode is to consult the RDU log (*rdu.log*) and the DPE log (*dpe.log*) files. The list of devices is logged whenever the server is started and whenever there is a change in the list of devices enabled for diagnostics.

The devices enabled for diagnostics appear in the log files with the log level of 5-notification. For details on log files, see Chapter 25, "Monitoring Component Logs."

Examples

This example features log output while troubleshooting an MTA:

```
bac-test.example.com:2005 03 04 18:38:24 EST:%BAC-DIAGNOSTICS-3-4055:[##MTA-9a Unconfirmed
FQDN Request Received from [/10.10.10.5 ['kdcquery']. Client with IP Address [10.10.20.2]
and MAC Address [1,6,00:00 :ca:b7:7e:91]]]
bac-test.example.com:2005 03 04 18:38:24 EST:%BAC-DIAGNOSTICS-3-4082:[Results of BACC
Lookup. FQDN: [1-6-00-00-ca-b7-7e-91.example.com MAC: 1,6,00:00:ca:b7:7e:91. Client with
IP Address [10.10.20.2] and MAC Address [1,6,00:00:ca:b7:7e:91]]]
bac-test.example.com:2005 03 04 18:38:24 EST:%BAC-DIAGNOSTICS-3-4070:[##MTA-9b FQDN Reply
Sent to [/10.10.20.2(41142) for MTA 1,6,00:00:ca:b7:7e:91. Client with IP Address
[10.10.20.2] and MAC Address [1,6, 00:00:ca:b7:7e:91]]]
bac-test.example.com:2005 03 04 18:38:26 EST:%BAC-DIAGNOSTICS-3-4132:[##MTA-13 Incoming
APREQ received from [/10.10.20.2:1293. Client with IP Address [10.10.20.2] and MAC
Address [1,6,00:00:ca:b7:7e:91]]]
bac-test.example.com:2005 03 04 18:38:26 EST:%BAC-DIAGNOSTICS-3-4141:[##MTA-13 APREP sent
to [/10.10.20.2(1293) For MTA 1,6,00:00:ca:b7:7e:91. Client with IP Address [10.10.20.2]
and MAC Address [1,6,00:00: ca:b7:7e:91]]]
bac-test.example.com:2005 03 04 18:38:26 EST:%BAC-DIAGNOSTICS-3-0764:[[##MTA-15 SNMPv3
INFORM Received From 10.10.20.2. Client with IP Address [10.10.20.2] and MAC Address
[1,6,00:00:ca:b7:7e:91]]]
```

```
bac-test.example.com:2005 03 04 18:38:26 EST:%BAC-DIAGNOSTICS-3-0764:[[##MTA-19 SNMPv3 SET
Sent to 10.10.20.2. Client with IP Address [10.10.20.2] and MAC Address
[1,6,00:00:ca:b7:7e:91]]]
bac-test.example.com:2005 03 04 18:38:26 EST:%BAC-DIAGNOSTICS-3-1092:[Received a TFTP
[read] request from [10.10.20.2:1271] for [bpr01060000cab77e910002]; Client with MAC
Address [1,6,00:00:ca:b7:7e:91] and IP Address [10.10.20.2]]
bac-test.example.com:2005 03 04 18:38:26 EST:%BAC-DIAGNOSTICS-3-1155:[##MTA-23 Finished
handling [read] request from [10.10.20.2:1271] for [bpr01060000cab77e910002]; Transferred
[236] bytes to Client with MAC Address [1,6,00:00:ca:b7:7e:91] and IP Address
[10.10.20.2]]
bac-test.example.com:2005 03 04 18:38:27 EST:%BAC-DIAGNOSTICS-3-0764:[[##MTA-25 SNMP
Provisioning State INFORM Received from 10.10.20.2. Client with IP Address [10.10.20.2]
and MAC Address [1,6,00:00:ca:b7:7e:91]]]
bac-test.example.com:2005 03 04 18:38:27 EST:%BAC-DIAGNOSTICS-3-0764:[[MTA Configuration
Confirmed, Returned 'pass' as the final MTA provisioning state for 10.10.20.2. Client with
IP Address [10.10.20.2] and MAC Address [1,6,00:00:ca:b7:7e:91]]]
```

Troubleshooting Using Diagnostics Tool

You can use the diagnostics tool to collect performance statistics—down to a specific type of statistic—for Prime Cable Provisioning servers. Using individual scripts for each task that this tool performs, you can:

- Gather diagnostics concurrently (**startDiagnostics.sh**)
- Stop diagnostics prematurely (**stopDiagnostics.sh**)
- Determine the status of diagnostics collection (**statusDiagnostics.sh**)

You can run the diagnostic tool concurrently when you face an issue and need additional data for troubleshooting, or you can set it to run periodically on a given schedule via a cron job.

Caution When using the diagnostics tool, ensure that sufficient space is available on your systems for storing the diagnostic data.

The diagnostic tool resides in the following locations for the:

- RDU—*BPR_HOME/rdu/diagnostics/bin*
- DPE—*BPR_HOME/dpe/diagnostics/bin*
- Cisco Prime Network Registrar—*BPR_HOME/cnr_ep/diagnostics/bin*

Note You can bundle the collected diagnostics using the **bundleState.sh** script. For details, see Bundling Server State for Support, page 26-10.

Using startDiagnostics.sh Tool

You can run the **startDiagnostics.sh** tool in two modes:

- Interactive—In this mode, you can select the diagnostic data that you require from a list of options.

- Noninteractive—In this mode, you first generate a response file that contains arguments. Then, when you run the **startDiagnostics.sh** script, the tool collects diagnostics data based on the arguments specified in the response file.

Syntax Description

startDiagnostics.sh [**-r** *response_file*] | [**-g** *response_file*] [**-help**]

- **startDiagnostics.sh**—Runs diagnostics in the interactive mode
- *response_file*—Identifies the response file
- **-r** *response_file*—Uses the response file generated to run the diagnostics tool in the noninteractive mode
- **-g** response_file—Generates the response file without running diagnostics
- **-help**—Displays help for the tool. You must use the **-help** option exclusively. Do not use this with any other option.

Running startDiagnostics.sh in Interactive Mode

When you enter **startDiagnostics.sh** without specifying any argument, the diagnostics tool runs in the interactive mode and prompts you to select the statistics that you want from the RDU, the DPE, and the Network Registrar servers.

Caution Ensure that you process statistics with caution because doing so could severely impact system performance.

Syntax Description

startDiagnostics.sh [**-help**]

- **startDiagnostics.sh**—Runs diagnostics in the interactive mode
- **-help**—Displays help for the tool. You must use the **-help** option exclusively. Do not use this with any other option.

Examples

```
# ./startDiagnostics.sh

Please enter directory where to put output files [] /var/CSCObac
   Please enter the duration of the diagnostic (sec) [600]

Please select statistics you would like to gather on RDU

   CPU statistics (y/n/q)? [y]
   Process statistics (y/n/q)? [n]
   IO statistics (y/n/q)? [y]
   Memory statistics (y/n/q)? [y]
   Network statistics (y/n/q)? [y]
   RDU API traffic (y/n/q)? [y]
   RDU CNR traffic (y/n/q)? [y]
   RDU DPE traffic (y/n/q)? [y]
   RDU CNR extension traffic (y/n/q)? [y]
   RDU SNMP traffic (y/n/q)? [y]
   System Configuration (y/n/q)? [y]
```

```
Enter addition argument for RDU API traffic
   Please enter RDU Server port [49187]

Enter addition arguments for RDU DPE traffic
   Enter DPE ip addr if you want to capture traffic by ip addr [] 10.10.29.1
   Enter DPE port number if you want to capture traffic by port number [] 49186

Enter addition arguments for RDU CNR_EX traffic
   Enter Ip addr if you want to capture traffic by Cnr Extension IP addr [] 10.10.85.2
   Enter port number if you want to capture traffic by Cnr Extension port []

You could run statusDiagnostics.sh to find out the status of the diagnostics.
You could run stopDiagnostics.sh to stop the diagnostics.
You could run bundleState.sh to bundle the output when diagnostics is complete.
```

Note If you do not enable statistics for the following options, the tool does not request values for additional arguments, as featured in the example:

- RDU-API traffic
- RDU-DPE traffic
- RDU-Network Registrar extension traffic

After you run the **startDiagnostics.sh** tool, output files for each statistic are created under the directory in which you run the tool. You can also bundle the output files and forward them to the Cisco Technical Assistance Center for support. To do so, enter **y** at the System Diagnostics Capture prompt.

For example:

```
System Configuration (y/n/q)? [y]
```

For more details on bundling server state, see Bundling Server State for Support, page 26-10.

Running startDiagnostics.sh in Noninteractive Mode

Before running the **startDiagnostics.sh** tool in the noninteractive mode for the first time, you must generate the response file. Thereafter, you only need to run a single command, which collects diagnostics based on the arguments contained in the response file.

Syntax Description

startDiagnostics.sh {**-g** *response_file* | **-r** *response_file*} [**-help**]

- **-g**—Generates the response file. You need to use this option only when generating a response file for the first time.
- **-r**—Runs the diagnostics tool using the response file
- *response_file*—Specifies the name of the response file
- **-help**—Displays help for the tool. You must use the **-help** option exclusively. Do not use this with any other option.

Examples

This results occurs when you generate the response file.

```
# ./startDiagnostics.sh -g response.txt

Please enter directory where to put output files [] /var/CSCObac
   Please enter the duration of the diagnostic (sec) [600]

Please select statistics you would like to gather on RDU

   CPU statistics (y/n/q)? [y]
   Process statistics (y/n/q)? [n]
   IO statistics (y/n/q)? [y]
   Memory statistics (y/n/q)? [y]
   Network statistics (y/n/q)? [y]
   RDU API traffic (y/n/q)? [y] n
   RDU CNR traffic (y/n/q)? [y]
   RDU DPE traffic (y/n/q)? [y] n
   RDU CNR extension traffic (y/n/q)? [y] n
   RDU SNMP traffic (y/n/q)? [y]
   System Configuration (y/n/q)? [y]

Finished generate response file (response.txt).
```

The *response.txt* is generated under the directory in which you run the **startDiagnostics.sh** script; in this case, *BPR_HOME/rdu/diagnostics/bin*. A sample response file generated for RDU diagnostics is featured here:

```
test.bundle.direcotry=/var/CSCObac
test.bundle.duration.sec=100
test.cpu.enable=true
test.process.enable=false
test.io.enable=true
test.memory.enable=true
test.network.enable=true
test.rdu_api_traffic.enable=true
test.rdu_cnr_traffic.enable=true
test.rdu_dpe_traffic.enable=true
test.rdu_cnr_ex_traffic.enable=true
test.rdu_snmp_traffic.enable=true
test.system_config.enable=true
test.rdu.port=49187
test.dpe.port=49186
test.dpe.ip=10.10.29.1
test.cnr_ex.ip=10.10.85.2
test.cnr_ex.port=
EOF
```

This result occurs when you run the diagnostics tool using the response file that you have generated.

```
# ./startDiagnostics.sh -r response.txt

You could run statusDiagnostics.sh to find out the status of the diagnostics.
You could run stopDiagnostics.sh to stop the diagnostics.
```

After you run the **startDiagnostics.sh** tool, output files for each statistic are created under the directory in which you run the tool.

Using statusDiagnostics.sh Tool

Use the **statusDiagnostics.sh** tool to determine the status of diagnostic collection for the statistics that you require.

Syntax Description

statusDiagnostics.sh, which displays the status of diagnostic collection for each statistic.

Note You cannot use the **-help** option with the **statusDiagnostics.sh** tool.

Examples

```
# ./statusDiagnostics.sh
   CPU diagnostic is running.
   Process diagnostics stopped.
   IO diagnostic is running.
   Memory diagnostic is running.
   Network diagnostic is running.
   Rdu api traffic diagnostic is running.
   Rdu cnr traffic diagnostic is running.
   Rdu dpe traffic diagnostic is running.
   Rdu cnr_ex traffic diagnostic is running.
   Rdu snmp traffic diagnostic is running.
```

Using stopDiagnostics.sh Tool

Use the **stopDiagnostics.sh** tool to stop running diagnostics for any one statistic or for all statistics. You can run this tool in the interactive mode or noninteractive mode.

Running stopDiagnostics.sh in Interactive Mode

Running **stopDiagnostics.sh** in the interactive mode, without any argument, prompts you to specify if you want to stop diagnostics for all statistics or for specific statistics.

Syntax Description

stopDiagnostics.sh [**-help**]

- **stopDiagnostics.sh**—Stops diagnostic collection in the interactive mode.
- **-help**—Displays help for the tool. You must use the **-help** option exclusively. Do not use this with any other option.

Examples

```
# ./stopDiagnostics.sh

This script allowed to stop specific diagnostic or all diagnostics.
If you would like to stop specific diagnostics, say no to question below.

   Would you like to stop all diagnostics (y/n/q)? [y]
```

Running stopDiagnostics.sh in Noninteractive Mode

Running **stopDiagnostics.sh** in the noninteractive mode stops diagnostics for all statistics.

Syntax Description

stopDiagnostics.sh -a [**-help**]

- **-a**—Stops diagnostics for all statistics without prompting you.
- **-help**—Displays help for the tool. You must use the **-help** option exclusively. Do not use this with any other option.

Examples

```
# ./stopDiagnostics.sh -a
#
```

Bundling Server State for Support

You can generate server configuration and other diagnostics information using the diagnostics tools in the *BPR_HOME/*{*rdu* | *dpe*}*/diagnostics/bin* directory. (For information on how to run these tools, see Troubleshooting Using Diagnostics Tool, page 26-5.) To make this diagnostic information available for support to the Cisco Technical Assistance Center, you must bundle the output directory that is created using the diagnostics tools into an archive. To perform this task, you use the **bundleState.sh** tool.

Note that the **bundleState.sh** tool does not gather diagnostics; it only zips and tars the data that tools such as **startDiagnostics.sh** collect.

At a minimum, the diagnostics that you bundle must include information related to system configuration. To generate system information, use either:

- captureConfiguration.sh—Collects system configuration information such as mount and disk setup, memory, and operating system and hardware data. When running this script, you must specify the output directory.
- startDiagnostics.sh—Collects performance statistics for Prime Cable Provisioning servers. When running this script to capture system configuration, you must enter **y** at the System Configuration prompt. For example:

  ```
  System Configuration (y/n/q)? [y]
  ```

  For details, see Using startDiagnostics.sh Tool, page 26-5.

For specific problems, Cisco support personnel may instruct you to collect additional diagnostics and add it to the bundle.

Syntax Description

bundleState.sh *archive_directory output_directory* [**-help**]

- *archive_directory*—Directory where you want to bundle.
- *output_directory*—Directory where you want to output the bundle.
- **-help**—Displays help for the tool. You must use the **-help** option exclusively. Do not use this with any other option.

Examples

```
# ./bundleState.sh /var/CSCObac /var/CSCObac
/var/CSCObac/state-20071129-064042
Creating state bundle for Cisco support...
+ /var/CSCObac/state-20071129-064042.bpr
+ Compressing state bundle...
+ Size: 3736K compressed, 83776K uncompressed
```

Troubleshooting DOCSIS Networks

For information on troubleshooting the DOCSIS technology with respect to Prime Cable Provisioning and the Cisco uBR7246 CMTS, see *Troubleshooting uBR Cable Modems Not Coming Online* at: http://www.cisco.com/en/US/tech/tk86/tk89/technologies_tech_note09186a0080094eb1.shtml

Troubleshooting PacketCable eMTA Provisioning

This section features information that will help you solve possible issues in a PacketCable voice technology deployment.

- Troubleshooting Tools, page 26-15
- Troubleshooting Scenarios, page 26-15
- Certificate Trust Hierarchy, page 26-19

This section assumes that you are familiar with the PacketCable Multimedia Terminal Adapter (MTA) Device Provisioning Specification, PKT-SPPROV1.5-I01-050128. See the PacketCable website for details.

Provisioning PacketCable embedded MTAs (eMTAs) is a relatively complex process; however, with the right tools and “tricks of the trade,” getting eMTAs operational can be straightforward.

This section assumes that Network Registrar and Prime Cable Provisioning are both in use; however, much of the information also applies for other deployments. Basic knowledge of Network Registrar (scopes, policies, basic DNS zone setup, and record entry) and Prime Cable Provisioning (Class of Service, DHCP Criteria, files, and Prime Cable Provisioning directory structure) is assumed.

The PacketCable eMTA provisioning process consists of 25 steps for the Secure flow; the Basic flow has far fewer steps. To troubleshoot eMTAs, knowledge of these 25 steps from the PacketCable provisioning specification is absolutely essential. See Configuring PacketCable, page 7-4.

This section contains the following topics:

- Components, page 26-11
- Key Variables, page 26-13

Components

Before troubleshooting eMTAs, you should be familiar with the following system components.

- eMTA
- DHCP Server
- DNS Server
- KDC

- PacketCable Provisioning Server
- Call Management Server

eMTA

The eMTA is a cable modem and an MTA in one box, with a common software image. The CM and MTA each have their own MAC addresses and each performs DHCP to get its own IP address. The eMTA contains, at minimum, three certificates. One certificate is a unique MTA certificate. A second certificate identifies the MTA manufacturer. Both the device and manufacture certificates are sent by the MTA to authenticate itself to the KDC. The third certificate is a telephony root certificate used to verify the certificates sent by the KDC to the MTA. The KDC certificates will be chained from the telephony root, therefore the telephony root must reside on the MTA to validate the authenticity of the KDC certificates. The MTA portion receives its own configuration file, which it uses to identify its controlling call agent, among other things.

DHCP Server

The DOCSIS specifications mandate that cable modems negotiate their IP addresses using DHCP. The MTA, like most CPE on a DOCSIS network, must use DHCP to obtain its IP address and other crucial information (DNS servers, PacketCable Option 122 for Kerberos realm name, provisioning server FQDN).

Note The cable modem portion, in addition to its normally required DHCP options, also requests, and must receive, Option 122 suboption 1, which it passes to the MTA portion as the IP address of the correct DHCP server from which to accept offers.

When using Prime Cable Provisioning with PacketCable support, be aware that a correctly configured Prime Cable Provisioning will automatically populate the ToD server, DNS servers, TFTP server, as well as the Option 122 fields; these do not need to be explicitly set in the Network Registrar policy.

DNS Server

The Domain Name System (DNS) server is fundamental in PacketCable provisioning. The PacketCable provisioning server, which is the device provisioning engine (DPE) in a Prime Cable Provisioning architecture, must have an address (A) record in the appropriate zone, because its fully qualified domain name (FQDN) is provided to the MTA in Option 122 by the DHCP server. The KDC realm must have a zone of the same name as the realm name containing a server (SRV) record that contains the FQDN of the Kerberos server.

The Kerberos server identified in the SRV record must itself have an A record in the appropriate zone. The call management server (CMS) identified in the MTA configuration file must also have an A record in the appropriate zone. Lastly, the MTAs themselves must have A records in the appropriate zone, because the CMS reaches the MTA by resolving its FQDN. Dynamic DNS (DDNS) is the preferred method of creating A records for the MTA. See Cisco Prime Network Registrar documentation for information on configuring and troubleshooting DDNS.

KDC

The KDC is responsible for authenticating MTAs. As such, it must check the MTA certificate, and provide its own certificates so that the MTA can authenticate the KDC. It also communicates with the DPE (the provisioning server) to validate that the MTA is provisioned on the network.

PacketCable Provisioning Server

The PacketCable provisioning server is responsible for communicating the location of the MTA configuration file to the MTA, and/or provisioning MTA parameters via SNMP. SNMPv3 is used for all communication between the MTA and the provisioning server. The keys used to initiate SNMPv3 communication are obtained by the MTA during its authentication phase with the KDC. Provisioning server functionality is provided by the DPE in a Prime Cable Provisioning architecture.

Call Management Server

The call management server (CMS) is essentially a soft switch, or call-agent, with additional PacketCable functionality to control, among other things, quality of service on a cable network. The MTA sends a network call signaling (NCS) restart in progress (RSIP) message to the CMS upon successful PacketCable provisioning.

Key Variables

This section describes the key variables required to provision an eMTA correctly.

- Certificates, page 26-13
- Scope-Selection Tags, page 26-14
- MTA Configuration File, page 26-14

Certificates

The *MTA_Root.cer* file contains the MTA root certificate (a certificate that is rooted in the official PacketCable MTA root).

You must know in advance what telephony root certificate is required for the MTAs you want to provision. Deployments in production networks use telephony certificates rooted in the PacketCable real root. There is also a PacketCable test root used in testing environments.

The KDC certificates used by the KDC to authenticate itself to the MTA must be rooted in the same telephony root that is stored on the MTA (PacketCable real or test root). Most MTA vendors support test images that have Telnet and/or HTTP login capabilities so that you can determine which telephony root is enabled, and change the root used (in most cases, you can only select between the PacketCable real or test root).

The most common scenario has the KDC loaded with certificates (from the *BPR_HOME/kdc/<Operating System>/packetcable/certificates* directory) as follows:

- *CableLabs_Service_Provider_Root.cer*
- *Service_Provider.cer*
- *Local_System.cer*
- *KDC.cer*
- *MTA_Root.cer*

The first four certificates comprise the telephony certificate chain. The *MTA_Root.cer* file contains the MTA root used by the KDC to validate the certificates sent by the MTA.

Note See Using PKCert.sh, page 28-3, for information on installing and managing KDC certificates.

To determine if you are using PacketCable test root, open the *CableLabs_Service_Provider_Root.cer* file in Windows, and validate that the Subject OrgName entry is **O = CableLabs**, and/or check that the Subject Alternative name reads **CN=CABLELABS GENERATED TEST ROOT FOR EQUIPMENT TEST PURPOSES ONLY**.

The KDC certificate (*KDC.cer*) contains the realm name to use. The realm name that Prime Cable Provisioning (and the corresponding DNS zone) is configured to use must match this realm name. Additionally, the MTA configuration file realm org name must match the organization name as seen in the telephony root.

