Cisco Broadband Access Center Administrator Guide

Release 4.0
December 2007

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-12296-01

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

CCSP, the Cisco Square Bridge logo, Follow Me Browsing, and StackWise are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn, and iQuick Study are service marks of Cisco Systems, Inc.; and Access Registrar, Aironet, ASIST, BPX, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Empowering the Internet Generation, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, FormShare, GigaDrive, GigaStack, HomeLink, Internet Quotient, IOS, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, LightStream, Linksys, MeetingPlace, MGX, the Networkers logo, Networking Academy, Network Registrar, *Packet*, PIX, Post-Routing, Pre-Routing, ProConnect, RateMUX, ScriptShare, SlideCast, SMARTnet, StrataView Plus, SwitchProbe, TeleRouter, The Fastest Way to Increase Your Internet Quotient, TransPath, and VCO are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0501R)

Cisco Broadband Access Center Administrator Guide 4.0
© 2002 - 2007 Cisco Systems, Inc. All rights reserved.

CONTENTS

Preface xxiii

Audience xxiii

How This Guide Is Organized xxiv

Conventions xxv

Product Documentation xxvi

Related Documentation xxvi

Obtaining Documentation and Submitting a Service Request xxvii

CHAPTER 1 **Broadband Access Center Overview** 1-1

Introduction to BAC 1-1

Technologies and Features 1-1

Supported Technologies and Standards 1-2

DOCSIS High-Speed Data 1-2

PacketCable Voice Services 1-2

CableHome 1-3

Supported Standards 1-3

Supported Devices 1-4

Features and Benefits 1-4

CHAPTER 2 **Broadband Access Center Architecture** 2-1

Deployment 2-1

Architecture 2-2

Regional Distribution Unit 2-3

Generating Device Configurations 2-3

Service-Level Selection 2-4

Device Provisioning Engines 2-5

DPE Licensing 2-6

TACACS+ and DPE Authentication 2-7

TACACS+ Privilege Levels 2-7

TACACS+ Client Settings 2-7

DPE-RDU Synchronization 2-7

Synchronization Process 2-8

General DPE States 2-8

TFTP Server 2-9

ToD Server 2-9
DOCSIS Shared Secret 2-10
Network Registrar 2-11
DHCP 2-11
DNS 2-12
Lease Query 2-12
Key Distribution Center 2-12
BAC Process Watchdog 2-13
SNMP Agent 2-13
Administrator User Interface 2-13
Provisioning Concepts 2-14
Provisioning Groups 2-14
Static versus Dynamic Provisioning 2-15
Provisioning Group Capabilities 2-15
Logging Events 2-16

CHAPTER 3 **Configuration Workflows** 3-1
Component Workflows 3-1
RDU Workflow 3-1
DPE Workflow 3-2
Network Registrar Workflow 3-3
Technology Workflows 3-5
DOCSIS Workflow 3-5
PacketCable Workflows 3-6
PacketCable Secure 3-6
PacketCable Basic 3-8
CableHome Workflow 3-10

CHAPTER 4 **CPE Provisioning Overview** 4-1
Overview 4-1
Device Object Model 4-2
Discovered Data 4-4
Configuration Generation and Processing 4-6
Static Files versus Template Files 4-6
Property Hierarchy 4-7
Templates and Property Hierarchy 4-7
Custom Properties 4-8
Device Deployment in BAC 4-8

CPE Registration Modes 4-8
Standard Mode 4-8
Promiscuous Mode 4-9
Roaming Mode 4-9
Mixed Mode 4-