The KDC certificate has a corresponding private key that must be installed in the *BPR_HOME/kdc/solaris* directory. Usually it is named KDC_private_key.pkcs8 or KDC_private_key_proprietary. When changing certificates, you must also change the private key.

Scope-Selection Tags

In most scenarios, Prime Cable Provisioning is involved in processing all DHCP requests from scopes with scope-selection tags that match selection criteria specified in the DHCP Criteria page of the Prime Cable Provisioning administrator user interface. Client class can also be used to tie scopes to Prime Cable Provisioning processing; ensure you make this association before you attempt to provision devices.

MTA Configuration File

The MTA configuration file contains the location of the CMS. Additionally, it must contain an entry for Realm Name. This value must match that of the certificate chain in use.

Certain table entries within the MTA configuration file are indexed by the realm name delivered to the MTA in Option 122. This realm name entry in the MTA configuration file must match that delivered in Option 122. For example, if **DEF.COM** was the realm name delivered in Option 122, MTA configuration file entries in the pktcMtaDevRealm table would be indexed with a suffix made up of the ASCII-coded character values (in dot-delimited decimal format when using the Cisco Broadband Configurator) of the realm name; for example 68.69.70.46.67.79.77. There are many free ASCII conversion pages available on the web to make this conversion easier.

Troubleshooting Tools

The 25 eMTA Secure provisioning steps contained in the PacketCable MTA Device Provisioning Specification are shown in Figure 7-1. This section describes:

- Logs, page 26-15
- Ethereal, SnifferPro, or Other Packet Capture Tools, page 26-15

Logs

These log files are used to maintain the following information:

- The Network Registrar has two logs (*name_dhcp_1_log* and *name_dns_1_log*), which contain the most recent logging entries from Network Registrar. Look in these files for DHCP- or DNS-related problems.
- The *BPR_HOME/kdc/logs/kdc.log* file shows all KDC interactions with MTAs, and KDC interactions with the DPE.
- The *BPR_DATA/dpe/logs/dpe.log* file shows the major steps related to SNMPv3 interaction with the MTA.

Note Turning on the tracing of snmp, registration server, and registration server detail messages, using the command-line interface (CLI), helps to troubleshoot potential PacketCable problems. For information on the appropriate troubleshooting commands, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.

Ethereal, SnifferPro, or Other Packet Capture Tools

A packet capture tool is indispensable when troubleshooting the eMTAs. The Ethereal version, as packaged by CableLabs, includes numerous packet decoders specific to PacketCable. These include the Kerberos AS and AP packets.

- If you suspect that a specific failure is DHCP-related, capture packets while filtering on packets sourced from, or destined to, the CMTS cable interface IP address and the DHCP server IP address.
- If you suspect that a specific failure is related to any of the 25 steps occurring after DHCP, filter all packets to and from the eMTA IP address. This provides a very concise, easy-to-follow trace of provisioning steps 5 through 25, as shown in Figure 7-1.

Troubleshooting Scenarios

The scenarios listed in Table 26-3 are possible failures involving eMTAs.

Table 26-3 Troubleshooting Scenarios

If this problem occurs...	Which indicates this potential cause...	To correct it, you should...
The KDC does not start.	The KDC certificate does not correspond to the private key.	Ensure that you have matching certificates and private key.
	The KDC license expired or is missing.	Restore KDC license to *BPR_HOME/kdc* directory.

Table 26-3 Troubleshooting Scenarios (continued)

If this problem occurs...	Which indicates this potential cause...	To correct it, you should...
The MTA device does not appear in the Prime Cable Provisioning Devices page.	An incorrect cable helper address may have been configured.	Fix the helper address.
	The scope-selection tags do not match the DHCP Criteria selected in the Prime Cable Provisioning user interface.	Verify that the MTA scope-selection tags match those in the PacketCable DHCP Criteria created, in Prime Cable Provisioning, for the relevant MTAs.
	The Network Registrar extension point is not properly installed.	Reinstall the Network Registrar extension point. See the *Cisco Prime Cable Provisioning 5.0 Quick Start Guide*.
	The cable modem portion did not receive Option 122.	Verify that the tags on the scope of the cable modem portion match the DOCSIS DHCP Criteria configured for Prime Cable Provisioning.
The MTA device does not accept the DHCP offer and continually cycles through the DHCP flow.	There are invalid DHCP options configured.	Check that scope policy includes the DNS server option, and/or check that the *cnr_ep.properties* file includes entries for primary and secondary DNS servers.
	The DHCP offer may have come from a DHCP server different from the one indicated in the cable modem portion's Option 122 suboption 1.	Check the *cnr_ep.properties* file to ensure that the main and backup DHCP servers are set correctly.
Both the *kdc.log* file and the ethereal trace indicate that the MTA device never contacts the KDC.	An incorrect DNS server is specified in the *cnr_ep.properties* file or the MTA scope policy, or both.	Check or correct *cnr_ep.properties* DNS servers.
	A zone is missing or has been incorrectly set up for the Kerberos realm.	Make sure a zone with same name as realm is created and contains an 'SRV' record of format '_kerberos._udp 0 0 88 *KDC FQDN*'.
	There is a missing or incorrect KDC 'A' record entry.	Ensure that an 'A' record exists for the FQDN contained in the Kerberos zone's 'SRV' record.
	The DPE FQDN cannot be resolved.	Ensure that the provFQDNs entry in *dpe.properties* has the correct FQDN and IP of the DPE.

Table 26-3 Troubleshooting Scenarios (continued)

If this problem occurs...	Which indicates this potential cause...	To correct it, you should...
The KDC reports failure during the Kerberos AS-Request.	The MTA certificate does not match the MTA root used by KDC.	Verify that the *MTA_Root.cer* is correct by comparing the *MTA_Root.cer* against that used on a working system. If it is correct, the MTA itself could have a certificate problem. This situation is extremely rare and if this is the case, contact the MTA manufacturer.
	FQDN lookup by KDC to Prov Server failed. The device may not yet be provisioned in Prime Cable Provisioning.	Verify that the device appears. It should be given both a Class of Service and a DHCP Criteria.
	A clock skew error. See PacketCable Workflows, page 7-5, for additional information.	Ensure that all Prime Cable Provisioning network elements are clock-synced via NTP. See the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.
	A mismatch may exist between the KDC and the DPE. **Note** If other devices are provisioned correctly, this is probably not the cause of the problem.	Check that these three entries exist in the *BPR_HOME/kdc/<Operating System>/keys* directory: • mtafqdnmap,dpe.abc.com@DEF.COM • mtaprovsrvr,dpe.abc.com@DEF.COM • krbtgt,DEF.COM@DEF.COM The DPE FQDN and realm name on your system will be different from this example. Contents of these entries must match the entry in either the dpe.properties 'KDCServiceKey' entry, or the keys generated using the KeyGen utility.
The KDC reports success at the AS-Request/Reply (steps 9 and 10 in Figure 7-1), but the MTA device never moves past step 9.	There is a certificate mismatch between the telephony root loaded or enabled on the MTA, and that loaded on the KDC.	Check certificates on MTA and KDC.
	Although highly unlikely, it is possible that there is a corrupted telephony certificate chain. **Note** If other devices are provisioned correctly, this is not the cause of the problem.	Ensure that the correct certificate is loaded or enabled on MTA. If no devices can be provisioned correctly, try a different certificate on the KDC.
Failure at AP Request/Reply (step 14 in Figure 7-1).	A clock skew error. See PacketCable Workflows, page 7-5, for additional information.	Ensure that all Prime Cable Provisioning network elements are clock-synced via NTP. See the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.
	Cannot resolve Prov Server FQDN.	Make sure that the provisioning server (DPE) has a correct DNS entry. Ensure that dpe.properties provFQDNs entry has the correct FQDN and IP of the provisioning server (DPE).
	There is no route from the MTA to the DPE.	Correct the routing problem.

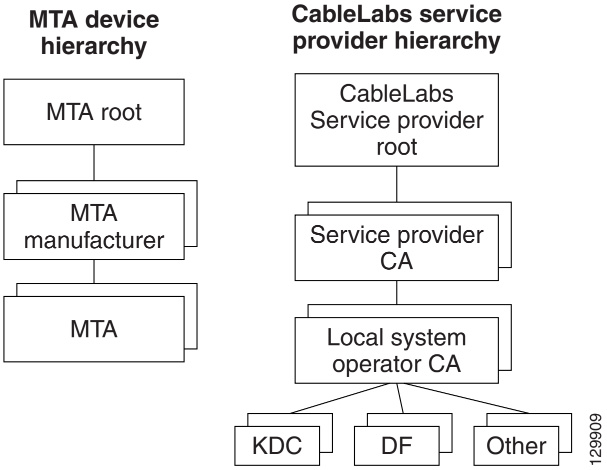
Table 26-3 Troubleshooting Scenarios (continued)

If this problem occurs...	Which indicates this potential cause...	To correct it, you should...
The MTA device never issues a TFTP request for a configuration file.	There is no route to the TFTP server running on the DPE.	Correct the routing problem.
The MTA device never receives the TFTP configuration file.	The configuration file is not cached at the DPE.	Wait until the next provisioning attempt, at which time the file should be cached. If this fails, reset the MTA.
	A conflicting TFTP server option is included in the network registrar MTA scope policy.	Because Prime Cable Provisioning inserts the DPE address for the TFTP server, you can safely remove this option from the policy.
The MTA device receives a configuration file, but the DPE fails to receive the SNMP Inform (step 25 in Figure 7-1) as seen in the *dpe.log* file.	One of: • An internal conflict in the configuration file. • A conflict with Realm origin of the telephony certificate chain. • A conflict with the Realm Name provided in Option 122.	Ensure that the MTA configuration file is consistent.
The MTA device reports success (step 25 in Figure 7-1) although an RSIP is not sent.	The MTA cannot resolve the IP address of the CMS FQDN given in the MTA configuration file.	Verify that a DNS entry exists for the CMS.
	The MTA cannot reach the IP address(es) of the CMS. This is an indication that no route is configured.	Resolve all routing problems.
The MTA device reports success (step 25 in Figure 7-1), although it proceeds to contact the KDC again for CMS service.	The MTA configuration file points to an incorrect cable modem.	Correct the configuration file, or reconfigure the Cisco BTS 10200 to use the FQDN listed in the configuration file.
	The MTA configuration file has its pktcMtaDevCmsIPsecCtrl value missing, or it is set to 1. This means it will perform secure NCS call signaling, or it will use an ASCII suffix that does not match that of the CMS FQDN.	Correct the configuration file. If you intend to perform secure signaling, take the necessary steps to configure the KDC and the BTS for support.
The MTA device reports success (step 25 in Figure 7-1), RSIPs, but gets no response or gets an error in response from the soft switch.	The MTA is unprovisioned or has been incorrectly provisioned on the Cisco BTS 10200.	Provision MTA on the Cisco BTS 10200.
	An eMTA DNS entry does not exist.	Place an entry in the correct DNS zone for the eMTA. Dynamic DNS is the preferred method. See Cisco Prime Network Registrar documentation for information on enabling DDNS.

Certificate Trust Hierarchy

There are two certificate hierarchies affiliated with Prime Cable Provisioning PacketCable, the MTA Device Certificate Hierarchy and the CableLabs Service Provider Certificate Hierarchy, as shown in Figure 26-1.

Figure 26-1 PacketCable Certificate Hierarchy



Before implementing PacketCable in Prime Cable Provisioning, you should thoroughly familiarize yourself with these technology documents:

- *RFC 2459 Internet X.509 Public Key Infrastructure Certificate and CRL Profile*
- *DOCSIS Baseline Privacy Plus Interface Specification, SP-BPI+-I11-040407, April 7, 2004*

Note While Euro PacketCable uses the security specifications from PacketCable [PKT-SP-SEC-I08-030415], some changes are needed in relation to the digital certificates that are used in a Euro-PacketCable environment. To keep Euro PacketCable and PacketCable as alike as possible, Euro PacketCable uses all PacketCable security technology, including new revisions of the security specifications [PKTSP-SEC-I08-030415].

The elements of the Euro-PacketCable certificates that are different from the PacketCable certificates are indicated in the tables below.

For Euro PacketCable, the Euro-PacketCable certificates are the only valid certificates; any requirements that are stated in [PKT-SP-SEC-I08-030415] for PacketCable that refer to PacketCable Certificates are changed to the corresponding requirements for the Euro-PacketCable certificates.

Euro-PacketCable-compliant eMTAs must have the Euro-DOCSIS root CVC CA's public key stored in the cable modem's nonvolatile memory instead of in the DOCSIS CVC CA's public key. Standalone MTAs that comply with Euro PacketCable must have the tComLabs CVC Root Certificate and the tComLabs CVC CA certificate stored in non-volatile memory. The CVC of manufacturers are verified by checking the certificate chain.

Certificate Validation

PacketCable certificate validation in general involves validation of an entire chain of certificates. For example, when the provisioning server validates an MTA Device certificate, the following chain of certificates is validated:

MTA Root Certificate + MTA Manufacturer Certificate + MTA Device Certificate

The signature on the MTA Manufacturer Certificate is verified with the MTA Root Certificate and the signature on the MTA Device Certificate is verified with the MTA Manufacturer Certificate. The MTA Root Certificate is self-signed and is known in advance to the provisioning server. The public key present in the MTA Root Certificate is used to validate the signature on this same certificate.

Usually the first certificate in the chain is not explicitly included in the certificate chain that is sent over the wire. In the cases where the first certificate is explicitly included it must already be known to the verifying party ahead of time and must *not* contain any changes to the certificate with the possible exception of the certificate serial number, validity period, and the value of the signature. If changes other than these exist in the CableLabs Service Provider Root Certificate that was passed over the wire in comparison to the known CableLabs Service Provider Root Certificate, the device making the comparison must fail the certificate verification.

The exact rules for certificate chain validation must fully comply with RFC 2459, where they are referred to as Certificate Path Validation. In general, X.509 certificates support a liberal set of rules for determining if the issuer name of a certificate matches the subject name of another. The rules are such that two name fields may be declared to match even though a binary comparison of the two name fields does not indicate a match. RFC 2459 recommends that certificate authorities restrict the encoding of name fields so that an implementation can declare a match or mismatch using simple binary comparison.

PacketCable security follows this recommendation. Accordingly, the DER-encoded tbsCertificate.issuer field of a PacketCable certificate must be an exact match to the DER-encoded tbsCertificate.subject field of its issuer certificate. An implementation may compare an issuer name to a subject name by performing a binary comparison of the DER-encoded tbsCertificate.issuer and tbsCertificate.subject fields.

The sections below specify the required certificate chain, which must be used to verify each certificate that appears at the leaf group (at the bottom) in the PacketCable certificate trust hierarchy illustrated in Figure 26-1.

Validity period nesting is not checked and intentionally not enforced. Thus, the validity period of a certificate need not fall within the validity period of the certificate that issued it.

MTA Device Certificate Hierarchy

The device certificate hierarchy exactly mirrors that of the DOCSIS1.1/BPI+ hierarchy. It is rooted at a CableLabs-issued PacketCable MTA Root Certificate, which is used as the issuing certificate of a set of manufacturer certificates. The manufacturer certificates are used to sign the individual device certificates.

The information contained in the following tables contains the PacketCable-specific values for the required fields according to RFC 2459. These PacketCable-specific values must be followed according to Table 26-4, except that Validity Periods should be as given in the respective tables. If a required field is not specifically listed for PacketCable, then follow the guidelines in RFC 2459.

MTA Root Certificate

This certificate must be verified as part of a certificate chain containing the MTA Root Certificate, the MTA Manufacturer Certificate, and the MTA Device Certificate.

Table 26-4 lists the values relevant to the MTA Root Certificate.

Table 26-4 MTA Root Certificate

MTA Root Certificate		

Subject Name Form	**PacketCable**
	C=US
	O=CableLabs
	OU=PacketCable
	CN=PacketCable Root Device Certificate Authority
		Euro PacketCable
		C=BE
		O=tComLabs
		OU=Euro-PacketCable
		CN=Euro-PacketCable Root Device Certificate Authority

Intended Usage	This certificate is used to sign MTA Manufacturer Certificates and is used by the KDC. This certificate is not used by the MTAs and thus does not appear in the MTA MIB.	
Signed By	Self-signed	
Validity Period	20+ years. It is intended that the validity period is long enough that this certificate is never reissued.	
Modulus Length	2048	
Extensions	keyUsage[c,m](keyCertSign, cRLSign) subjectKeyIdentifier[n,m] basicConstraints[c,m](cA=true, pathLenConstraint=1)	

MTA Manufacturer Certificate

This certificate must be verified as part of a certificate chain containing the MTA Root Certificate, the MTA Manufacturer Certificate, and the MTA Device Certificate. The state/province, city, and manufacturer’s facility are optional attributes. A manufacturer may have more than one manufacturer’s certificate, and there may exist one or more certificates per manufacturer. All certificates for the same manufacturer may be provided to each MTA either at manufacture time or during a field update. The MTA must select an appropriate certificate for its use by matching the issuer name in the MTA Device Certificate with the subject name in the MTA Manufacturer Certificate. If present, the authorityKeyIdentifier of the device certificate must match the subjectKeyIdentifier of the manufacturer certificate as described in RFC 2459. The *CompanyName* field that is present in O and CN may be different in the two instances.

Table 26-5 lists the values relevant to the MTA Manufacturer Certificate.

Table 26-5 MTA Manufacturer Certificates

MTA Manufacturer Certificate		

Subject Name Form	**PacketCable**
	C=US
	O=CableLabs
	OU=PacketCable
	CN=PacketCable Root Device Certificate Authority
		Euro PacketCable
		C=*Country of Manufacturer*
		O=*Company Name*
		[stateOrProvinceName = *State/Province*]
		[localityName=*City*]
		OU=Euro-PacketCable
		[organizationalUnitName= *Manufacturing Location*]
		CN=*Company Name* Euro-PacketCable CA

Intended Usage	This certificate is issued to each MTA manufacturer and can be provided to each MTA as part of the secure code download as specified by the PacketCable Security Specification (either at manufacture time, or during a field update). This certificate appears as a read-only parameter in the MTA MIB. This certificate along with the MTA Device Certificate is used to authenticate the MTA device identity (MAC address) during authentication by the KDC.	
Signed By	MTA Root Certificate CA	
Validity Period	20 years	
Modulus Length	2048	
Extensions	keyUsage[c,m](keyCertSign, cRLSign), subjectKeyIdentifier[n,m], authorityKeyIdentifier[n,m](keyIdentifier=*subjectKeyIdentifier value from CA certificate*), basicConstraints[c,m](cA=true, pathLenConstraint=0)	

MTA Device Certificate

This certificate must be verified as part of a certificate chain containing the MTA Root Certificate, the MTA Manufacturer Certificate, and the MTA Device Certificate. The state/province, city, and manufacturer's facility are optional attributes. The MAC address must be expressed as six pairs of hexadecimal digits separated by colons; for example, "00:60:21:A5:0A:23". The alpha hexadecimal characters (A-F) must be expressed as uppercase letters. The MTA device certificate should not be replaced or renewed.

Table 26-6 lists the values relevant to the MTA Device Certificate.

Table 26-6 MTA Device Certificates

MTA Device Certificate		
Subject Name Form	**PacketCable** C=*Country* O=*Company Name* [ST=*State/Province*] [L=*City*], OU=PacketCable [OU=*Product Name*] [OU=*Manufacturer's Facility*] CN=*MAC Address*	**Euro PacketCable** C=*Country of Manufacturer* O=*Company Name* [ST=*State/Province*] [L=*City*] OU=Euro-PacketCable [OU=*Product Name*] [OU=*Manufacturing Location*] CN=*MAC Address*
Intended Usage	This certificate is issued by the MTA manufacturer and installed in the factory. The provisioning server cannot update this certificate. This certificate appears as a read-only parameter in the MTA MIB. This certificate is used to authenticate the MTA device identity (MAC address) during provisioning.	
Signed By	MTA Manufacturer Certificate CA	
Validity Period	At least 20 years	
Modulus Length	1024, 1536, or 2048	
Extensions	keyUsage[c,o](digitalSignature, keyEncipherment) authorityKeyIdentifier[n,m](keyIdentifier=*subjectKeyIdentifier value from CA certificate*)	

MTA Manufacturer Code Verification Certificates

Code Verification Certificate (CVC) specification for eMTAs must be identical to the DOCSIS 1.1 CVC, specified in DOCSIS specification SP-BPI+-I11-040407.

CableLabs Service Provider Certificate Hierarchy

The Service Provider Certificate Hierarchy is rooted at a CableLabs-issued CableLabs Service Provider Root certificate. That certificate is used as the issuing certificate of a set of service provider's certificates. The service provider's certificates are used to sign an optional local system certificate. If the local system certificate exists then that is used to sign the ancillary equipment certificates; otherwise, the ancillary certificates are signed by the Service Provider's CA.

The information contained in Table 26-7 contains the specific values for the required fields according to RFC 2459. These specific values must be followed. If a required field is not specifically listed, then the guidelines in RFC 2459 must be followed exactly.

CableLabs Service Provider Root Certificate

Before any Kerberos key management can be performed, an MTA and a KDC need to perform mutual authentication using the PKINIT extension to the Kerberos protocol. An MTA authenticates a KDC after it receives a PKINIT Reply message containing a KDC certificate chain. In authenticating the KDC, the MTA verifies the KDC certificate chain, including the KDC's Service Provider Certificate signed by the CableLabs Service Provider Root CA.

Table 26-7 lists the values relevant to the CableLabs Service Provider Root Certificate.

Table 26-7 CableLabs Service Provider Root Certificates

CableLabs Service Provider Root Certificate		

Subject Name Form	**PacketCable**
	C=US
	O=CableLabs
	CN=CableLabs Service Provider Root CA
		Euro PacketCable
		C=BE
		O=tComLabs
		CN=tComLabs Service Provider Root CA

Intended Usage	This certificate is used to sign Service Provider CA certificates. This certificate is installed into each MTA at the time of manufacture or with a secure code download as specified by the PacketCable Security Specification and cannot be updated by the provisioning server. Neither this root certificate nor the corresponding public key appears in the MTA MIB.	
Signed By	Self-signed	
Validity Period	20+ years. It is intended that the validity period is long enough that this certificate is never reissued.	
Modulus Length	2048	
Extensions	keyUsage[c,m](keyCertSign, cRLSign) subjectKeyIdentifier[n,m] basicConstraints[c,m](cA=true)	

Service Provider CA Certificate

This is the certificate held by the service provider, signed by the CableLabs Service Provider Root CA. It is verified as part of a certificate chain that includes the CableLabs Service Provider Root Certificate, the Telephony Service Provider Certificate, an optional Local System Certificate, and an end-entity server certificate. The authenticating entities normally already possess the CableLabs Service Provider Root Certificate and it is not transmitted with the rest of the certificate chain.

The fact that a Service Provider CA Certificate is always explicitly included in the certificate chain allows a Service Provider the flexibility to change its certificate without requiring reconfiguration of each entity that validates this certificate chain (for example, an MTA validating a PKINIT Reply). Each time the Service Provider CA Certificate changes, its signature must be verified with the CableLabs Service Provider Root Certificate. However, a new certificate for the same Service Provider must preserve the same value of the OrganizationName attribute in the SubjectName. The *Company* field that is present in O and CN may be different in the two instances.

Table 26-8 lists the values relevant to the CableLabs Service Provider CA Certificate.

Table 26-8 CableLabs Service Provider CA Certificates

CableLabs Service Provider Root Certificate		

Subject Name Form	**PacketCable**
	C=*Country*
	O=*Company*
	CN=*Company* CableLabs Service Provider CA
		Euro PacketCable
		C=*Country*
		O=*Company*
		CN=*Company* tComLabs Service Provider CA

Intended Usage	This certificate is used to sign Service Provider CA certificates. This certificate is installed into each MTA at the time of manufacture or with a secure code download as specified by the PacketCable Security Specification and cannot be updated by the provisioning server. Neither this root certificate nor the corresponding public key appears in the MTA MIB.	
Signed By	Self-signed	
Validity Period	20+ years. It is intended that the validity period is long enough that this certificate is never reissued.	
Modulus Length	2048	
Extensions	keyUsage[c,m](keyCertSign cRLSign), subjectKeyIdentifier[n,m] basicConstraints[c,m](cA=true)	

Local System CA Certificates

A Service Provider CA may delegate the issuance of certificates to a regional Certification Authority called Local System CA (with the corresponding Local System Certificate). Network servers are allowed to move freely between regional Certification Authorities of the same Service Provider. Therefore, the MTA MIB does not contain any information regarding a Local System Certificate (which might restrict an MTA to KDCs within a particular region).

Table 26-9 lists the values relevant to the Local System CA Certificate.

Table 26-9 Local System CA Certificates

Local System CA Certificate		

Subject Name Form	**PacketCable**
	C=*Country*
	O=*Company*
	OU=*Local System Name*
	CN=*Company* CableLabs Local System CA
		Euro PacketCable
		C=*Country*
		O=*Company*
		OU=*Local System Name*
		CN=*Company* tComLabs Local System CA

Intended Usage	A Service Provider CA may delegate the issuance of certificates to a regional Certification Authority called a Local System CA (with the corresponding Local System Certificate). Network servers are allowed to move freely between regional Certification Authorities of the same Service Provider. Therefore, the MTA MIB does not contain any information regarding a Local System Certificate (which might restrict an MTA to KDCs within a particular region).	

Table 26-9 Local System CA Certificates (continued)

Local System CA Certificate	
Signed By	Service Provider CA Certificate
Validity Period	20 years.
Modulus Length	1024, 1536, 2048
Extensions	keyUsage[c,m](keyCertSign, cRLSign), subjectKeyIdentifier[n,m], authorityKeyIdentifier[n,m](keyIdentifier=*subjectKeyIdentifier value from CA certificate*), basicConstraints[c,m](cA=true, pathLenConstraint=0)

Operational Ancillary Certificates

All these are signed by either the Local System CA or by the Service Provider CA. Other ancillary certificates may be added to this standard at a later time.

KDC Certificate

This certificate must be verified as part of a certificate chain containing the CableLabs Service Provider Root Certificate, the Service Provider CA Certificate, and the Ancillary Device Certificates. The PKINIT specification requires the KDC certificate to include the subjectAltName v.3 certificate extension, the value of which must be the Kerberos principal name of the KDC.

Table 26-10 lists the values relevant to the KDC Certificate.

Table 26-10 KDC Certificates

Key Distribution Center Certificate		

Subject Name Form	**PacketCable**
	C=*Country*
	O=*Company*,
	[OU=*Local System Name*]
	OU= CableLabs Key Distribution Center
	CN=*DNS Name*
		Euro PacketCable
		C=*Country*
		O=*Company*
		[OU=*Local System Name*]
		OU=tComLabs Key Distribution Center
		CN=*DNS Name*

Intended Usage	To authenticate the identity of the KDC server to the MTA during PKINIT exchanges. This certificate is passed to the MTA inside the PKINIT replies and is therefore not included in the MTA MIB and cannot be updated or queried by the provisioning server.	
Signed By	Service Provider CA Certificate or Local System Certificate	
Validity Period	20 years	
Modulus Length	1024, 1536, or 2048	
Extensions	keyUsage[c,o](digitalSignature)authorityKeyIdentifier[n,m](keyIdentifier= *subjectKeyIdentifier value from CA certificate*)subjectAltName[n,m]	

Delivery Function (DF)

This certificate must be verified as part of a certificate chain containing the CableLabs Service Provider Root Certificate, the Service Provider CA Certificate, and the Ancillary Device Certificates. This certificate is used to sign phase 1 IKE intradomain exchanges between DFs (which are used in electronic surveillance). Although the Local System Name is optional, it is required when the Local System CA signs this certificate. The IP address must be specified in standard dotted-quad notation; for example, 245.120.75.22.

Table 26-11 lists the values relevant to the DF Certificate.

Table 26-11 DF Certificates

DF Certificate		

Subject Name Form	**PacketCable**
	C=*Country*
	O=*Company*
	[OU=*Local System Name*]
	OU=PacketCable Electronic Surveillance
	CN=*IP address*
		Euro PacketCable
		C=*Country*
		O=*Company*
		[OU=*Local System Name*]
		OU=Euro-PacketCable Electronic Surveillance
		CNe=*IP address*

Intended Usage	To authenticate IKE key management, used to establish IPsec Security Associations between pairs of DFs. These Security Associations are used when a subject that is being legally wiretapped forwards the call, and event messages containing call information have to be forwarded to a new wiretap server (DF).	
Signed By	Service Provider CA Certificate or Local System CA Certificate	
Validity Period	20 years	
Modulus Length	2048	
Extensions	keyUsage[c,o](digitalSignature) authorityKeyIdentifier[n,m](keyIdentifier=*subjectKeyIdentifier value from CA certificate*) subjectAltName[n,m] (dNSName=*DNSName*)	

PacketCable Server Certificates

These certificates must be verified as part of a certificate chain containing the CableLabs Service Provider Root Certificate, the Service Provider Certificate, the Local System Operator Certificate (if used), and the Ancillary Device Certificates. These certificates are used to identify various servers in the PacketCable system. For example, they may be used to sign phase 1 IKE exchanges or to authenticate a PKINIT exchange. Although the Local System Name is optional, it is required when the Local System CA signs this certificate. 2IP address values must be specified in standard dotted decimal notation; for example, 245.120.75.22. DNS Name values must be specified as a fully qualified domain name (FQDN); for example, device.packetcable.com.

Table 26-12 lists the values relevant to the PacketCable Server Certificate.

Table 26-12 PacketCable Server Certificates

PacketCable Server Certificates		

Subject Name Form	**PacketCable**
	C=*Country*
	O=*Company*
	OU=PacketCable
	OU=[*Local System Name*]
	OU=*Sub-System Name*
	CN=*Server Identifier*[:*Element ID*]
	The value of *Server Identifier* must be the server's FQDN or its IP address, optionally followed by a colon (:) and an Element ID with no spaces before or after the colon.
	Element ID is the identifier that appears in billing event messages. It must be included in the certificate of every server that is capable of generating event messages. This includes a CMS, CMTS, and MGC. [8] defines the Element ID as a 5-octet right-justified, space-padded, ASCII-encoded, numerical string. When converting the Element ID for use in a certificate, spaces must be converted to ASCII zeros (0x48).
	For example, a CMTS with Element ID 311 and IP address 123.210.234.12 will have a common name "123.210.234.12: 00311".
	The value of *Sub-System Name* must be one of the following:
	• For Border Proxy: bp
	• For Cable Modem Termination System: cmts
	• For Call Management Server: cms
	• For Media Gateway: mg
	• For Media Gateway Controller: mgc
	• For Media Player: mp
	• For Media Player Controller: mpc
	• For Provisioning Server: ps
	• For Record Keeping Server: rks
	• For Signaling Gateway: sg
		Euro PacketCable
		C=*Country*
		O=*Company*
		OU=Euro-PacketCable
		[OU=*Local System Name*]
		OU=*Sub-system Name*
		CN=*Server Identifier*[:*Element ID*]
		See [PKT-SP-SEC-IO8-030415] for additional specifications on the commonName.

Intended Usage	These certificates are used to identify various servers in the PacketCable system. For example, they may be used to sign phase 1 IKE exchanges or to authenticate a device in a PKINIT exchange.	
Signed By	Telephony Service Provider Certificate or Local System Certificate	

Table 26-12 PacketCable Server Certificates (continued)

PacketCable Server Certificates	
Validity Period	Set by MSO policy
Modulus Length	2048
Extensions	keyUsage[c,o](digitalSignaturekeyEncipherment) authorityKeyIdentifier[n,m](keyIdentifier=*subjectKeyIdentifier value from CA cert*) subjectAltName[n,m](dNSName=*DNSName* | iPAddress=*IP AddressName*) The keyUsage tag is optional. When it is used it must be marked as critical. Unless otherwise described below, the subjectAltName extension must include the corresponding name value as specified in the CN field of the subject.

The CN attribute value for CMS certificates must be the Element ID. The subjectAltName extension must include either the IP address or the FDQN of the CMS. The CN attribute value for CMTS certificates must be the Element ID. The subjectAltName extension must include either the IP address or the FDQN of the CMTS.

The CN attribute value for MGC certificates must be the Element ID. The subjectAltName extension must include either the IP address or the FDQN of the MGC.

Certificate Revocation

Out of scope for PacketCable at this time.

Code Verification Certificate Hierarchy

The CableLabs Code Verification Certificate (CVC) PKI is generic in nature and applicable to all CableLabs projects needing CVCs. This means the basic infrastructure can be re-used for every CableLabs project. There may be differences in the end-entity certificates required for each project, but in the cases where end-entity certificates overlap, one end-entity certificate could be used to support the overlap.

The CableLabs CVC hierarchy does not apply to eMTAs.

Common CVC Requirements

The following requirements apply to all Code Verification Certificates:

- Certificates must be DER encoded.
- Certificates must be version 3.
- Certificates must include the extensions that are specified in the following tables and must *not* include any additional extensions.
- The public exponent must be F4 (65537 decimal).

CableLabs Code Verification Root CA Certificate

This certificate must be validated as part of the certificate chain containing the CableLabs Code Verification Root CA Certificate, the CableLabs Code Verification CA, and the Code Verification Certificates. See Certificate Validation, page 26-20, for additional information on how to validate certificates.

Table 26-13 lists the values relevant to the CableLabs Code Verification Root CA Certificate.

Table 26-13 CableLabs Code Verification Root CA Certificates

CableLabs Code Verification Root CA Certificate		

Subject Name Form	**PacketCable**
	C=US
	O=CableLabs
	CN=CableLabs CVC Root CA
		Euro PacketCable
		C = BE
		O = tComLabs
		CN = tComLabs CVC Root CA

Intended Usage	This certificate is used to sign Code Verification CA Certificates. This certificate must be included in the S-MTA's nonvolatile memory at manufacture time.	
Signed By	Self-signed	
Validity Period	20+ years	
Modulus Length	2048	
Extensions	KeyUsage [c,m] (keyCertSign, cRL Sign) subjectkeyidentifier [n,m] basicConstraints [c,m](cA=true)	

CableLabs Code Verification CA Certificate

The CableLabs Code Verification CA Certificate must be validated as part of a certificate chain containing the CableLabs Code Verification Root CA Certificate, the CableLabs Code Verification CA Certificate, and the Code Verification Certificate. See Certificate Validation, page 26-20, for additional information on how to validate certificates. There may be more than one CableLabs Code Verification CA. An S-MTA must support one CableLabs CVC CA at a time.

Table 26-14 lists the values relevant to the CableLabs Code Verification CA Certificate.

Table 26-14 CableLabs Code Verification CA Certificates

CableLabs Code Verification CA Certificate		

Subject Name Form	**PacketCable**
	C=US
	O=CableLabs
	CN=CableLabs CVC CA
		Euro PacketCable
		C = BE
		O = tComLabs
		CN = tComLabs CVC CA

Intended Usage	This certificate is issued to CableLabs by the CableLabs Code Verification Root CA. This certificate issues Code Verification Certificates. This certificate must be included in the S-MTA's nonvolatile memory at manufacture time.	
Signed By	CableLabs Code Verification Root CA	
Validity Period	Set by CableLabs policy	

Table 26-14 CableLabs Code Verification CA Certificates (continued)

CableLabs Code Verification CA Certificate	
Modulus Length	2048
Extensions	KeyUsage [c,m] (keyCertSign, cRL Sign) subjectKeyIdentifier [n,m] authorityKeyIdentifier [n,m] basicConstraints [c,m](cA=true, pathLenConstraint=0)

Manufacturer Code Verification Certificate

The CableLabs Code Verification CA issues this certificate to each authorized Manufacturer. It is used in the policy set by the cable operator for secure software download.

Table 26-15 lists the values relevant to the Manufacturer Code Verification Certificate.

Table 26-15 Manufacturer Code Verification Certificates

Manufacturer Code Verification Certificate		

Subject Name Form	**PacketCable**
	C=*Country*
	O=*Company Name*
	[ST=*State/Province*]
	[L=*City*]
	CN=*Company Name* Mfg CVC
		Euro PacketCable
		C=*Country*
		O=*Company Name*
		[ST=*state/province*]
		[L=*City*]
		CN=*Company Name* Mfg CVC

Intended Usage	The CableLabs Code Verification CA issues this certificate to each authorized Manufacturer. It is used in the policy set by the cable operator for secure software download.	
Signed By	CableLabs Code Verification CA	tComLabs Code Verification CA Certificate
Validity Period	Set by CableLabs policy	
Modulus Length	1024, 1536, 2048	
Extensions	extendedKeyUsage [c,m] (id-kp-codeSigning) authorityKeyIdentifier [n,m]	

The Company Name in the Organization may be different than the Company Name in the Common Name.

Service Provider Code Verification Certificate

The Service Provider Code Verification Certificate must be validated as part of a certificate chain containing the CableLabs Code Verification Root CA Certificate, the CableLabs Code Verification CA Certificate, and the Service Provider Code Verification Certificate. See Certificate Validation, page 26-20, for additional information on how to validate certificates.

Table 26-16 lists the values relevant to the Service Provider Code Verification Certificate.

Table 26-16 Service Provider Code Verification Certificates

Service Provider Code Verification Certificate		

Subject Name Form	C=*Country*
	O=*Company Name*
	[ST=*State/Province*]
	[L=*City*]
	CN=*Company Name* Service Provider CVC
		C=*Country*
		O=*Company Name*
		[ST=*State/Province*]
		[L=*City*]
		CN=*Company Name* Service Provider CVC

Intended Usage	The CableLabs Code Verification CA issues this certificate to each authorized Service Provider. It is used in the policy set by the cable operator for secure software download.	
Signed By	CableLabs Code Verification CA	tComLabs Code Verification CA Certificate
Validity Period	Set by CableLabs policy	
Modulus Length	1024, 1536, 2048	
Extensions	extendedKeyUsage [c,m] (id-kp-codeSigning) authorityKeyIdentifier [n,m]	

The Company Name in the Organization may be different than the Company Name in the Common Name.

Certificate Revocation Lists for CVCs

The S-MTA is not required to support Certificate Revocation Lists (CRLs) for CVCs.

CHAPTER 27

Frequently Asked Questions

This chapter provides a list of frequently asked questions about Provisioning Prime Cable Provisioning Center.

This section lists answers to FAQs about Prime Cable Provisioning provisioning.

- Prime Cable Provisioning Configuration, page 27-1
- IPv6 Configuration, page 27-4
- CMTS Configuration, page 27-6

Prime Cable Provisioning Configuration

This section features FAQs related to general Prime Cable Provisioning configurations.

- How do I enable or disable Network Registrar extensions?
- How do I enable tracing for Network Registrar extensions?
- Why does the DPE server registration fails?
- Why does RDU crash while updating the agent.conf?
- Why are the components not been able to communicate?
- Why does connection drop between a Solaris DPE and Linux RDU
- Why does execution of a reliable batch fails for Radius-only user?
- Why does BAC 4.x and 4.x.x API clients get unrecognized batch ID?

How do I enable or disable Network Registrar extensions?

The procedures described in this section assume that:

- The Prime Cable Provisioning component is installed in */opt/CSCObac*.
- Cisco Prime Network Registrar is installed in */opt/nwreg2*.

To manually install Network Registrar extension points:

Step 1 Log into the Network Registrar server, with *root* access.

Step 2 Take a backup and copy the *libbprextensions.so* directory to the *NR_HOME/local/extensions/dhcp/dex/* directory.

Step 3 Take a backup and copy the *cnr_ep.properties* file to the *BPR_HOME/cnr_ep/conf* directory.

Step 4 Configure extensions from the Network Registrar command-line tool (**nrcmd**) using:

NR_HOME/**local/usrbin/nrcmd -s -b <** *BPR_HOME*/**cnr_ep/bin/bpr_cnr_enable_extpts.nrcmd**

To manually disable Network Registrar extension points:

Step 1 Log into the Network Registrar server, with *root* access.

Step 2 Enter:

NR_HOME/**local/usrbin/nrcmd -s -b <** *BPR_HOME*/**cnr_ep/bin/bpr_cnr_disable_extpts.nrcmd**

Step 3 Delete the *libbprextensions.so* file, which is located in the *NR_HOME/local/extensions/dhcp/dex/* directory.

How do I enable tracing for Network Registrar extensions?

To enable tracing for Network Registrar extension points:

Step 1 Log into the Network Registrar web UI. The default login and password are **admin** and **changeme**.

Step 2 From the menu, click **DHCP > DHCP Server** page.

The Manage DHCP Server page appears.

Step 3 Click the DHCP Server link.

The Edit DHCP Server page appears.

Step 4 Expand the Extensions category, and set the **extension-trace-level** value as 3 or 4.

Step 5 To view incoming and outgoing packets, expand the Logging category, and select the **incoming-packet-detail** and **outgoing-packet-detail** check boxes.

Step 6 Click **Modify Server**.

Step 7 Reload the DHCP server.

Why does the DPE server registration fails?

The registration of your DPE servers may be failing because the DPEs are not up to the requirements of the provisioning group.

Check the DPE log files for error messages that indicate that you must:

- Enable additional configuration, for example, if you must enable the TFTP service on the DPE.
- Upgrade the servers to enable features that are available only in this release of Prime Cable Provisioning.

Why does RDU crash while updating the agent.conf?

For RDU or DPE, you must not configure any extended JVM arguments through agent.conf.

Why are the components not been able to communicate?

The firewall must be disabled on the servers on which the Prime Cable Provisioningcomponents are installed. To know the ports that are being used by Prime Cable Provisioning, see Port Information from the Support Site.

Why does connection drop between a Solaris DPE and Linux RDU

Drop in connection between Solaris DPE and Linux RDU occurs when there is a huge regeneration of device configuration behind a single CM or when configuration regeneration for 1 million CM devices happens using CoS level change where a CRS job is kicked in.

Why does execution of a reliable batch fails for Radius-only user?

A reliable batch submitted by a Radius only user cannot be guaranteed to execute across reboots or when the user logs out. This applies to those users that do not have their privileges defined in an RDU account but are only provided by a remote Radius server.

Why does BAC 4.x and 4.x.x API clients get unrecognized batch ID?

In Prime Cable Provisioning 5.0, a few new command error messages as well as error codes are added that the API clients of earlier releases do not recognize and in such a case the API clients keep asking for the batch response to the RDU. But as the batch was already processed and the response was sent earlier, the RDU does not remember the batch response and hence responds with an error message from the second attempt onwards.

For successful communication between the Prime Cable Provisioning 5.0 RDU with 4.x and 4.x.x API clients, ensure that the 5.0 bpr.jar, bacbase.jar, and bac-common.jar files are copied to the 4.x and 4.x.x API client setup. These jars are loaded to the appropriate classpaths. We recommend that you use Java version 1.6.0_32 or later to support the API client in Prime Cable Provisioning.

Note If you are upgrading from 4.2.x to 5.0, ensure that all your API clients are upgraded to 4.2.1 before upgrading the RDU to 5.0. After upgrading all the RDUs to 5.0 upgrade all the API clients to 5.0.

Some of the scenarios in which the RDU can send new command error code are listed below.

- User does not have the specific privileges to execute a given command.
- Instance level authorization failure.
- User information (accessible privileges, accessible domains and active sessions) is not cached in the RDU. This can happen for a persistent API client with Prime Cable Provisioning jars prior to 4.2, as they do not re-authenticate while reconnecting to the RDU. Hence the user information is not cached in the RDU. The workaround to resolve this issue is to restart the API client after every RDU restart and the user must re-authenticate after the restart of the API client. To fix this issue, upgradethe API client jars of earlier releases to Prime Cable Provisioning 5.0 release.

Note The user information is stored in the RDU cache only when the user is authenticated. The user information remains in the cache until the last active user session expires or is terminated.

IPv6 Configuration

This section features FAQs related to IPv6 while configuring Prime Cable Provisioning.

- How do I enable provisioning in IPv6 for DPE?
- How do I configure an IPv4 interface for provisioning?
- DPE is configured for IPv6 provisioning, but Prime Cable Provisioning does not provision IPv6 DOCSIS 3.0 devices. Why?
- When searching for all devices using their MAC address, some IPv6 devices do not show up. Why?
- How do I enable IPv6 on an interface?
- How do I configure IPv6 on a loopback interface?
- How do I disable a stateful DHCPv6 client on Solaris 10?
- How do I assign a static IP address to an interface?

How do I enable provisioning in IPv6 for DPE?

To enable IPv6 provisioning for the DPE, complete this procedure from the DPE command line:

Step 1 For enabling IPv6 provisioning, you must configure two interfaces using the following commands:

a. To configure the DPE to use the specified interface, identified by its IP address, when communicating with Network Registrar extensions, enter:

interface ip *ip_address* **pg-communication**

ip_address—Identifies the IPv4 address of a specific DPE interface.

b. To configure the specified interface, identified by its IP address, to handle provisioning requests, enter:

interface ip *ip_address* **provisioning**

ip_address—Specifies the IP address of the interface in the IPv6 format.

Step 2 Enable these services using the respective commands:

- TFTP—**service tftp** *1..1* **ipv6 enabled true**
- ToD—**service tod** *1..1* **ipv6 enabled true**

Step 3 Reload the DPE using the **dpe reload** command.

How do I configure an IPv4 interface for provisioning?

To configure an IPv4 interface for provisioning, you must set the fully qualified domain name (FQDN) for that interface using this command:

```
# interface ip ip_address provisioning fqdn fqdn
```

- *ip_address*—Specifies the IP address of the interface in the IPv4 format.
- *fqdn*—Identifies the FQDN that is set on the specified interface.

DPE is configured for IPv6 provisioning, but Prime Cable Provisioning does not provision IPv6 DOCSIS 3.0 devices. Why?

You must enable DOCSIS 3.0 for the provisioning group to which the DPE belongs.

On the Prime Cable Provisioning administrator user interface:

Step 1 Choose **Servers > Provisioning Group**.

The Provisioning Group Details page appears.

Step 2 Click the Provisioning Groups link corresponding to the specific DPE.

Step 3 In the Capabilities Management area, click the **Enabled** radio button corresponding to IPv6 - DOCSIS 3.0.

Step 4 Click **Submit**.

When searching for all devices using their MAC address, some IPv6 devices do not show up. Why?

Some IPv6 devices do not appear following a search for all devices using the MAC address option because devices such as the Vista IPv6 computer do not report their MAC address in the Solicit message. As a result, they are known only by their DUID.

If a device reports its MAC address in the CableLabs Device ID option, then you can locate that device using its DUID or its MAC address.

How do I enable IPv6 on an interface?

To enable IPv6 on an interface, run the following commands:

```
# ifconfig intf inet6 plumb up
# /usr/lib/inet/in.ndpd
# touch /etc/hostname6.intf
```

where *intf* identifies the interface on which you want to enable IPv6.

How do I configure IPv6 on a loopback interface?

Before you configure IPv6 on a loopback interface, confirm if the loopback interface is up using this command:

```
# ifconfig -a
```

If the loopback interface is not up, log in as *root* and run the following commands:

```
# ifconfig lo0 inet6 plumb
# route add -inet6 ::1/128 localhost
# ifconfig lo0 inet6 up
```

How do I disable a stateful DHCPv6 client on Solaris 10?

To disable a stateful DHCPv6 client on Solaris 10, you must change the *ndpd.conf* file using these commands:

```
# cat > /etc/inet/ndpd.conf <<EOF
ifdefault StatefulAddrConf off
EOF
```

How do I assign a static IP address to an interface?

While assigning a static IP address is not essential, to do so, run this command:

```
# ifconfig bge0 inet6 addif 2001:420:3800:601::1/64 up
```

CMTS Configuration

This section describes some FAQs related to configuring a cable modem termination system (CMTS):

- How do I know that both cable line cards are using the cable bundle 1?
- Is there an IPv6 cable-helper address that I can use?
- How do I configure multiple IPv6 subnets similar to IPv4 primary and secondary IPv4 subnets?
- How do I view the list of IPv6 modems on the CMTS?
- How do I configure a CMTS interface to accept only IPv6 single stack?
- What does the modem state init(x) mean?

How do I know that both cable line cards are using the cable bundle 1?

You must add this setting for each cable interface:

```
interface Cable3/0
 cable bundle 1
```

Is there an IPv6 cable-helper address that I can use?

Yes, this setting on the bundle is equivalent to the helper-address in IPv4:

```
    ipv6 dhcp relay destination FC00:420:3800:710::2 GigabitEthernet0/1
```

How do I configure multiple IPv6 subnets similar to IPv4 primary and secondary IPv4 subnets?

While you can assign multiple prefixes to a bundle for IPv6, there are no primary or secondary types for these subnets in IPv6.

How do I view the list of IPv6 modems on the CMTS?

Use the following command to see the list of IPv6 modems:

```
show cable modem ipv6
```

How do I configure a CMTS interface to accept only IPv6 single stack?

You must add this option to the interface of the cable modem termination system (CMTS):

```
(config-if)# cable ip-init ipv6
```

What does the modem state init(x) mean?

The **show cable modems** (scm) command displays the connected cable modems and their respective states.

Table 27-1 lists the various modem states in both IPv4 and IPv6.

Table 27-1 Cable Modem States

State	Description
IPv4	
init(d)	DHCP Discover
init(io)	DHCP Offer
init(dr)	DHCP Request
init(i)	DHCP Ack
init(o)	TFTP Request

Table 27-1 Cable Modem States (continued)

State	Description
Init(t)	ToD Request
online	Online
IPv6	
init6(s)	Solicit
init6(a)	Advertise
init6(r)	Request
init6(i)	Reply
init6(o)	IPv6 TFTP Request
init6(t)	IPv6 ToD request
online	Online

CHAPTER 28

Prime Cable Provisioning Support Tools

This chapter contains information on, and explains the use of tools that help you maintain Prime Cable Provisioning as well as speed and improve the installation, deployment, and use of this product.

This chapter discusses:

Note This section contains several examples of tool use. In many cases, the tool filenames include a path specified as *BPR_HOME*. This indicates the default home directory location.

Prime Cable Provisioning Tools

Prime Cable Provisioning provides automated tools that you use to perform certain functions more efficiently. Table 28-1 lists the various tools that this Prime Cable Provisioning release supports.

Table 28-1 Prime Cable Provisioning Tools

Tool	Description	Refer...
Configuration File Utility	Used to test, validate, and view Prime Cable Provisioning template and configuration files.	Using Configuration File Utility for Template, page 19-33
Prime Cable Provisioning Process Watchdog	Interacts with the Prime Cable Provisioning watchdog daemon to observe the status of the Prime Cable Provisioning system components, and stop or start servers.	Using Prime Cable Provisioning Process Watchdog from CLI, page 23-2
RDU Log Level Tool	Sets the log level of the RDU, and enables or disables debugging log output.	Using the RDU Log Level Tool, page 25-5
PacketCable Certificates Tool	Installs, and manages, the KDC certificates that are required by the KDC for its operation.	Using PKCert.sh, page 28-3
KeyGen Tool	Generates PacketCable service keys.	Using KeyGen Tool, page 28-9
Changing Network Registrar Properties Tool	Used to change key configuration and SSL related properties used by Prime Cable Provisioning extensions that are incorporated into the Prime Network Registrar DHCP server.	Using changeNRProperties.sh, page 28-11
SNMP Agent Configuration Tool	Manages the SNMP agent.	Using snmpAgentCfgUtil.sh Tool, page 22-3
Diagnostics Tool	Collects server data related to system performance and troubleshooting.	Troubleshooting Using Diagnostics Tool, page 26-5
BundleState.sh Tool	Bundles diagnostics data related to server state for support escalations.	Bundling Server State for Support, page 26-10

Table 28-1 Prime Cable Provisioning Tools (continued)

Tool	Description	Refer...
Disk Space Monitoring Tool	Sets threshold values for one or more file systems. When these thresholds are surpassed, an alert is generated until additional disk space is available.	Using disk_monitor.sh, page 28-13
changeSSLProperties.sh Tool	Used to change various SSL properties like enable or disable SSL connection, change secure key, secure port number, secret key and key password.	Once you enable encryption, Prime Cable Provisioning will not be able to support the 4.0 and 4.0.x API clients., page 28-16
ws-cli.sh Tool	Used to change key PWS configuration properties like adding and deleting RDU accounts, changing the log severity level.	Using ws-cli.sh, page 28-19

Using PKCert.sh

The PKCert tool creates the KDC certificate and its corresponding private key. It also allows you to verify certificate chains and copy and rename a certificate chain to the names required by the KDC.

Note This tool is available only when the KDC component is installed.

Running PKCert Tool

Run the PKCert tool by executing the PKCert.sh command, which resides by default in the *BPR_HOME/kdc* directory.

Syntax Description

PKCert.sh *function option*

- *function*—Identifies the function to be performed. You can choose:
  - **-c**—Creates a KDC certificate. See Creating a KDC Certificate, page 28-4.
  - **-v**—Verifies and normalizes the PacketCable certificate set. See Validating KDC Certificates, page 28-5.
  - **-z**—Sets the log level for debug output that is stored in the *pkcert.log* file. See Setting Log Level for Debug Output, page 28-6.

Note If you have trouble using these options, specify **-?** to display available help information.

- *option*—Implements optional functions, depending on the function you selected.

Creating a KDC Certificate

To create the KDC certificate:

Step 1 Change directory to */opt/CSCObac/kdc*.

Step 2 Run the PKCert.sh tool using this syntax:

PKCert.sh -s *dir* **-d** *dir* **-c** *cert* **-e -r** *realm* **-a** *name* **-k** *keyFile* [**-n** *serial#*] [**-o**]

- **-s** *dir*—Specifies the source directory
- **-d** *dir*—Specifies the destination directory
- **-c** *cert*—Uses the service provider certificate (DER encoded)
- **-e**—Identifies the certificate as a Euro-PacketCable certificate
- **-r** *realm*—Specifies the Kerberos realm for the KDC certificate
- **-a** *name*—Specifies the DNS name of the KDC
- **-k** *keyFile*—Uses the service provider private key (DER encoded)
- **-n** *serial#*—Sets the certificate serial number
- **-o**—Overwrites existing files

When a new certificate is created and installed, the new certificate identifies the realm in the subject alternate name field. The new certificate is unique to its current environment in that it contains the:

- KDC realm.
- DNS name associated with this KDC that the Multimedia Terminal Adapter (MTA) will use.

Examples

```
# ./PKCert.sh  -c "-s . -d /opt/CSCObac/kdc/<Operating System>/packetcable/certificates
-k CLCerts/Test_LSCA_privkey.der -c CLCerts/Test_LSCA.cer -r PCTEST.CISCO.COM -n 100
-a kdc.pctest.cisco.com -o"
Pkcert Version 1.0
Logging to pkcert.log
Source Directory: .
Destination Directory: /opt/CSCObac/kdc/<Operating System>/packetcable/certificates
Private Key File: CLCerts/Test_LSCA_privkey.der
Certificate File: CLCerts/Test_LSCA.cer
Realm: PCTEST.CISCO.COM
Serial Number: 100
DNS Name of KDC: kdc.pctest.cisco.com
WARNING - Certificate File will be overwritten
SP Cert subject name: C=US,O=CableLabs\, Inc.,OU=ABC Cable Company,CN=Shared-01 CableLabs
Local System CA
File written: /opt/CSCObac/kdc/<Operating
System>/packetcable/certificates/KDC_private_key.pkcs8
File written: /opt/CSCObac/kdc/<Operating
System>/packetcable/certificates/KDC_private_key_proprietary.
File written: /opt/CSCObac/kdc/<Operating
System>/packetcable/certificates/KDC_PublicKey.der
File written: /opt/CSCObac/kdc/<Operating System>/packetcable/certificates/KDC.cer
```

```
KDC Certificate Successfully Created at /opt/CSCObac/kdc/<Operating
System>/packetcable/certificates/KDC.cer
```

This command creates the following files:

- */opt/CSCObac/kdc/<Operating System>/packetcable/certificates/KDC.cer*
- */opt/CSCObac/kdc/<Operating System>/packetcable/certificates/KDC_private_key.pkcs8*.

The KDC certificate will have a realm set to PCTEST.CISCO.COM, a serial number set to 100, and the fully qualified domain name (FQDN) of the KDC server set to kdc.pctest.cisco.com.

Validating KDC Certificates

This command examines all files in the source directory specified and attempts to identify them as X.509 certificates. If legitimate X.509 certificates are found, the files are properly renamed and copied to the destination directory. An error is generated when more than one legitimate chain of certificates for a particular purpose (service provider or device) is identified. If this occurs, you must remove the extra certificate from the source directory and run the command again.

Note When you enter the **PKCert.sh -v -?** command, usage instructions for validating KDC certificates by using the PKCert tool appear.

To validate the KDC certificate:

Step 1 Change directory to */opt/CSCObac/kdc*.

Step 2 Run the PKCert.sh tool using this syntax:

PKCert.sh -v -s *dir* **-d** *dir* **-r** *dir* **-e**

- **-s** *dir*—Specifies the source directory
- **-d** *dir*—Specifies the destination directory
- **-o**—Overwrites any existing files
- **-r** *dir*—Specifies the reference certificate directory
- **-e**—Identifies the certificate as a Euro-PacketCable certificate

Verification is performed against reference certificates built into this package. If you specify the **-d** option, the certificates are installed in the target directory with name normalization.

Examples

```
# ./PKCert.sh  -v "-s /opt/CSCObac/kdc/TestCerts -d /opt/CSCObac/kdc/<Operating
System>/packetcable/certificates -o"
Pkcert Version 1.0
Logging to pkcert.log
Output files will overwrite existing files in destination directory

Cert Chain(0)    Chain Type: Service Provider
[Local File]                          [Certificate Label]                    [PacketCable
Name]
CableLabs_Service_Provider_Root.cer   CableLabs_Service_Provider_Root.cer
Service_Provider.cer                  Service_Provider.cer
Local_System.cer                      Local_System.cer
```

```
KDC.cer                              KDC.cer

Cert Chain(1)    Chain Type: Device
[Local File]                         [Certificate Label]                    [PacketCable
Name]
MTA_Root.cer                         MTA_Root.cer
File written: /opt/CSCObac/kdc/<Operating
System>/packetcable/certificates/CableLabs_Service_Provider_Root.cer
File written: /opt/CSCObac/kdc/<Operating
System>/packetcable/certificates/Service_Provider.cer
File written: /opt/CSCObac/kdc/<Operating
System>/packetcable/certificates/Local_System.cer
File written: /opt/CSCObac/kdc/<Operating System>/packetcable/certificates/KDC.cer

Service Provider Certificate Chain Written to Destination Directory
/opt/CSCObac/kdc/<Operating System>/packetcable/certificates


File written: /opt/CSCObac/kdc/<Operating System>/packetcable/certificates/MTA_Root.cer

Device Certificate Chain Written to Destination Directory /opt/CSCObac/kdc/<Operating
System>/packetcable/certificates
```

Setting Log Level for Debug Output

This command enables you to set the log level for debug output that is logged in *pkcert.log,* which resides in *BPR_HOME/kdc*. You can use the data in the log file to troubleshoot any problems that may have occurred while performing the requested tasks.

To set the log level for debug output:

Step 1 Change directory to */opt/CSCObac/kdc*.

Step 2 Run the PKCert.sh tool using this syntax:

PKCert.sh -s *dir* **-d** *dir* **-k** *keyFile* **-c** *cert* **-r** *realm* **-a** *name* **-n** *serial#* **-o** {**-z error | info | debug}**

- **-s** *dir*—Specifies the source directory
- **-d** *dir*—Specifies the destination directory
- **-k** *keyFile*—Uses the service provider private key (DER encoded)
- **-c** *cert*—Uses the service provider certificate (DER encoded)
- **-r** *realm*—Specifies the Kerberos realm for the KDC certificate
- **-a** *name*—Specifies the DNS name of the KDC
- **-n** *serial#*—Sets the certificate serial number
- **-o**—Overwrites existing files
- **-z**—Sets the log level for debug output that is stored in the *pkcert.log* file. The values you can choose are:
  - **error**—Specifies the logging of error messages.
  - **info**—Specifies the logging of informational messages.
  - **debug**—Specifies the logging of debug messages. This is the default setting.

Examples

Example 1

In this example, the log level is set for collecting error messages.

```
# ./PKCert.sh  -c "-s /var/certsInput -d /var/certsOutput -k
/var/certsInput/Local_System.der -c /var/certsInput/Local_System.cer -r PCTEST.CISCO.COM
-n 100 -a kdc.pctest.cisco.com -o -z error"
Pkcert Version 1.0
Logging to pkcert.log
Source Directory: /var/certsInput
Destination Directory: /var/certsOutput
Private Key File: /var/certsInput/Local_System.der
Certificate File: /var/certsInput/Local_System.cer
Realm: PCTEST.CISCO.COM
Serial Number: 100
DNS Name of KDC: kdc.pctest.cisco.com
Setting debug to error
WARNING - Certificate File will be overwritten
SP Cert subject name: C=US,O=CableLabs\, Inc.,OU=ABC Cable Company,CN=Shared-01 CableLabs
Local System CA
File written: /var/certsOutput/KDC_private_key.pkcs8
File written: /var/certsOutput/KDC_private_key_proprietary.
File written: /var/certsOutput/KDC_PublicKey.der
File written: /var/certsOutput/KDC.cer
KDC Certificate Successfully Created at /var/certsOutput/KDC.cer

Copy KDC.cer to the KDC certificate directory (i.e. /opt/CSCObac/kdc/<Operating
System>/packetcable/certificates)
Copy KDC_private_key.pkcs8 to the KDC platform directory (i.e. /opt/CSCObac/kdc/solaris)
Copy KDC_private_key_proprietary. to the KDC platform directory (i.e.
/opt/CSCObac/kdc/solaris)
```

Example 2

In this example, the log level is set for collecting information messages.

```
# ./PKCert.sh -c "-s /var/certsInput
> -d /var/certsOutput
> -k /var/certsInput/Local_System.der
> -c /var/certsInput/Local_System.cer
> -r PCTEST.CISCO.COM
> -n 100
> -a kdc.pctest.cisco.com
> -o -z info"
 INFO [main] 2007-05-02 06:32:26,280 (PKCert.java:97) - Pkcert Version 1.0
Pkcert Version 1.0
Logging to pkcert.log
Source Directory: /var/certsInput
Destination Directory: /var/certsOutput
Private Key File: /var/certsInput/Local_System.der
Certificate File: /var/certsInput/Local_System.cer
Realm: PCTEST.CISCO.COM
Serial Number: 100
DNS Name of KDC: kdc.pctest.cisco.com
Setting debug to info
 INFO [main] 2007-05-02 06:32:26,289 (PKCCreate.java:69) - PKCCreate startup
WARNING - Certificate File will be overwritten
 INFO [main] 2007-05-02 06:32:26,291 (PKCCreate.java:341) - WARNING - Certificate File
will be overwritten
SP Cert subject name: C=US,O=CableLabs\, Inc.,OU=ABC Cable Company,CN=Shared-01 CableLabs
Local System CA
File written: /var/certsOutput/KDC_private_key.pkcs8
File written: /var/certsOutput/KDC_private_key_proprietary.
File written: /var/certsOutput/KDC_PublicKey.der
File written: /var/certsOutput/KDC.cer
```

```
KDC Certificate Successfully Created at /var/certsOutput/KDC.cer

Copy KDC.cer to the KDC certificate directory (i.e. /opt/CSCObac/kdc/<Operating
System>/packetcable/certificates)
Copy KDC_private_key.pkcs8 to the KDC platform directory (i.e. /opt/CSCObac/kdc/solaris)
Copy KDC_private_key_proprietary. to the KDC platform directory (i.e.
/opt/CSCObac/kdc/solaris)
```

Example 3

In this example, the log level is set for debugging.

Note The sample output has been trimmed for demonstration purposes.

```
# ./PKCert.sh -c "-s /var/certsInput -d /var/certsOutput -k
/var/certsInput/Local_System.der -c /var/certsInput/Local_System.cer -r PCTEST.CISCO.COM
-n 100 -a kdc.pctest.cisco.com -o -z debug"
 INFO [main] 2007-05-02 06:32:06,029 (PKCert.java:97) - Pkcert Version 1.0
Pkcert Version 1.0
Logging to pkcert.log
Source Directory: /var/certsInput
Destination Directory: /var/certsOutput
Private Key File: /var/certsInput/Local_System.der
Certificate File: /var/certsInput/Local_System.cer
Realm: IPFONIX.COM
Serial Number: 100
DNS Name of KDC: bacdev3-dpe-4.cisco.com
Setting debug to debug
 INFO [main] 2007-05-02 06:32:06,038 (PKCCreate.java:69) - PKCCreate startup
WARNING - Certificate File will be overwritten
 INFO [main] 2007-05-02 06:32:06,039 (PKCCreate.java:341) - WARNING - Certificate File
will be overwritten
DEBUG [main] 2007-05-02 06:32:06,054 (PKCert.java:553) - Characters Read: 1218
DEBUG [main] 2007-05-02 06:32:06,056 (PKCert.java:583) - Binary File:
/var/certsInput/Local_System.der Read. Length: 1218
DEBUG [main] 2007-05-02 06:32:06,062 (PKCert.java:553) - Characters Read: 943
DEBUG [main] 2007-05-02 06:32:06,063 (PKCert.java:583) - Binary File:
/var/certsInput/Local_System.cer Read. Length: 943
DEBUG [main] 2007-05-02 06:32:06,064 (PKCert.java:455) - Jar File Path:
/opt/CSCObac/lib/pkcerts.jar
DEBUG [main] 2007-05-02 06:32:06,065 (PKCert.java:456) - Opened jar file:
/opt/CSCObac/lib/pkcerts.jar
DEBUG [main] 2007-05-02 06:32:06,067 (PKCert.java:460) - Jar entry unfiltered:
Tag_Packetcable_Tag/
DEBUG [main] 2007-05-02 06:32:06,068 (PKCert.java:460) - Jar entry unfiltered:
Tag_Packetcable_Tag/CableLabs_Service_Provider_Root.cer
...
DEBUG [main] 2007-05-02 06:32:06,115 (PKCert.java:472) - File:
Tag_Packetcable_Tag/Manu.cer
DEBUG [main] 2007-05-02 06:32:06,116 (PKCert.java:472) - File:
Tag_Packetcable_Tag/Service_Provider.cer
DEBUG [main] 2007-05-02 06:32:06,121 (PKCCreate.java:91) - Found 7 files in jar.
DEBUG [main] 2007-05-02 06:32:06,827 (KDCCert.java:98) - SP Cert subject name:
C=US,O=CableLabs\, Inc.,OU=ABC Cable Company,CN=Shared-01 CableLabs Local System CA
SP Cert subject name: C=US,O=CableLabs\, Inc.,OU=ABC Cable Company,CN=Shared-01 CableLabs
Local System CA
DEBUG [main] 2007-05-02 06:32:07,687 (KDCCert.java:293) - Setting issuer to:
C=US,O=CableLabs\, Inc.,OU=ABC Cable Company,CN=Shared-01 CableLabs Local System CA
DEBUG [main] 2007-05-02 06:32:07,699 (KDCCert.java:231) - DERVisibleToGeneral
org.bouncycastle.asn1.DERGeneralString@bd0b4ea6

DEBUG [main] 2007-05-02 06:32:07,700 (KDCCert.java:231) - DERVisibleToGeneral
```

```
org.bouncycastle.asn1.DERGeneralString@5035bc0

DEBUG [main] 2007-05-02 06:32:07,701 (KDCCert.java:231) - DERVisibleToGeneral
org.bouncycastle.asn1.DERGeneralString@5035bc0

DEBUG [main] 2007-05-02 06:32:07,703 (KDCCert.java:210) - DERCombineTagged [0] IMPLICIT
    DER ConstructedSequence
        ObjectIdentifier(1.3.6.1.5.2.2)
        Tagged [0]
            DER ConstructedSequence
                Tagged [0]
                    org.bouncycastle.asn1.DERGeneralString@5035bc0
                Tagged [1]
                    DER ConstructedSequence
                        Tagged [0]
                            Integer(2)
                        Tagged [1]
                            DER ConstructedSequence
                                org.bouncycastle.asn1.DERGeneralString@bd0b4ea6
                                org.bouncycastle.asn1.DERGeneralString@5035bc0

File written: /var/certsOutput/KDC_private_key.pkcs8
File written: /var/certsOutput/KDC_private_key_proprietary.
File written: /var/certsOutput/KDC_PublicKey.der
File written: /var/certsOutput/KDC.cer
KDC Certificate Successfully Created at /var/certsOutput/KDC.cer

Copy KDC.cer to the KDC certificate directory (i.e. /opt/CSCObac/kdc/<Operating
System>/packetcable/certificates)
Copy KDC_private_key.pkcs8 to the KDC platform directory (i.e. /opt/CSCObac/kdc/solaris)
Copy KDC_private_key_proprietary. to the KDC platform directory (i.e.
/opt/CSCObac/kdc/solaris)
```

Using KeyGen Tool

The KeyGen tool is used to generate PacketCable service keys. The service keys are symmetric triple data encryption standard (triple DES or 3DES) keys (shared secret) required for KDC communication. The KDC server requires service keys for each of the provisioning FQDNs of the DPE. Any changes made to the DPE provisioning FQDN from the DPE command-line interface (CLI) requires a corresponding change to the KDC service key filename. This change is necessary because the KDC service key uses the DPE provisioning FQDN as part of its filename.

The KeyGen tool, which resides in the *BPR_HOME/kdc* directory, uses command-line arguments for the DPE provisioning FQDN, realm name, and a password, and generates the service key files.

Note When running this tool, remember to enter the same password that you used to generate the service key on the DPE (by using the **service packetCable** *1..1* **registration kdc-service-key** command from the DPE CLI). For information on setting this password, see the *Cisco Prime Cable Provisioning 5.0 DPE CLI Reference Guide*.

The KDC server reads the service keys on startup. Any modification to the service keys requires that you restart the KDC server.

Syntax Description

keygen *options fqdn realm password*

- *options* are:
  - **-?**—Displays this usage message and exits the command.
  - **-v** or **-version**—Displays the version of this tool and exits the command.
  - **-q** or **-quiet**—Implements a quiet mode whereby no output is created.
  - **-c** or **-cms**—Creates a service key for the CMS system.
- *fqdn*—Identifies the FQDN of the DPE and is a required entry.
- *realm*—Identifies the Kerberos realm and is a required entry.
- *password*—Specifies the password to be used. This is also a required field. The password must be from 6 to 20 characters.

Three service key files are written in the KDC keys directory using this filename syntax:

mtafqdnmap,*fqdn@REALM*

mtaprovsrvr,*fqdn@REALM*

krbtgt,*REALM@REALM*

- *fqdn*—Identifies the FQDN of the DPE.
- *REALM*—Identifies the Kerberos realm.

The service key file always contains a version field of 0x0000.

Examples

```
# keygen dpe.cisco.com CISCO.COM changeme
```

When this command is implemented, these KDC service keys are written to the *BPR_HOME/kdc/<Operating System>/keys* directory:

```
mtafqdnmap,dpe.cisco.com@CISCO.COM
mtaprovsrvr,dpe.cisco.com@CISCO.COM
krbtgt,CISCO.COM@CISCO.COM
```

Restart the KDC, so that the new keys are recognized. Use this Prime Cable Provisioning process watchdog command to restart the KDC:

```
# /etc/init.d/bprAgent restart kdc
```

This example illustrates the generation of a CMS service key:

```
# keygen -c cms-fqdn.com CMS-REALM-NAME changeme
```

When this command is implemented, this CMS service key is written to the *BPR_HOME/kdc/<Operating System>/keys* directory.

```
cms,cms-fqdn.com@CMS-REALM-NAME
```

Verifying the KDC Service Keys

Once you generate the service keys on the KDC and the DPE, verify if the service keys match on both components.

The KeyGen tool requires you to enter the same password that you used to generate the service key on the DPE using the **service packetCable** *1..1* **registration kdc-service-key** command. Once you set this password on the DPE, you can view the service key from the *dpe.properties* file, which resides in the *BPR_HOME/dpe/conf* directory. Look for the value against the */pktcbl/regsvr/KDCServiceKey*= property.

For example:

```
# more dpe.properties
/pktcbl/regsvr/KDCServiceKey=2e:d5:ef:e9:5a:4e:d7:06:67:dc:65:ac:bb:89:e3:2c:bb:
71:5f:22:bf:94:cf:2c
```

Note The output of this example has been trimmed for demonstration purposes.

To view the service key generated on the KDC, run the following command from the *BPR_HOME/kdc/<Operating System>/keys* directory:

od -Ax -tx1 mtaprovsrvr,*fqdn@REALM*

- *fqdn*—Identifies the FQDN of the DPE.
- *REALM*—Identifies the Kerberos realm.

The output that this command generates should match the value of the */pktcbl/regsvr/KDCServiceKey*= property in the *dpe.properties* file.

For example:

```
# od -Ax -tx1 mtaprovsrvr,dpe.cisco.com@CISCO.COM
0000000 00 00 2e d5 ef e9 5a 4e d7 06 67 dc 65 ac bb 89
0000010 e3 2c bb 71 5f 22 bf 94 cf 2c
000001a
```

In the examples shown here, note that the service key generated at the KDC matches the service key on the DPE.

Using changeNRProperties.sh

The Prime Cable Provisioning installation program establishes values for configuration properties used by Prime Cable Provisioning extensions that are incorporated into the Network Registrar DHCP server. You use the **changeNRProperties.sh** command, which is found in the *BPR_HOME/cnr_ep/bin* directory, to change key configuration properties.

Invoking the script without any parameters displays a help message listing the properties that can be set.

To run this command:

Step 1 Change directory to *BPR_HOME/cnr_ep/bin*.

Step 2 Run the **changeNRProperties.sh** command using this syntax:

changeNRProperties.sh *options*

Where *options* are:

- **-help**—Displays this help message. The **-help** option must be used exclusively. Do not use this with any other option.

- **-ep enabled | disabled**—Enables or disables the PacketCable property. Enter **-ep enabled** to enable the property, and **-ep disabled** to disable it.
- **-ec enabled | disabled**—Enables or disables the CableHome property. Enter **-ec enabled** to enable the property, and **-ec disabled** to disable it.
- **-d**—Displays the current properties. The **-d** option must be used exclusively. Do not use this with any other option.
- **-s** *secret*—Identifies the Prime Cable Provisioning shared secret. For example, if the shared secret is the word *secret*, enter **-s secret**.
- **-f** *fqdn*—Identifies the RDU FQDN. For example, if you use rdu.example.com as the fully qualified domain name, enter **-f rdu.example.com**.
- **-p** *port*—Identifies the RDU port you want to use. For example, if you want to use port number 49187, enter **-p 49187**.
- **-r** *realm*—Identifies the PacketCable realm. For example, if your PacketCable realm is EXAMPLE.COM, enter **-r EXAMPLE.COM**.

  Note You must enter the realm in uppercase letters.

- **-g** *prov_group*—Identifies the provisioning group. For example, if you are using provisioning group called group1, enter **-g group1**.
- **-t 00 | 01**—Identifies whether or not the PacketCable TGT is set to off or on. For example, to set the TGT to off, enter **-t 00;** to set this to on, enter **-t 01**.
- **-a** *ip*—Identifies the PacketCable primary DHCP server address. For example, if the IP address of your primary DHCP server is 10.10.10.2, enter **-a 10.10.10.2**.
- **-b** *ip*—Identifies the PacketCable secondary DHCP server address. For example, if the IP address of your secondary DHCP server is 10.10.10.4, enter **-b 10.10.10.4**. You can also enter **-b null** to set a null value, if appropriate.
- **-y** *ip*—Identifies the PacketCable primary DNS server address. For example, if the IP address of the PacketCable primary DNS server is 10.10.10.6, enter **-y 10.10.10.6**.
- **-z** *ip*—Identifies the PacketCable secondary DNS server address. For example, if the IP address of your secondary DNS server is 10.10.10.8, enter **-z 10.10.10.8**. You can also enter **-z null** to set a null value, if appropriate.
- **-o** *prov_ip man_ip*—Sets the management address to use for communication with the DPE identified by the given provisioning address. For example, if the IP address of your provisioning group is 10.10.10.7, enter **-o 10.10.10.7 10.14.0.4**. You can also enter a null value, if appropriate; for example, **-o 10.10.10.7 null**.
- **-ssl**—Enables or disables CNR-EP secure mode of communication with the RDU.
- **-ckl**—Sets the rootCA.pem certificate location. By default, the certificate is stored in the *BPR_HOME/lib/security* directory.
- **-ckp**—Changes the keystore password.
- **-sk** *secretkey*—Updates the secret key which is configured during installation and is used with shared secret for communication.

Step 3 Restart the DHCP server.

Examples

This is an example of changing the Network Registrar extensions by using the NR Extensions Properties tool:

```
# /opt/CSCObac/cnr_ep_bin/changeNRProperties.sh -g primary1
Current NR Properties:
RDU Port: 49187
RDU FQDN: rdu.example.com
Provisioning Group: primary1
Shared Secret: fggTaLg0XwKRs
PacketCable Enable: enabled
CableLabs client TGT: 01
CableLabs client Realm: EXAMPLE.COM
CableLabs client Primary DHCP Server: 10.10.1.2
CableLabs client Secondary DHCP Server: NOT SET
CableLabs client Primary DNS Server: 10.10.1.2
CableLabs client Secondary DNS Server: NOT SET
```

Note You must restart your NR DHCP server for the changes to take effect.

This is an example of viewing the current properties:

```
# /opt/CSCObac/cnr_ep_bin/changeNRProperties.sh -d
Current NR Properties:
RDU Port: 49187
RDU FQDN: rdu.example.com
Provisioning Group: primary1
Shared Secret: fggTaLg0XwKRs
PacketCable Enable: enabled
CableLabs client TGT: 01
CableLabs client Realm: EXAMPLE.COM
CableLabs client Primary DHCP Server: 10.10.1.2
CableLabs client Secondary DHCP Server: NOT SET
CableLabs client Primary DNS Server: 10.10.1.2
CableLabs client Secondary DNS Server: NOT SET
```

Using disk_monitor.sh

Monitoring available disk space is an important system administration task. You can use a number of custom written scripts or commercially available tools to do so.

The **disk_monitor.sh** command, which resides in the *BPR_HOME/rdu/samples/tools* directory, sets threshold values for one or more file systems. When these thresholds are surpassed, an alert is generated through the Solaris syslog facility, at 60-second intervals, until additional disk space is available.

Note We recommend that, at a minimum, you use the **disk_monitor.sh** script to monitor the *BPR_DATA* and *BPR_DBLOG* directories.

Syntax Description

disk_monitor.sh *filesystem-directory x [filesystem-directory* x*]*

- *filesystem-directory*—Identifies any directory in a file system to monitor.
- *x*—Identifies the percentage threshold applied to the specified file system.
- *filesystem-directory**—Identifies multiple file systems.
- *x**—Specifies percentage thresholds to be applied to multiple file systems.

Examples

Example 1

This example specifies that a notification be sent out when the */var/CSCObac* file system reaches 80 percent of its capacity.

```
# ./disk_monitor.sh /var/CSCObac 80
```

When the database logs disk space reaches 80-percent capacity, an alert similar to the following one is sent to the syslog file:

```
Dec 7 8:16:06 perf-u80-1 BPR: [ID 702911 local6.warning] File system /var/bpr usage is 81%
(threshold is 80%)
```

Example 2

This example describes how you can run the disk_monitor.sh tool as a background process. Specifying an ampersand (&) at the end of the command immediately returns output while running the process in the background.

```
# ./disk_monitor.sh /var/CSCObac 80 &
1020
```

Using admin.properties

Before you use the administrator user interface, examine the *adminui.properties* file. This file contains a variety of controls that specify the behavior of the interface.

You can open this file using any text editor, and change its content to perform the functions that you want. After you save the changes, restart the user interface so that the changes take effect.

To start the administrator user interface, enter:

```
# /etc/init.d/bprAgent start tomcat
```

To stop the administrator user interface, enter:

```
# /etc/init.d/bprAgent stop tomcat
```

To restart the administrator user interface, enter:

```
# /etc/init.d/bprAgent restart tomcat
```

You can configure the user interface by using the options available in the *adminui.properties* file. These options are controlled by Prime Cable Provisioning settings or defined in the *adminui.properties* file in the *BPR_HOME/rdu/conf* directory. The configuration parameters are:

- `/adminui/port`—Specifies the listening port of the RDU. The default port number is 49187.
- `/adminui/fqdn`—Specifies the fully qualified domain name of the host on which the RDU is running. The default value is the FQDN of the host; for example, bac_test.EXAMPLE.COM.
- `/adminui/maxReturned`—Specifies the maximum number of search results. You can set this value to a maximum of 5000. The default value is 1000.
- `/adminui/maxDetailsReturned` —Specifies the maximum number of search results when search for detailed information is requested. You can set this value to a maximum of 1000 which is also the default value.

Note If the memory of the deployed server is having a smaller heap size, then the maxReturned and maxDetailsReturned will become half of its values.

For example, if the value of maxReturned is set to 5000, it will retrieve only 2500.

- `/adminui/pageSize`—Specifies the number of search results displayed per page. You can set this number at 25, 50, or 75. The default value is 25.
- `/adminui/refresh`—Specifies if the refresh function is enabled or disabled. This option is, by default, disabled.
- `/adminui/extensions`—Specifies if the use of extensions in Cisco Prime Cable Provisioning is enabled or disabled. You use extensions to augment Cisco Prime Cable Provisioning behavior or add support for new device technologies. The use of extensions is, by default, enabled.
- `/adminui/maxFileSize`—Specifies the maximum size of a file uploaded to Cisco Prime Cable Provisioning. The default file size is 200 MB.
- `/adminui/refreshRate`—Specifies the duration (in seconds) after which a screen is refreshed. The default value is 90 seconds. Before setting a value for this option, ensure that the `/adminui/refresh` option is enabled.
- `/adminui/file/extensions`—Specifies the extensions of the files that the user interface supports. The supported extensions are by default *.bin*, *.cm*, and *.jar*.
- `/adminui/timeout`—Specifies the length of time after which an idle session times out. The default period is set as 10 minutes. In case of any value lesser than 10 minutes, the idle session time out still happens after 10 minutes.
- `/adminui/noOfLines`—Specifies the last number of lines from *rdu.log* or *dpe.log* that appear on the user interface. The default number of lines that appear is 250.
- `/adminui/redirectToHttps`—Specifies whether the Admin UI should be in HHTPS mode or not. The default is true.
- `/adminui/enableDomainAdministration`—Specifies whether Security Domain(RBAC) can be assigned to various entities. If set to true, the Instance Level Authorization check box is shown in the RDU Defaults page. The default value is false.

Example 28-1 Sample adminui.properties File

```
/adminui/port=49187
/adminui/fqdn=doc.example.com
/adminui/maxReturned=5000
/adminui/pageSize=25
/adminui/refresh=disabled
/adminui/extensions=enabled
/adminui/maxFileSize=20000000
/adminui/refreshRate=90
/adminui/file/extensions=.bin,.cm,.jar
/adminui/timeout=10
/adminui/noOfLines=250
/adminui/redirectToHttps=false
```

Note By default, Prime Cable Provisioning redirects all HTTP communications over HTTPS. If you want to bypass the HTTPS redirection, set the property adminui/redirectToHttps to false in the admin.properties file.

Using passwordEncryption.sh

The password encryption tool, passwordEncryption.sh allows you to enable password encryption using SHA1. This tool is available under BPR_HOME/rdu/bin. By default, SHA1 encryption is enabled for fresh installation of Prime Cable Provisioning 5.0 but disabled if you are upgrading from an earlier version to 5.0. If you wish to enable encryption post upgrade, execute the command:

```
./passwordEncryption.sh -enable
```

Once you enable encryption, Prime Cable Provisioning will not be able to support the 4.0 and 4.0.x API clients.

To check if the SHA1 encryption is enabled or not, execute the command:

```
./passwordEncryption.sh -status
```

Using changeSSLProperties.sh

You can use the **changeSSLProperties.sh** tool, which is found in the *BPR_HOME/ssl/bin* directory, to change key SSL configuration properties.

Table 28-2 lists the various options that you can use to change the SSL configuration.

Table 28-2 changeSSLProperties.sh Options

Option	Description	Option Parameters
./changeSSLProperties.sh **-ssl**	Use -**ssl** to enable or disable SSL or secure connection on RDU, API client, Admin UI or PWS. In case of Admin UI and PWS, this enables or disables the HTTPS mode of communication.	[rdu|api|adminui|pws] [enable/disable] For example: `./changeSSLProperties.sh -ssl rdu enable`
./changeSSLProperties.sh **-nssl**	Use -**nssl** to enable or disable non-secure connection with RDU, API client, Admin UI or PWS. In case of Admin UI and PWS, this enables or disables the HTTP mode of communication.	[rdu|api| adminui|pws] [enable/disable] For example: `./changeSSLProperties.sh -nssl rdu disable`
./changeSSLProperties.sh **-secret**	Use -**secret** to change the secret key for RDU, DPE and PWS,	[secret] For example: `./changeSSLProperties.sh -secret changeme`

Table 28-2 changeSSLProperties.sh Options (continued)

Option	Description	Option Parameters
./changeSSLProperties.sh **-csp**	Use **-csp** to change the default non-secure port number that RDU, Admin UI, API client or PWS listen on. By default, RDU listens on 49188. In case of an API client, the command lists all the secure RDU hosts and you can change the port number of any of those RDU hosts using the tool.	[rdu|api|adminui|pws] For example: `./changeSSLProperties.sh -csp rdu`
./changeSSLProperties.sh -**cnsp**	Use -**cnsp** to change the default non-secure port number that RDU, Admin UI, API client or PWS listen on. By default, RDU listens on 49187. In case of an API client, the command lists all the secure RDU hosts and you can change the port number of any of those RDU hosts using the tool.	[rdu|api|adminui|pws] For example: `./changeSSLProperties.sh -cnsp rdu`
./changeSSLProperties.sh -**list**	Use -**list** to list the secure or non-secure hosts. Use argument **s** to list the secure hosts and **ns** for non-secure hosts.	[s|ns] For example: `./changeSSLProperties.sh -list n`
./changeSSLProperties.sh -**ckl**	Use -**ckl** to changes the default keystore location. Respective property files get updated with this new keystore location. By default, the keystore is stored in BPR_HOME/lib/security folder.	[new location] For example: `./changeSSLProperties.sh -ckl /opt/CSCObac/lib/security/.keystore`
./changeSSLProperties.sh -**ckp**	Use -**ckp** to change the keystore password. You will be prompted to enter the old and new passwords. For security reasons all passwords will be prompted.	[new location] For example: `./changeSSLProperties.sh -ckp`
./changeSSLProperties.sh -**utp**	Use -**ctp** to update the truststore password in case you have changed the default truststore (cacerts) password. This option updates only the related property files and does not change cacerts password. Since cacerts can contain other trusted entries/certificate chains, there is no option to change the trust store passwords. However you can change the truststore (cacerts) password using java keytool command, if you wish so.	For example: `./changeSSLProperties.sh -utp`

Table 28-2 changeSSLProperties.sh Options (continued)

Option	Description	Option Parameters
./changeSSLProperties.sh **-cpkp**	Use **-cpkp** to change the password used to store the RDU, Admin UI and PWS keys. You will be prompted for old and new passwords. For security reasons all passwords will be prompted.	[rduladminuilpws] For example: `./changeSSLProperties.sh -cpkp`
./changeSSLProperties.sh **-gk**	Use **-gk** to generates a key pair, a public key and an associated private key. The new created RDU key pair is stored in the .keystore file under BPR_HOME/lib/security. The following values would be set by default (keylength 1024, validity 2 years, keyalg RSA, alias rducert, storetype JCEKS). You will be prompted for both keystore and key passwords.	For example: `./changeSSLProperties.sh -gk`
./changeSSLProperties.sh **-exp**	Use **-exp** to self-sign and export the certificate. This option locates you keystore file, self-signs the RDU certificate and exports rootCA.crt and rootCA.pem files to the BPR_HOME/lib/security folder.	For example: `./changeSSLProperties.sh -exp`
./changeSSLProperties.sh **-imp**	Use **-imp** to import a certificate to the cacerts trust store so that a chain of trust can be established between the certificate and RDU. If a chain of trust cannot be established, an error message appears. In case of CNR-EP you should copy the rootCA.pem file to the machine where CNR-EP is installed. The files must be copied under the BPR_HOME/bin/security folder.	[location form where to import] [alias] For example: `./changeSSLProperties.sh -imp`
./changeSSLProperties.sh **-help**	Use **-help** to view the help tips.	For example: `./changeSSLProperties.sh -help`

Using ws-cli.sh

You can use the **ws-cli.sh** tool, which is found in the *BPR_HOME/pws/bin* directory, to carry out some of the PWS configuration functions.

Table 28-3 lists the various options that are part of the ws-cli tool.

Table 28-3 WS CLI Tools

Option	Description	Option Parameters
./ws-cli.sh **-ardu,--addrdu <host> <port> <username> <password>**	Use this option to add a new RDU account. You could either use **-ardu** or **--addrdu**. Repeat the same command to add multiple RDUs.	For example: `#./ws-cli.sh -ardu test1-host  49187 admin  changeme`
./ws-cli.sh **-rrdu,--removerdu <host>**	Use this option to delete an existing RDU account. You could either use **-rrdu,** or **--removerdu**.	For example: `./ws-cli.sh -rrdu bac-test-lnx`
./ws-cli.sh -**srduc <host> <port> <username> <password>**	Use this option to update RDU username and password by providing host and port number. You could either use **-srduc,** or **--setrducredentials.**	For example: `#./ws-cli.sh -srductest1 -host 49187 admin changeme`
#./ws-cli.sh -lrdu <host>	Use this option to list the RDU commands. You could either use **-lrdu,** or **--listrdu.**	For example: `#./ws-cli.sh -lrdu test1-host`
./ws-cli.sh **-ll,--listlog <logger>**	Use this option to list all the loggers and their current severity levels. You could either use **-ll,** or **--listlog.**	For example: `./ws-cli.sh -ll`
./ws-cli.sh **-lp,--listproperty <property>**	Use this option to list all properties and their values. You could either use **-lp,** or **--listproperty.**	For example: `./ws-cli.sh -lp`
./ws-cli.sh **-rm,--removeproperty <property>**	Use this option to remove the specified property **-rm**,--**removeroperty**. Triggers running app to reload the properties.	For example: `./ws-cli.sh -removeproperty /cache/timeout`
./ws-cli.sh **-rp,--reloadproperty <property>**	Use this option to remove the specified property **-rp,--reloadproperty**. Triggers running app to reload the properties.	For example: `./ws-cli.sh -removeproperty /cache/timeout`
./ws-cli.sh **-rpd,--reloadproperty def**	Clears and reloads all the RDU property def caches.	For example: `./ws-cli.sh -rpd`
./ws-cli.sh **-sap,--saveproperty**	Saves the modifications.	For example: `./ws-cli.sh -sap`

Table 28-3 WS CLI Tools (continued)

Option	Description	Option Parameters
./ws-cli.sh **-sl,--setlog <logger=value>**	Updates the logger level to either error, warn, info, or debug.	For example: `./ws-cli.sh -sl general=DEBUG`
./ws-cli.sh **-sp,--setproperty <property=value>**	Adds a new property or updates an existing property.	For example: `./ws-cli.sh -sp /cache/timeout=100`

Scripts to Manage and Troubleshoot RDU Redundancy

Following are scripts that you can run to configure properties of HA resources as well as troubleshoot RDU redundancy. These scripts are available only when RDU is installed in redundancy mode. All these scripts are located under BPR_HOME/agent/HA/bin.

Table 28-4 lists the scripts that you can use to configure, monitor, and troubleshoot RDU redundancy.

Table 28-4 RDU Redundancy Scripts

Script name	Description	Syntax
monitor_ha_cluster.sh	Displays the summary of the CRM resource status. timing-details is an optional parameter, if it is provided, it displays complete summary about the CRM resources with timing details.	BPR_HOME/agent/HA/bin/monitor_ha_cluster.sh [timing-details]
manage_ha_resources.sh	This script is used to stop, start, and restart failed CRM resources.	BPR_HOME/agent/HA/bin/manage_ha_resources.sh <stop|start|resta rt> <res_IPaddr2_1|res_Filesystem_1|res_Filesystem_2 |res_Filesystem_3|res_bprAgent_1|all>
monitor_fs_sync_status.sh	This script prints the current status of file system syncup between primary and secondary node.	BPR_HOME/agent/HA/bin/monitor_fs_sync_status.sh
configure_vip.sh	This script configures the VIP address of redundancy setup. It takes two inputs, first the new VIP address and the second is CIDR value which should be in the range >0 and <=30.	BPR_HOME/agent/HA/bin/configure_vip.sh <ipaddress value> <CIDR value>

Table 28-4 RDU Redundancy Scripts

configure_failover_threshold.sh	This script configures failover timeout value and the migration threshold value of redundancy setup. You must specify the time in minutes.	BPR_HOME/agent/HA/bin/configure_failover_threshold.sh <timeout|threshold> <value in mins>
cleanup_ha_resources.sh	This script cleans up the CRM resources. Specify the resource name that you want to clean up as the parameter. To clean up all resources in one go, use the option *all*.	BPR_HOME/agent/HA/bin/cleanup_ha_resources.sh <res_IPaddr2_1|res_Filesystem_1|res_Filesystem_2|res_Filesystem_3|res_bprAgent_1|all>
standby_ha_switch.sh	This script troubleshoots or tests the setup manually. You can make primary or secondary node as standby using this script. Use the option *on* to make a node standby and *off* to get it online.	BPR_HOME/agent/HA/bin/standby_ha_switch.sh <primary|secondary> <on|off>
resolve_sb_victim_fs.sh	Run this script if you want to discard the updates being made to a server during a split-brain scenario. You must run the script from the server where you want to discard the updates. You need to mention which data should be discarded and give the option as either bprHome, bprData, or bprLog. To discard all the updates, use the option *all*.	BPR_HOME/agent/HA/bin/resolve_sb_victim.sh <bprHome|bprData|bprLog|all>
resolve_sb_survivor_fs.sh	This script should be run from the other server from where user wants to get update from, when there is a split-brain scenario. User has to mention option as either bprHome/bprData/bprLog.	BPR_HOME/agent/HA/bin/resolve_sb_survivor.sh <bprHome|bprData|bprLog|all>

PART 6

Appendices

This part contains the following sections:

- Technology Option Support
- Mapping PacketCable DHCP Options to Prime Cable Provisioning Properties

APPENDIX A

Technology Option Support

This chapter identifies the technology-specific options that Prime Cable Provisioning supports for each technology version and specifies the following attributes for each option:

- Option No.—Identifies the option number, as an integer or in dotted notation.
- Description—Describes the option.
- Encoding—Specifies the data format and the encoding of the option value. For detailed information on the encoding types, see Encoding Types for Defined Options, page 19-27.
- Validation—Specifies a validation rule that restricts the allowable option values.
- Multivalued—Indicates whether multiple options can be specified in a single configuration file. For suboptions, this value specifies whether the option can repeated within the parent option.
- Version—Identifies the technology versions that support the option number and encoding.

This appendix describes the options for these technologies:

- DOCSIS Option Support, page A-1
- DPoE Option Support, page A-22
- PacketCable Option Support, page A-23
- CableHome Option Support, page A-23

DOCSIS Option Support

Table A-1 describes DOCSIS options and identifies the specific version support for each option.

Table A-1 DOCSIS Options and Version Support

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
0	PAD	No length and no value	None	True	✓	✓	✓	✓
1	Downstream Frequency	Unsigned integer 32	Multiples of 62500	False	✓	✓	✓	✓
2	Upstream Channel ID	Unsigned integer 8	None	False	✓	✓	✓	✓
3	Network Access Control	Boolean	None	False	✓	✓	✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
4	Class of Service	Compound	None	True	✓	✓	✓	✓
4.1	Class ID	Unsigned integer 8	From 1 to 16	False	✓	✓	✓	✓
4.2	Maximum Downstream Rate	Unsigned integer 32	None	False	✓	✓	✓	✓
4.3	Maximum Upstream Rate	Unsigned integer 32	None	False	✓	✓	✓	✓
4.4	Upstream Channel Priority	Unsigned integer 8	Less than 8	False	✓	✓	✓	✓
4.5	Guaranteed Minimum Upstream Channel Data Rate	Unsigned integer 32	None	False	✓	✓	✓	✓
4.6	Maximum Upstream Channel Transmit Burst	Unsigned integer 16	None	False	✓	✓	✓	✓
4.7	Class-of-Service Privacy Enable	Boolean	None	False	✓	✓	✓	✓
6	CM MIC Configuration Setting	Byte 16	None	False	✓	✓	✓	✓
7	CMTS MIC Configuration Setting	Bytes	None	False	✓	✓	✓	✓
9	Software Upgrade Filename	NVTASCII	None	False	✓	✓	✓	✓
10	SNMP Write-Access Control	OIDCF	None	True	✓	✓	✓	✓
11	SNMP MIB Object	SNMPVarBind	None	True	✓	✓	✓	✓
14	CPE Ethernet MAC Address	MAC Address	None	True	✓	✓	✓	✓
15	Telephone Settings Option	Compound	None	False	✓	✓	✓	✓
15.2	Service Provider Name	NVTASCII	None	False	✓	✓	✓	✓
15.3	Telephone Number (1)	NVTASCII	None	False	✓	✓	✓	✓
15.4	Telephone Number (2)	NVTASCII	None	False	✓	✓	✓	✓
15.5	Telephone Number (3)	NVTASCII	None	False	✓	✓	✓	✓
15.6	Connection Threshold	Unsigned integer 8	None	False	✓	✓	✓	✓
15.7	Login Username	NVTASCII	None	False	✓	✓	✓	✓
15.8	Login Password	NVTASCII	None	False	✓	✓	✓	✓
15.9	DHCP Authenticate	Boolean	None	False	✓	✓	✓	✓
15.10	DHCP Server	IP Address	None	False	✓	✓	✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
15.11	RADIUS Realm	NVTASCII	None	False	✓	✓	✓	✓
15.12	PPP Authenticate	Unsigned integer 8	None	False	✓	✓	✓	✓
15.13	Demand Dial Inactivity Timer Threshold	Unsigned integer 32	None	False	✓	✓	✓	✓
16	SNMP IPv4 Address (No Longer Used)	IP Address	None	False	✓	✓	✓	✓
17	Baseline Privacy Configuration Setting	Compound	None	False	✓	✓	✓	✓
17.1	Authorize Wait Timeout	Unsigned integer 32	From 1 to 30	False	✓	✓	✓	✓
17.2	Reauthorize Wait Timeout	Unsigned integer 32	From 1 to 30	False	✓	✓	✓	✓
17.3	Authorization Grace Time	Unsigned integer 32	From 1 to 1800	False	✓			
17.3	Authorization Grace Time	Unsigned integer 32	From 1 to 6047999	False		✓	✓	✓
17.4	Operational Wait Timeout	Unsigned integer 32	From 1 to 10	False	✓	✓	✓	✓
17.5	Rekey Wait Timeout	Unsigned integer 32	From 1 to 10	False	✓	✓	✓	✓
17.6	TEK Grace Time	Unsigned integer 32	From 1 to 1800	False	✓			
17.6	TEK Grace Time	Unsigned integer 32	From 1 to 302399	False		✓	✓	✓
17.7	Authorize Reject Wait Timeout	Unsigned integer 32	From 1 to 600	False	✓	✓	✓	✓
17.8	SA Map Wait Timeout	Unsigned integer 32	From 1 to 10	False		✓	✓	✓
17.9	SA Map Max Retries	Unsigned integer 32	From 1 to 10	False		✓	✓	✓
18	Maximum Number of CPE	Unsigned integer 8	None	False	✓	✓	✓	✓
19	TFTP Server Timestamp	Unsigned integer 32	None	False	✓	✓	✓	✓
20	TFTP Server Provisioned Modem Address	IP Address	None	False	✓	✓	✓	✓
21	Software Upgrade TFTP Server	IP Address	None	False	✓	✓	✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
22	Upstream Packet Classification Encoding	Compound	None	True		✓	✓	✓
22.1	Classifier Reference	Unsigned integer 8	From 1 to 255	False		✓	✓	✓
22.2	Classifier Identifier	Unsigned integer 16	From 1 to 65535	False		✓	✓	✓
22.3	Service Flow Reference	Unsigned integer 16	From 1 to 65535	False		✓	✓	✓
22.4	Service Flow Identifier	Unsigned integer 32	Greater than 0	False		✓	✓	✓
22.5	Rule Priority	Unsigned integer 8	None	False		✓	✓	✓
22.6	Classifier Activation State	ActInact	None	False		✓	✓	✓
22.7	Dynamic Service Change Action	Unsigned integer 8	Less than 3	False		✓	✓	✓
22.9	IPv4 Packet Classification Encodings	Compound	None	False		✓	✓	✓
22.9.1	IPv4 Type of Service Range and Mask	Unsigned integer 8 triplet	None	False		✓	✓	✓
22.9.2	IP Protocol	Unsigned integer 16	Less than 258	False		✓	✓	✓
22.9.3	IPv4 Source Address	IP Address	None	False		✓	✓	✓
22.9.4	IPv4 Source Mask	IP Address	None	False		✓	✓	✓
22.9.5	IPv4 Destination Address	IP Address	None	False		✓	✓	✓
22.9.6	IPv4 Destination Mask	IP Address	None	False		✓	✓	✓
22.9.7	TCP/UDP Source Port Start	Unsigned integer 16	None	False		✓	✓	✓
22.9.8	TCP/UDP Source Port End	Unsigned integer 16	None	False		✓	✓	✓
22.9.9	TCP/UDP Destination Port Start	Unsigned integer 16	None	False		✓	✓	✓
22.9.10	TCP/UDP Destination Port End	Unsigned integer 16	None	False		✓	✓	✓
22.10	Ethernet LLC Packet Classification Encodings	Compound	None	False		✓	✓	✓
22.10.1	Destination MAC Address	MAC Address and Mask	None	False		✓	✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

				Multi-	DOCSIS Version			
Option No.	**Description**	**Encoding**	**Validation**	**valued**	**1.0**	**1.1**	**2.0**	**3.0**
22.10.2	Source MAC Address	MAC Address	None	False		✓	✓	✓
22.10.3	Ethertype/DSAP/ MacType	Unsigned integer 8 and unsigned integer 16	None	False		✓	✓	✓
22.11	IEEE 802.1P/Q Packet Classification Encodings	Compound	None	False		✓	✓	✓
22.11.1	IEEE 802.1P User_Priority	Unsigned integer 8 pair	Less than 8	False		✓	✓	✓
22.11.2	IEEE 802.1Q VLAN_ID	Unsigned integer 16	None	False		✓	✓	✓
22.12	IPv6 Packet Classification Encodings	Compound	None	False				✓
22.12.1	IPv6 Traffic Class Range and Mask	Unsigned integer 8 triplet	None	False				✓
22.12.2	IPv6 Flow Label	Unsigned integer 32	Greater than 0	False				✓
22.12.3	IPv6 Next Header Type	Unsigned integer 16	Less than 258	False				✓
22.12.4	IPv6 Source Address	IPv6 address	None	False				✓
22.12.5	IPv6 Source Prefix Length	Unsigned integer 8	Less than 129	False				✓
22.12.6	IPv6 Destination Address	IPv6 address	None	False				✓
22.12.7	IPv6 Destination Prefix Length	Unsigned integer 8	Less than 129	False				✓
22.13	CM Interface Mask (CMIM)	Bytes	None	False				✓
22.43	Vendor Specific Classifier Parameters	Compound	None	False		✓	✓	✓
22.43.8	Vendor ID	OUI	None	False		✓	✓	✓
22.43.5	L2VPN Encoding	Compound	None	False			✓	✓
22.43.5.1	VPNID Subtype	Bytes	None	False			✓	✓
22.43.5.2	NSI Encapsulation Subtype	Compound	None	False			✓	✓
22.43.5.2.1	Other Format Subtype	No length and no value	None	False			✓	✓
22.43.5.2.2	IEEE 802.1Q Format Subtype	Unsigned integer 16	None	False			✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
22.43.5.2.3	IEEE 802.1ad Format Subtype	Unsigned integer 32	None	False			✓	✓
22.43.5.2.4	MPLS Peer Format Subtype	Inet Address Peer	None	False			✓	✓
22.43.5.2.5	L2TPv3 Peer Format Subtype	Inet Address Peer	None	False			✓	✓
22.43.5.3	Enable eSAFE DHCP Snooping	Bytes	None	False			✓	✓
22.43.5.4	CM Interface Mask	Bytes	None	False			✓	✓
22.43.5.5	Attachment Group ID	Bytes	From 0 to 16	False			✓	✓
22.43.5.6	Source Attachment Individual ID	Bytes	From 0 to 16	False			✓	✓
22.43.5.7	Target Attachment Individual ID	Bytes	From 0 to 16	False			✓	✓
22.43.5.8	Ingress User Priority	Unsigned integer 8	From 0 to 7	False			✓	✓
22.43.5.9	User Priority Range	Unsigned integer 16	None	False			✓	✓
22.43.5.43	Vendor-Specific	Compound	None	False			✓	✓
23	Downstream Packet Classification Encoding	Compound	None	True		✓	✓	✓
23.1	Classifier Reference	Unsigned integer 8	From 1 to 255	False		✓	✓	✓
23.2	Classifier Identifier	Unsigned integer 16	From 1 to 65535	False		✓	✓	✓
23.3	Service Flow Reference	Unsigned integer 16	From 1 to 65535	False		✓	✓	✓
23.4	Service Flow Identifier	Unsigned integer 32	Greater than 0	False		✓	✓	✓
23.5	Rule Priority	Unsigned integer 8	None	False		✓	✓	✓
23.6	Classifier Activation State	Boolean	None	False		✓	✓	✓
23.7	Dynamic Service Change Action	Unsigned integer 8	Less than 3	False		✓	✓	✓
23.8	Classifier Error Encodings	Compound	None	False		✓	✓	✓
23.9	IPv4 Packet Classification Encodings	Compound	None	False		✓	✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

				Multi-	DOCSIS Version			
Option No.	Description	Encoding	Validation	valued	1.0	1.1	2.0	3.0
23.9.1	IPv4 Type of Service Range and Mask	Unsigned integer 8 triplet	None	False		✓	✓	✓
23.9.2	IP Protocol	Unsigned integer 16	Less than 258	False		✓	✓	✓
23.9.3	IPv4 Source Address	IP Address	None	False		✓	✓	✓
23.9.4	IPv4 Source Mask	IP Address	None	False		✓	✓	✓
23.9.5	IPv4 Destination Address	IP Address	None	False		✓	✓	✓
23.9.6	IPv4 Destination Mask	IP Address	None	False		✓	✓	✓
23.9.7	TCP/UDP Source Port Start	Unsigned integer 16	None	False		✓	✓	✓
23.9.8	TCP/UDP Source Port End	Unsigned integer 16	None	False		✓	✓	✓
23.9.9	TCP/UDP Destination Port Start	Unsigned integer 16	None	False		✓	✓	✓
23.9.10	TCP/UDP Destination Port End	Unsigned integer 16	None	False		✓	✓	✓
23.10	Ethernet LLC Packet Classification Encodings	Compound	None	False			✓	✓
23.10.1	Destination MAC Address	MAC Address and Mask	None	False		✓	✓	✓
23.10.2	Source MAC Address	MAC Address	None	False		✓	✓	✓
23.10.3	Ethertype/DSAP/MacType	Unsigned integer 8 and unsigned integer 16	None	False		✓	✓	✓
23.11	IEEE 802.1P/Q Packet Classification Encodings	Compound	None	False		✓	✓	✓
23.11.1	IEEE 802.1P User_Priority	Unsigned integer 8 pair	Less than 8	False		✓	✓	✓
23.11.2	IEEE 802.1Q VLAN_ID	Unsigned integer 16	None	False		✓	✓	✓
23.12	IPv6 Packet Classification Encodings	Compound	None	False				✓
23.12.1	IPv6 Traffic Class Range and Mask	Unsigned integer 8 triplet	None	False				✓
23.12.2	IPv6 Flow Label	Unsigned integer 32	Greater than 0	False				✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
23.12.3	IPv6 Next Header Type	Unsigned integer 16	Less than 258	False				✓
23.12.4	IPv6 Source Address	IPv6 address	None	False				✓
23.12.5	IPv6 Source Prefix Length	Unsigned integer 8	Less than 129	False				✓
23.12.6	IPv6 Destination Address	IPv6 address	None	False				✓
23.12.7	IPv6 Destination Prefix Length	Unsigned integer 8	Less than 129	False				✓
23.43	Vendor Specific Classifier Parameters	Compound	None	False		✓	✓	✓
23.43.5	L2VPN Encoding	Compound	None	False			✓	✓
23.43.5.1	VPNID Subtype	Bytes	None	False			✓	✓
23.43.5.2	NSI Encapsulation Subtype	Compound	None	False			✓	✓
23.43.5.2.1	Other Format Subtype	No length and no value	None	False			✓	✓
23.43.5.2.2	IEEE 802.1Q Format Subtype	Unsigned integer 16	None	False			✓	✓
23.43.5.2.3	IEEE 802.1ad Format Subtype	Unsigned integer 32	None	False			✓	✓
23.43.5.2.4	MPLS Peer Format Subtype	Inet Address Peer	None	False			✓	✓
23.43.5.2.5	L2TPv3 Peer Format Subtype	Inet Address Peer	None	False			✓	✓
23.43.5.3	Enable eSAFE DHCP Snooping	Bytes	None	False			✓	✓
23.43.5.4	CM Interface Mask	Bytes	None	False			✓	✓
23.43.5.5	Attachment Group ID	Bytes	From 0 to 16	False			✓	✓
23.43.5.6	Source Attachment Individual ID	Bytes	From 0 to 16	False			✓	✓
23.43.5.7	Target Attachment Individual ID	Bytes	From 0 to 16	False			✓	✓
23.43.5.8	Ingress User Priority	Unsigned integer 8	From 0 to 7	False			✓	✓
23.43.5.9	User Priority Range	Unsigned integer 16	None	False			✓	✓
23.43.5.43	Vendor-Specific	Compound	None	False			✓	✓
23.43.8	Vendor ID	OUI	None	False		✓	✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
24	Upstream Service Flow Scheduling	Compound	None	True		✓	✓	✓
24.1	Service Flow Reference	Unsigned integer 16	Greater than 0	False		✓	✓	✓
24.3	Service Identifier	Unsigned integer 16	None	False		✓	✓	✓
24.4	Service Class Name	ZTASCII	None	False		✓	✓	✓
24.6	Quality of Service Parameter Set Type	Bit Flag 8	Less than 8	False		✓	✓	✓
24.7	Traffic Priority	Unsigned integer 8	Less than 8	False		✓	✓	✓
24.8	Upstream Maximum Sustained Traffic Rate	Unsigned integer 32	None	False		✓	✓	✓
24.9	Maximum Traffic Burst	Unsigned integer 32	None	False		✓	✓	✓
24.10	Minimum Reserved Traffic Rate	Unsigned integer 32	None	False		✓	✓	✓
24.11	Assumed Minimum Reserved Rate Packet Size	Unsigned integer 16	None	False		✓	✓	✓
24.12	Timeout for active QoS Parameters	Unsigned integer 16	None	False		✓	✓	✓
24.13	Timeout for Admitted QoS Parameters	Unsigned integer 16	None	False		✓	✓	✓
24.14	Maximum Concatenated Burst	Unsigned integer 16	None	False		✓	✓	✓
24.15	Service Flow Scheduling Type	Service Flow	From 1 to 6	False		✓	✓	✓
24.16	Request/Transmission Policy	Bit Flag 32	Less than 512	False		✓	✓	✓
24.17	Nominal Polling Interval	Unsigned integer 32	None	False		✓	✓	✓
24.18	Tolerated Poll Jitter	Unsigned integer 32	None	False		✓	✓	✓
24.19	Unsolicited Grant Size	Unsigned integer 16	None	False		✓	✓	✓
24.20	Nominal Grant Interval	Unsigned integer 32	None	False		✓	✓	✓
24.21	Tolerated Grant Jitter	Unsigned integer 32	None	False		✓	✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
24.22	Grants per Interval	Unsigned integer 8	Less than 128	False		✓	✓	✓
24.23	IPv4 Type of Service Overwrite	Unsigned integer 8 pair	None	False		✓	✓	✓
24.24	Unsolicited Grant Time Reference	Unsigned integer 32	None	False		✓	✓	✓
24.25	Multiplier to Contention Request Backoff Window	Unsigned integer 8	From 4 to 12	False				✓
24.26	Multiplier to Number of Bytes Requested	Unsigned integer 8	Values 1, 2, 4, 8, or 16	False				✓
24.27	Maximum Requests per SID Cluster	Unsigned integer 8	Less than 256	False				✓
24.28	Maximum Outstanding Bytes per SID Cluster	Unsigned integer 32	Less than 4294967296	False				✓
24.29	Maximum Total Bytes Requested per SID Cluster	Unsigned integer 32	Less than 4294967296	False				✓
24.30	Maximum Time in the SID Cluster	Unsigned integer 16	Less than 65535	False				✓
24.31	Service Flow Required Attribute Mask	Bit Flag 32	None	False				✓
24.32	Service Flow Forbidden Attribute Mask	Bit Flag 32	None	False				✓
24.33	Service Flow Attribute Aggregation Mask	Bit Flag 32	None	False				✓
24.34	Application Identifier	Bit Flag 32	None	False				✓
24.43	Vendor Specific QoS Parameters	Compound	None	False		✓	✓	✓
24.43.8	Vendor ID	OUI	None	False		✓	✓	✓
24.43.5	L2VPN Encoding	Compound	None	False			✓	✓
24.43.5.1	VPNID Subtype	Bytes	None	False			✓	✓
24.43.5.2	NSI Encapsulation Subtype	Compound	None	False			✓	✓
24.43.5.2.1	Other Format Subtype	No length and no value	None	False			✓	✓
24.43.5.2.2	IEEE 802.1Q Format Subtype	Unsigned integer 16	None	False			✓	✓
24.43.5.2.3	IEEE 802.1ad Format Subtype	Unsigned integer 32	None	False			✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
24.43.5.2.4	MPLS Peer Format Subtype	Inet Address Peer	None	False			✓	✓
24.43.5.2.5	L2TPv3 Peer Format Subtype	Inet Address Peer	None	False			✓	✓
24.43.5.3	Enable eSAFE DHCP Snooping	Bytes	None	False			✓	✓
24.43.5.4	CM Interface Mask	Bytes	None	False			✓	✓
24.43.5.5	Attachment Group ID	Bytes	From 0 to 16	False			✓	✓
24.43.5.6	Source Attachment Individual ID	Bytes	From 0 to 16	False			✓	✓
24.43.5.7	Target Attachment Individual ID	Bytes	From 0 to 16	False			✓	✓
24.43.5.8	Ingress User Priority	Unsigned integer 8	From 0 to 7	False			✓	✓
24.43.5.9	User Priority Range	Unsigned integer 16	None	False			✓	✓
24.43.5.43	Vendor-Specific	Compound	None	False			✓	✓
25	Downstream Service Flow Scheduling	Compound	None	True		✓	✓	✓
25.1	Service Flow Reference	Unsigned integer 16	Greater than 0	False		✓	✓	✓
25.3	Service Identifier	Unsigned integer 16	None	False		✓	✓	✓
25.4	Service Class Name	ZTASCII	None	False		✓	✓	✓
25.6	Quality of Service Parameter Set Type	Bit Flag 8	Less than 8	False		✓	✓	✓
25.7	Traffic Priority	Unsigned integer 8	Less than 8	False		✓	✓	✓
25.8	Downstream Maximum Sustained Traffic Rate	Unsigned integer 32	None	False		✓	✓	✓
25.9	Maximum Traffic Burst	Unsigned integer 32	None	False		✓	✓	✓
25.10	Minimum Reserved Traffic Rate	Unsigned integer 32	None	False		✓	✓	✓
25.11	Assumed Minimum Reserved Rate Packet Size	Unsigned integer 16	None	False		✓	✓	✓
25.12	Timeout for active QoS Parameters	Unsigned integer 16	None	False		✓	✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

				Multi-	DOCSIS Version			
Option No.	Description	Encoding	Validation	valued	1.0	1.1	2.0	3.0
25.13	Timeout for Admitted QoS Parameters	Unsigned integer 16	None	False		✓	✓	✓
25.14	Maximum Downstream Latency	Unsigned integer 32	None	False		✓	✓	✓
25.23	IPv4 Type of Service (DSCP) Overwrite	Unsigned integer 8 pair	None	False				✓
25.31	Service Flow Required Attribute Mask	Bit Flag 32	None	False				✓
25.32	Service Flow Forbidden Attribute Mask	Bit Flag 32	None	False				✓
25.33	Service Flow Attribute Aggregation Mask	Bit Flag 32	None	False				✓
25.34	Application Identifier	Bit Flag 32	None	False				✓
25.43	Vendor Specific QoS Parameters	Compound	None	False		✓	✓	✓
25.43.8	Vendor ID	OUI	None	False		✓	✓	✓
26	Payload Header Suppression	Compound	None	True		✓	✓	✓
26.1	Classifier Reference	Unsigned integer 8	Greater than 0	False		✓	✓	✓
26.2	Classifier Identifier	Unsigned integer 16	Greater than 0	False		✓	✓	✓
26.3	Service Flow Reference	Unsigned integer 16	Greater than 0	False		✓	✓	✓
26.4	Service Flow Identifier	Unsigned integer 32	Greater than 0	False		✓	✓	✓
26.5	Dynamic Service Change Action	SrvChangeAct	Less than 4	False		✓	✓	✓
26.7	Payload Header Suppression Field (PHSF)	Bytes	None	False		✓	✓	✓
26.8	Payload Header Suppression Index (PHSI)	Unsigned integer 8	Greater than 0	False		✓	✓	✓
26.9	Payload Header Suppression Mask (PHSM)	Bytes	None	False		✓	✓	✓
26.10	Payload Header Suppression Size (PHSS)	Unsigned integer 8	None	False		✓	✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
26.11	Payload Header Suppression Verification (PHSV)	Verify	None	False		✓	✓	✓
26.13	Dynamic Bonding Change Action	Unsigned integer 8	Less than 2	False				✓
26.43	Vendor Specific PHS Parameters	Compound	None	False		✓	✓	✓
26.43.8	Vendor ID	OUI	None	False		✓	✓	✓
28	Maximum Number of Classifiers	Unsigned integer 16	None	False		✓	✓	✓
29	Privacy Enable	Boolean	None	False		✓	✓	✓
32	Manufacturer CVC	Bytes	None	False		✓	✓	✓
33	Co-signer CVC	Bytes	None	False		✓	✓	✓
34	SnmpV3 Kickstart Value	Compound	None	False		✓	✓	✓
34.1	SnmpV3 Kickstart Security Name	NVTASCII	None	False		✓	✓	✓
34.2	SnmpV3 Kickstart Manager Public Number	Bytes	None	False		✓	✓	✓
35	Subscriber Management Control	Bytes	None	False		✓	✓	✓
36	Subscriber Management CPE IPv4 Table	IP Address N	None	False		✓	✓	✓
37	Subscriber Management Filter Groups	Bytes	None	False		✓	✓	✓
38	SNMPv3 Notification Receiver	Compound	None	True		✓	✓	✓
38.1	SNMPv3 Notification Receiver IPv4 Address	IP address	None	False		✓	✓	✓
38.2	SNMPv3 Notification Receiver UDP Port	Unsigned integer 16	None	False		✓	✓	✓
38.3	SNMPv3 Notification Receiver Trap Type	SNMP Trap Type	From 1 to 5	False		✓	✓	✓
38.4	SNMPv3 Notification Receiver Timeout	Unsigned integer 16	None	False		✓	✓	✓
38.5	SNMPv3 Notification Receiver Retries	Unsigned integer 16	From 0 to 255	False		✓	✓	✓
38.6	Notification Receiver Filtering Parameters	OID	None	False		✓	✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
38.7	Notification Receiver Security Name	NVTASCII	None	False		✓	✓	✓
38.8	SNMPv3 Notification Receiver IPv6 Address	IPv6 address	None	False				✓
39	Enable 2.0 Mode	Boolean	None	False			✓	✓
40	Enable Test Modes	Boolean	None	True			✓	✓
41	Downstream Channel List	Compound	None	True			✓	✓
41.1	Single Downstream Channel	Compound	None	True			✓	✓
41.1.1	Single Downstream Channel Timeout	Unsigned integer 16	None	False			✓	✓
41.1.2	Single Downstream Channel Frequency	Unsigned integer 32	Multiples of 62500	False			✓	✓
41.2	Downstream Frequency Range	Compound	None	True			✓	✓
41.2.1	Downstream Frequency Range Timeout	Unsigned integer 16	None	False			✓	✓
41.2.2	Downstream Frequency Range Start	Unsigned integer 32	Multiples of 62500	False			✓	✓
41.2.3	Downstream Frequency Range End	Unsigned integer 32	Multiples of 62500	False			✓	✓
41.2.4	Downstream Frequency Range Step Size	Unsigned integer 32	None	False			✓	✓
41.3	Default Scanning	Unsigned integer 16	None	True			✓	✓
42	Multicast MAC Address	MAC Address	None	True			✓	✓
43	DOCSIS Extension Field (OUI FF-FF-FF)	Compound	None	True	✓	✓	✓	✓
43.1	CM Load Balancing Policy ID	Unsigned integer 32	None	False			✓	✓
43.2	CM Load Balancing Priority	Unsigned integer 32	None	False			✓	✓
43.3	CM Load Balancing Group ID	Unsigned integer 32	None	False			✓	✓
43.4	CM Ranging Class ID Extension	Unsigned integer 16	None	False			✓	✓
43.5	L2VPN Encoding	Compound	None	False			✓	✓
43.5.1	VPNID Subtype	Bytes	None	False			✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
43.5.2	NSI Encapsulation Subtype	Compound	None	False			✓	✓
43.5.2.1	Other Format Subtype	No length and no value	None	False			✓	✓
43.5.2.2	IEEE 802.1Q Format Subtype	Unsigned integer 16	None	False			✓	✓
43.5.2.3	IEEE 802.1ad Format Subtype	Unsigned integer 32	None	False			✓	✓
43.5.2.4	MPLS Peer Format Subtype	Inet Address Peer	None	False			✓	✓
43.5.2.5	L2TPv3 Peer Format Subtype	InetAddress Peer	None	False			✓	✓
43.5.3	Enable eSAFE DHCP Snooping	Bytes	None	False			✓	✓
43.5.4	CM Interface Mask	Bytes	None	False			✓	✓
43.5.5	Attachment Group ID	Bytes	From 0 to 16	False			✓	✓
43.5.6	Source Attachment Individual ID	Bytes	From 0 to 16	False			✓	✓
43.5.7	Target Attachment Individual ID	Bytes	From 0 to 16	False			✓	✓
43.5.8	Ingress User Priority	Unsigned integer 8	From 0 to 7	False			✓	✓
43.5.9	User Priority Range	Unsigned integer 16	None	False			✓	✓
43.5.43	Vendor-Specific	Compound	None	False			✓	✓
43.6	Extended CMTS MIC Configuration Setting	Compound	None	False				✓
43.6.1	Extended CMTS MIC HMAC type	Unsigned integer 8	Values 1, 2, 43	False				✓
43.6.2	Extended CMTS MIC Bitmap	Bytes	None	False				✓
43.6.3	Explicit Extended CMTS MIC Digest Subtype	Bytes	None	False				✓
43.7	Source Address Verification (SAV) Authorization Encoding	Compound	None	False			✓	✓
43.7.1	Name of an SAV Group configured in the CMTS	ZTASCII	From 1 to 15	False			✓	✓
43.7.2	SAV Static Prefix Subtype Encodings	Compound	None	False			✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
43.7.2.1	SAV Static Prefix Address Subtype	IPv4 or IPv6 Address	None	False			✓	✓
43.7.2.2	SAV Static Prefix Length Subtype	Bit Flag 8	Less than 129	False			✓	✓
43.8	Vendor ID	OUI	None	False	✓	✓	✓	✓
43.9	Cable Modem Mask Subtype Encodings	Compound	None	False				✓
43.9.1	Cable Modem Required Attribute Mask	Bit flag 32	None	False				✓
43.9.2	Cable Modem Forbidden Attribute Mask	Bit flag 32	None	False				✓
43.10	IP Multicast Join Authorization Encoding	Suboptions	None	False				✓
43.10.1	Name of an IP Multicast Profile configured in the CMTS	NVTASCII	From 1 to 15	True				✓
43.10.2	IP Multicast Join Authorization Static Session Rule Subtype Encodings	Compound	None	True				✓
43.10.2.1	Rule Priority	Unsigned integer 8	None	False				✓
43.10.2.2	Authorization Action	Authorization action	None	False				✓
43.10.2.3	Source Prefix Address Subtype	IPv4 or IPv6 address	None	False				✓
43.10.2.4	Source Prefix Length Subtype	Bit flag 8	Less than 129	False				✓
43.10.2.5	Group Prefix Address Subtype	IPv4 or IPv6 address	None	False				✓
43.10.2.6	Group Prefix Length Subtype	Bit flag 8	Less than 129	False				✓
43.10.3	Maximum Multicast Sessions Encoding	Unsigned integer 16	None	False				✓
43	DOCSIS Extension Field (OUI 00-00-0C)	Compound	None	True	✓	✓	✓	✓
43.1	Static Downstream Frequency	Unsigned integer 32	None	False	✓	✓	✓	✓
43.2	Sync Loss Timeout	Unsigned integer 32	None	False	✓	✓	✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
43.3	Update Boot Monitor Image	NVTASCII	None	False	✓	✓	✓	✓
43.4	Power Backoff	Unsigned integer 16	None	False	✓	✓	✓	✓
43.8	Vendor ID	OUI	None	False	✓	✓	✓	✓
43.9	Update Factory System Image	Boolean	None	False	✓	✓	✓	✓
43.10	Phone Lines	Unsigned integer 8	None	False	✓	✓	✓	✓
43.11	IP Precedence Settings	Compound	None	True	✓	✓	✓	✓
43.11.1	IP Precedence Value	Unsigned integer 8	None	False	✓	✓	✓	✓
43.11.2	Rate Limit	Unsigned integer 32	None	False	✓	✓	✓	✓
43.128	IOS Configuration Filename	NVTASCII	None	False	✓	✓	✓	✓
43.129	IOS Config File Without Console Disable	NVTASCII	None	False	✓	✓	✓	✓
43.131	IOS CLI Command	NVTASCII	None	True	✓	✓	✓	✓
43.132	1.0 Plus Flow Encodings	Compound	None	False	✓	✓	✓	✓
43.132.1	1.0 Plus Flow ID	Unsigned integer 8	None	False	✓	✓	✓	✓
43.132.2	Class ID	Unsigned integer 8	None	False	✓	✓	✓	✓
43.132.3	Unsolicited Grant Size	Unsigned integer 16	From 1 to 65535	False	✓	✓	✓	✓
43.132.4	Nominal Grant Interval	Unsigned integer 32	From 1 to 65535	False	✓	✓	✓	✓
43.132.5	Grants Per Interval	Unsigned integer 8	From 0 to 127	False	✓	✓	✓	✓
43.132.6	Embedded Voice Calls	Unsigned integer 8	From 0 to 127	False	✓	✓	✓	✓
43.132.7	Hold Queue Length	Unsigned integer 16	From 0 to 4096	False	✓	✓	✓	✓
43.132.8	Fair Queue	Compound	None	False	✓	✓	✓	✓
43.132.8.1	Congestive Discard Threshold	Unsigned integer 16	From 1 to 4096	False	✓	✓	✓	✓
43.132.8.2	Number of Dynamic Conversation Queues	Unsigned integer 16	From 16 to 4096	False	✓	✓	✓	✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
43.132.8.3	Number of Reservable Conversation Queues	Unsigned integer 16	From 0 to 1000	False	✓	✓	✓	✓
43.132.9	Custom Queue List Length	Unsigned integer 8	From 1 to 16	False	✓	✓	✓	✓
43.132.10	Random Detection	Boolean	None	False	✓	✓	✓	✓
43.132.11	Priority Group	Unsigned integer 8	From 1 to 16	False	✓	✓	✓	✓
43.132.12	Service Policy File	NVTASCII	None	False	✓	✓	✓	✓
43.132.13	Inactivity Timer	Unsigned integer 16	From 1 to 10080	False	✓	✓	✓	✓
43.132.14	COS Tag	NVTASCII	None	False	✓	✓	✓	✓
43.133	Downstream Sub Channel ID	Unsigned integer 8	From 0 to 15	False	✓	✓	✓	✓
43.134	SU Tag	NVTASCII	None	False	✓	✓	✓	✓
45	Downstream Unencrypted Traffic (DUT) Filtering Encoding	Compound	None	False			✓	✓
45.1	Downstream Unencrypted Traffic (DUT) Control	Boolean	None	False			✓	✓
45.2	Downstream Unencrypted Traffic (DUT) CMIM	Bytes	None	False			✓	✓
53	SNMPv1v2c Coexistence Configuration	Compound	None	True				✓
53.1	SNMPv1v2c Community Name	ZTASCII	From 1 to 32	False				✓
53.2	SNMPv1v2c Transport Address Access	Compound	None	True				✓
53.2.1	SNMPv1v2c Transport Address	Transport address and mask	None	False				✓
53.2.2	SNMPv1v2c Transport Address Mask	Transport address and mask	None	False				✓
53.3	SNMPv1v2c Access View Type	Access view type	None	False				✓
53.4	SNMPv1v2c Access View Name	ZTASCII	From 1 to 32	False				✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
54	SNMPv3 Access View	Compound	None	True				✓
54.1	SNMPv3 Access View Name	ZTASCII	From 1 to 32	False				✓
54.2	SNMPv3 Access View Subtree	OID	None	False				✓
54.3	SNMPv3 Access View Mask	Bytes	From 1 to 16	False				✓
54.4	SNMPv3 Access View Type	Access view control	None	False				✓
55	SNMP CPE Access Control	CPE access control	None	False				✓
56	Channel Assignment Configuration Settings	Compound	None	True				✓
56.1	Transmit Channel Assignment Configuration Setting	Unsigned integer 8	None	False				✓
56.2	Receive Channel Assignment Configuration Setting	Unsigned integer 32	None	False				✓
58	Software Upgrade IPv6 TFTP Server	IPv6 address	None	False				✓
59	TFTP Provisioned Modem IPv6 Address	IPv6 address	None	False				✓
60	Upstream Drop Packet Classification Encoding	Compound	None	True				✓
60.1	Classifier Reference	Unsigned integer 8	From 1 to 255	False				✓
60.2	Classifier Identifier	Unsigned integer 16	From 1 to 65535	False				✓
60.5	Rule Priority	Unsigned integer 8	None	False				✓
60.6	Classifier Activation State	ActInact	None	False				✓
60.7	Dynamic Service Change Action	Unsigned integer 8	Less than 3	False				✓
60.9	IPv4 Packet Classification Encodings	Compound	None	False				✓
60.9.1	IPv4 Type of Service Range and Mask	Unsigned integer 8 triplet	None	False				✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
60.9.2	IP Protocol	Unsigned integer 16	Less than 258	False				✓
60.9.3	IPv4 Source Address	IP address	None	False				✓
60.9.4	IPv4 Source Mask	IP address	None	False				✓
60.9.5	IPv4 Destination Address	IP address	None	False				✓
60.9.6	IPv4 Destination Mask	IP address	None	False				✓
60.9.7	TCP/UDP Source Port Start	Unsigned integer 16	None	False				✓
60.9.8	TCP/UDP Source Port End	Unsigned integer 16	None	False				✓
60.9.9	TCP/UDP Destination Port Start	Unsigned integer 16	None	False				✓
60.9.10	TCP/UDP Destination Port End	Unsigned integer 16	None	False				✓
60.10	Ethernet LLC Packet Classification Encodings	Compound	None	False				✓
60.10.1	Destination MAC Address	MAC address and mask	None	False				✓
60.10.2	Source MAC Address	MAC address	None	False				✓
60.10.3	Ethertype/DSAP/MacType	Unsigned integer 8 and 16	None	False				✓
60.11	IEEE 802.1P/Q Packet Classification Encodings	Compound	None	False				✓
60.11.1	IEEE 802.1P User_Priority	Unsigned integer 8 pair	Less than 8	False				✓
60.11.2	IEEE 802.1Q VLAN_ID	Unsigned integer 16	None	False				✓
60.12	IPv6 Packet Classification Encodings	Compound	None	False				✓
60.12.1	IPv6 Traffic Class Range and Mask	Unsigned integer 8 triplet	None	False				✓
60.12.2	IPv6 Flow Label	Unsigned integer 32	Greater than 0	False				✓
60.12.3	IPv6 Next Header Type	Unsigned integer 16	Less than 258	False				✓
60.12.4	IPv6 Source Address	IPv6 address	None	False				✓

Table A-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
60.12.5	IPv6 Source Prefix Length	Unsigned integer 8	Less than 129	False				✓
60.12.6	IPv6 Destination Address	IPv6 address	None	False				✓
60.12.7	IPv6 Destination Prefix Length	Unsigned integer 8	Less than 129	False				✓
60.13	CM Interface Mask (CMIM)	Bytes	None	False				✓
60.43	Vendor Specific Classifier Parameters	Compound	None	False				✓
60.43.8	Vendor ID	OUI	None	False				✓
61	Subscriber Management CPE IPv6 Table	IPv6 Address N	None	False				✓
62	Upstream Drop Classifier Group ID	Bytes	None	False				✓
63	Subscriber Management Control Max CPE IPv6 Prefix	Unsigned integer 16	None	False				✓
64	CMTS Static Multicast Session Encoding	Compound	None	True				✓
64.1	Static Multicast Group Encoding	IPv4 or IPv6 address	None	False				✓
64.2	Static Multicast Source Encoding	IPv4 or IPv6 address	None	False				✓
64.3	Static Multicast CMIM Encoding	Bytes	None	False				✓
255	End-of-Data Marker	No length and no value	None	False	✓	✓	✓	✓

DPoE Option Support

Table A-2 identifies the DPoE options that Prime Cable Provisioning supports.

Table A-2 DPoE Options

Option No.	Description	Encoding	Validation	Multi-valued
22.14	: IEEE 802.1ad S-VLAN and C-VLAN Frame Classification Encodings	SubOptions		false
22.14.1	IEEE 802.1ad S-VLAN TPID	Bytes	@length 2 2	false
22.14.2	IEEE 802.1ad S-VLAN VID	Bytes	@length 2 2	false
22.14.5	IEEE 802.1ad C-VLAN TPID	Bytes	@length 2 2	False
22.14.6	IEEE 802.1ad C-VLAN VID	Bytes	@length 2 2	False
22.15	IEEE 802.1ah I-TAG Packet Classification Encodings			False
22.15.1	IEEE 802.1ah I-TAG I-TPID	Bytes	@length 2 2	False
22.15.2	IEEE 802.1ah I-TAG I-SID	Bytes	@length 3 3	false

PacketCable Option Support

Table A-3 identifies the PacketCable MTA options that Prime Cable Provisioning supports.

Table A-3 PacketCable MTA Options

Option No.	Description	Encoding	Validation	Multi-valued	PacketCable Version		
					1.0	1.1	1.5
11	SNMP MIB Object	SNMPVarBind with 1-byte length	None	True	✓	✓	✓
38	SNMPv3 Notification Receiver	Suboptions	None	True	✓	✓	✓
38.1	SNMPv3 Notification Receiver IP Address	IP address	None	False	✓	✓	✓
38.2	SNMPv3 Notification Receiver UDP Port Number	Unsigned integer 16	None	False	✓	✓	✓
38.3	SNMPv3 Notification Receiver Trap Type	SNMP trap type	From 1 to 5	False	✓	✓	✓
38.4	SNMPv3 Notification Receiver Timeout	Unsigned integer 16	None	False	✓	✓	✓
38.5	SNMPv3 Notification Receiver Retries	Unsigned integer 16	From 0 to 255	False	✓	✓	✓
38.6	Notification Receiver Filtering Parameters	OID	None	False	✓	✓	✓
38.7	Notification Receiver Security Name	NVTASCII	None	False	✓	✓	✓
43	Vendor-Specific Information	Suboptions	None	True	✓	✓	✓
43.8	Vendor ID	OUI	None	False	✓	✓	✓
64	SNMP MIB Object	SNMPVarBind with 2-byte length	None	True	✓	✓	✓
254	Telephony Config File Start/End	Unsigned integer 8	Must be 1 or 255	False	✓	✓	✓

CableHome Option Support

Table A-4 identifies the non-secure CableHome options that Prime Cable Provisioning supports.

Table A-4 CableHome Options and Version Support

Option No.	Description	Encoding	Validation	Multi-valued	CableHome Version 1.0
0	PAD	No length and no value	None	True	✓
9	Software Upgrade Filename	NVTASCII	None	False	✓

Table A-4 CableHome Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	CableHome Version 1.0
10	SNMP Write-Access Control	OIDCF	None	True	✓
12	Modem IP Address	IP address	None	False	✓
14	CPE Ethernet MAC Address	MAC address	None	True	✓
21	Software Upgrade TFTP Server	IP address	None	False	✓
28	SNMP MIB Object	SNMPVarBind	None	True	✓
32	Manufacturer CVC	Bytes	None	False	✓
33	Co-signer CVC	Bytes	None	True	✓
34	SnmpV3 Kickstart Value	Suboptions	None	False	✓
34.1	SnmpV3 Kickstart Security Name	NVTASCII	None	False	✓
38	SNMPv3 Notification Receiver	Suboptions	None	True	✓
38.1	SNMPv3 Notification Receiver IP Address	IP address	None	False	✓
38.2	SNMPv3 Notification Receiver UDP Port Number	Unsigned integer 16	None	False	✓
38.3	SNMPv3 Notification Receiver Trap Type	SNMP trap type	From 1 to 5	False	✓
38.4	SNMPv3 Notification Receiver Timeout	Unsigned integer 16	None	False	✓
38.5	SNMPv3 Notification Receiver Retries	Unsigned integer 16	None	False	✓
38.6	Notification Receiver Filtering Parameters	OID	None	False	✓
38.7	Notification Receiver Security Name	NVTASCII	None	False	✓
43	Vendor-Specific Information	Suboptions	None	True	✓
43.1	Vendor ID	OUI	None	False	✓
53	PS MIC. A 20-octet SHA-1 hash of PS config file	Bytes	None	False	✓
255	End-of-Data Marker	No length and no value	None	False	✓

APPENDIX B

Mapping PacketCable DHCP Options to Prime Cable Provisioning Properties

This chapter identifies the mapping of Prime Cable Provisioning properties to the PacketCable DHCP options used for PacketCable provisioning and includes:

- Option 122 and Cisco Prime Cable Provisioning Property Comparison, page B-2
- Option 177 and Cisco Prime Cable Provisioning Property Comparison, page B-3

The minimum required set of these properties is configured, during installation, in the *BPR_HOME/cnr_ep/conf/cnr_ep.properties* file. This file resides on the Cisco Prime Network Registrar host. The set of properties defined in *cnr_ep.properties* is applied to all PacketCable voice technology devices in the provisioning group. Like other Prime Cable Provisioning properties, you can also set these properties on a device or a Class of Service. Setting them at the RDU, using either the administrator user interface or the application programming interface (API), overrides the corresponding values set in the *cnr_ep.properties* file.

Note See Using KeyGen Tool, page 28-9, for information on changing these key configuration properties.

Prime Cable Provisioning supports both PacketCable DHCP Option 122 (as specified in RFC 3495 and 3594) and the deprecated PacketCable DHCP Option 177. Prime Cable Provisioning does not ignore DHCP requests when it cannot populate option 122 and/or 177 content. Whatever Option 122/177 content is available is populated and the decision to ignore the option is left to the eMTA.

When Prime Cable Provisioning receives a DHCP request asking for both options 122 and 177, Prime Cable Provisioning ignores the request for Option 177 and populates only Option 122 content.

Caution There should be only one instance of each property in *BPR_HOME/cnr_ep/conf/cnr_ep.properties*.

Option 122 and Cisco Prime Cable Provisioning Property Comparison

Table B-1 identifies the Prime Cable Provisioning properties as they apply to the definition of Option 122 in RFC 3495 and RFC 3594.

Table B-1 DHCP Option 122 to Prime Cable Provisioning Property Comparison

DHCP Option	Type	Prime Cable Provisioning Property Name
6	IP addr	*/ccc/dns/primary*
6	IP addr	*/ccc/dns/secondary*
122.1	IP addr	*/ccc/dhcp/primary*
122.2	IP addr	*/ccc/dhcp/secondary*
122.3	FQDN	*/ccc/prov/fqdn* **Note** Option 122.3 is automatically filled by Prime Cable Provisioning; consequently, do not set this property manually.
122.4	Integer	*/ccc/kerb/auth/backoff/nomTimeout* */ccc/kerb/auth/backoff/maxTimeout* */ccc/kerb/auth/backoff/maxRetries*
122.5	Integer	*/ccc/kerb/app/backoff/nomTimeout* */ccc/kerb/app/backoff/maxTimeout* */ccc/kerb/app/backoff/maxRetries*
122.6	String	*/ccc/kerb/realm*
122.7	Boolean	*/ccc/tgt*
122.8	Integer	*/ccc/prov/timer*
122.9	Integer	*/ccc/security/ticket/invalidation*

⚠ **Caution** If any of */ccc/kerb/auth/backoff/nomTimeout*, */ccc/kerb/auth/backoff/maxTimeout*, or */ccc/kerb/auth/backoff/maxRetries* are defined, they must all be defined. Similarly, if any of */ccc/kerb/app/backoff/nomTimeout*, */ccc/kerb/app/backoff/maxTimeout*, or */ccc/kerb/app/backoff/maxRetries* are defined, they must all be defined.

Option 177 and Cisco Prime Cable Provisioning Property Comparison

In accordance with PacketCable compliance wave 26, Option 177 is deprecated, and Option 122 is now the preferred MTA provisioning option. For legacy devices that still support Option 177, Table B-2 identifies the Prime Cable Provisioning properties as they apply to the definition of Option 177.

Table B-2 DHCP Option 177 to Prime Cable Provisioning Property Comparison

Option 177	Type	Prime Cable Provisioning Property Names
177.1	ip addr	*/pktcbl/dhcp/primary*
177.2	ip addr	*/pktcbl/dhcp/secondary*
177.3	fqdn	*/pktcbl/snmp/entity/fqdn*
177.4	ip addr	*/pktcbl/dns/primary*
177.5	ip addr	*/pktcbl/dns/secondary*
177.6	string	*/pktcbl/snmp/realm*
177.7	boolean	*/pktcbl/snmp/tgt*
177.8	integer	*/pktcbl/provisioning/timer*
177.10	integer	*/pktcbl/kerberos/authentication/backoff/nomTimeout* */pktcbl/kerberos/authentication/backoff/maxTimeout* */pktcbl/kerberos/authentication/backoff/maxRetries*
177.11	integer	*/pktcbl/kerberos/application/backoff/nomTimeout* */pktcbl/kerberos/application/backoff/maxTimeout* */pktcbl/kerberos/application/backoff/maxRetries*
177.12	integer	*/pktcbl/snmp/kerberos/ticket/invalidation*

GLOSSARY

A

alert A syslog or SNMP message notifying an operator or administrator of a problem.

API Application programming interface. Specification of function-call conventions that defines an interface to a service.

audit log A log file containing a summary of major changes in the RDU database. This includes changes to system defaults, technology defaults, DHCP criteria, and Class of Service.

B

Prime Cable Provisioning An integrated solution for data-over-cable service providers to configure and manage broadband modems, and enable and administer subscriber self-registration and activation.
Prime Cable Provisioning is a scalable product capable of supporting millions of devices.

bandwidth The difference between the highest and lowest frequencies available for network signals. The term is also used to describe the rated throughput capacity of a given network medium or protocol.

broadband Transmission system that multiplexes multiple independent signals onto one cable. In Telecommunications terminology, any channel having a bandwidth greater than a voice-grade channel (4 kHz). In LAN terminology, a coaxial cable on which analog signaling is used.

C

cable modem termination system *See* CMTS.

CableHome A CableLabs initiative to develop a standardized infrastructure to let cable operators extend high-quality, value-added services to the home local area network.

caching A form of replication in which information learned during a previous transaction is used to process later transactions.

chaddr DHCP client hardware (MAC) address. Sent in an RFC 2131 packet between the client and server.

client class A Network Registrar feature that provides differentiated services to users that are connected to a common network. The client class is used in the Prime Cable Provisioning DHCP criteria to provide differentiated DHCP services to devices.

CMTS Cable modem termination system. A CMTS is a component that exchanges digital signals with cable modems on a cable network. Either a router or bridge, typically at the cable headend. Usually located in the cable provider's local office.

CMTS shared secret *See* shared secret.

configuration file A file containing configuration parameters for the device to be provisioned.

configuration generation The process of generating configurations at the RDU for devices and distributing them to the DPE. The configuration instructions are cached by the DPE and informed about action needed to be performed on the CPE.

CPE Customer premises equipment. Terminating equipment, such as telephones, computers, and modems, supplied and installed at a customer location.

D

Data Over Cable Service Interface Specification *See* DOCSIS.

DHCP Dynamic Host Configuration Protocol. Designed by the Internet Engineering Task Force (IETF) to reduce the amount of configuration that is required when using TCP/IP. DHCP allocates IP addresses to hosts. It also provides all the parameters that hosts require to operate and exchange information on the Internet network to which they are attached.

DNS Domain Name System. Handles the growing number of Internet users. DNS translates names, such as www.cisco.com, into Internet Protocol (IP) addresses, such as 192.168.40.0, so that computers can communicate with each other.

DOCSIS Data over cable service interface specification. DOCSIS defines functionality in cable modems involved in high-speed data distribution over cable television system networks.

DOCSIS Shared Secret Shared secret for communication between DOCSIS devices in a Prime Cable Provisioning deployment.

domain Portion of the DNS naming hierarchy tree that refers to general groupings of networks based on organization type or geography. The hierarchy is root, top- or first-level, and second-level domain.

DPE Device provisioning engine. The DPE caches device information. These distributed servers automatically synchronize with the RDU to obtain the latest configurations and provide Prime Cable Provisioning scalability.

DSTB Digital set-top box. A device that enables a television to become a user interface to the Internet and to receive and decode digital television signals.

dual stack A mode of DOCSIS cable modem operation in which the modem is manageable simultaneously via both IPv4 and IPv6 addresses.

DUID DHCP Unique Identifier. The primary device identifier in DHCPv6.

dynamic configuration file A dynamically created configuration file that uses template files to provide greater flexibility and security in the provisioning process.

E

eMTA Embedded MTA. A single node that contains both an MTA and a cable modem.

eSAFE embedded Service Application Functional Entity. A mixed-IP mode device that consists of an IPv6 embedded cable modem and an IPv4 eMTA.

F

FQDN Fully qualified domain name. FQDN is the full name of a system, rather than just its hostname. For example, cisco is a hostname and www.cisco.com is an FQDN.

G

giaddr DHCP gateway (relay agent) IP address. Sent in an RFC 2131 packet between the client and server.

I

Internet Protocol (IP, IPv4) Network layer for the TCP/IP protocol suite. Internet Protocol (version 4) is a connectionless, best-effort packet switching protocol. Defined in RFC 791.

IP address An IP address is a 32-bit number that identifies each sender or receiver of information that is sent in packets across the Internet.

IPv6 IP version 6. Replacement for the current version of IP (version 4). IPv6 includes support for flow ID in the packet header, which can be used to identify flows. Formerly called IPng (next generation).

F

FQDN Fully qualified domain name. FQDN is the full name of a system, rather than just its hostname. For example, cisco is a hostname and www.cisco.com is an FQDN.

K

KDC A key distribution center that implements limited Kerberos functionality. Used in the provisioning of PacketCable MTAs.

Kerberos A secret-key network authentication protocol that uses a choice of cryptographic algorithms for encryption and a centralized key database for authentication.

L

lease query Process by which a relay agent can request lease (and reservation) data directly from a DHCP server in addition to gleaning it from client/server transactions.

LVM In Linux kernel, LVM(Logical Volume Manager) allows you to create a volume group which can be further divided into logical volumes based on the requirement. The LVM also provides the flexibility to resize the volume group and logical volumes based on the dynamic memory usage

M

MAC address Standardized data link layer address that is required for every port or device that connects to a LAN. Other devices in the network use these addresses to locate specific ports in the network and to create and update routing tables and data structures. MAC addresses are 6 bytes long and are controlled by IEEE, Also known as hardware address, MAC-layer address, or physical address. Compare with *network address*.

MSO Multiple system operator. A company that operates more than one cable TV or broadband system.

MTA Multimedia Terminal Adapter. Equipment at the customer end of a broadband (PacketCable) network.

multiple service operator *See* MSO.

N

NAT Network address translation. Mechanism for reducing the need for globally unique IP addresses. NAT allows an organization with addresses that are not globally unique to connect to the Internet by translating those addresses into globally routeable address space. This is also known as Network Address Translation.

network address Network layer address referring to a logical, rather than a physical, network device. Also called a protocol address. Compare with *MAC address*.

network administrator Person responsible for operation, maintenance, and management of a network. *See also* network operator.

network operator Person who routinely monitors and controls a network, performing such tasks as reviewing and responding to alarms, monitoring throughput, configuring new circuits, and resolving problems. *See also* network administrator.

Network Time Protocol *See* NTP.

NR Cisco Prime Network Registrar. A software product that provides IP addresses, configuration parameters, and DNS names to DOCSIS cable modems and PCs, based on network and service policies.

NTP Network Time Protocol. NTP is a protocol designed to synchronize server clocks over a network.

O

option, DHCP DHCP configuration parameter and other control information stored in the options field of a DHCP message. DHCP clients determine what options get requested and sent in a DHCP packet. Network Registrar allows for creating option definitions as well as the option sets to which they belong.

Organizationally Unique Identifier (OUI) Assigned by the IEEE to identify the owner or ISP of a VPN.

P

PacketCable A CableLabs initiative for interoperable interface specifications to deliver advanced, real-time multimedia services over a two-way cable network. Built on top of cable modem infrastructure to enable a wide range of multimedia services, such as IP telephony, multimedia conferencing, interactive gaming, and general multimedia applications.

provisioning API A series of Prime Cable Provisioning functions that programs can use to make the operating system perform various functions.

provisioning groups Groupings of devices with a defined set of associated DPE and DHCP servers, based on either network topology or geography.

publishing The process of publishing provisioning information to an external datastore in real time. Publishing plug-ins must be developed to write data to a datastore.

PWS The Provisioning Web Service component of Prime Cable Provisioning provides a SOAP based web interface that supports provisioning operation similar to other API clients.

R

RBAC Role Based Access Control. Provides an approach to restrict access to system functions and resources to authorized users. Ensures better user management and security.

RDU Regional distribution unit. The primary server in the Prime Cable Provisioning provisioning system, it manages generation of device configurations, processes all API requests, and manages the Prime Cable Provisioning system.

realm The logical network served by a single Kerberos database and a set of Key Distribution Centers.

realm names By convention, realm names are generally all uppercase letters, to differentiate the realm from the Internet domain. *See* realm.

redundancy In internetworking, the duplication of devices, services, or connections so that, in the event of a failure, the redundant devices, services, or connections can perform the work of those that failed.

relay agent Device that connects two or more networks or network systems. In DHCP, a router on a virtual private network that is the IP helper for the DHCP server.

S

selection tags Selection tags associated with Network Registrar scopes. These tags define the clients and client classes associated with a scope.

shared secret A character string used to provide secure communication between two servers or devices.

single stack A mode of DOCSIS cable modem operation in which the modem operates with only one IP address type (v4 or v6) at any given time.

static configuration files These files are used as a configuration file for a device. For example, a static configuration file called *gold.cm* would identify the gold DOCSIS class of service. Prime Cable Provisioning treats this file type like any other binary file.

T

template files Text files that contain DOCSIS or PacketCable MTA options and values that, when used in conjunction with a DOCSIS or PacketCable MTA Class of Service, provide dynamic file generation.

TFTP Trivial File Transfer Protocol. Simplified version of File Transfer Protocol (FTP) that allows files to be transferred from one computer to another over a network.

TLV Type-Length-Value. A tuple within a DOCSIS or PacketCable configuration file.

tuple In programming languages, a tuple is an ordered set of values. Common uses for the tuple as a data type are: for passing a string of parameters from one program to another, or to represent a set of value attributes in a relational database.

Type Length Value *See* TLV.

U

uBr	Universal Broadband Router (such as the Cisco 7246 or 7223), which is the Cisco router implementation of a DOCSIS CMTS.

V

VoIP	VoIP is the ability to make telephone calls and send faxes over IP-based data networks with a suitable quality of service (QoS) and superior cost/benefit.

W

watchdog	A watchdog is a daemon process that is used to monitor, stop, start, and restart Prime Cable Provisioning component processes such as the RDU, Tomcat, and the SNMP agent.

X

XGCP	A Gateway Control Protocol used to pass data between networks. This includes M (for Media) GCP and S (Simple) GCP.

INDEX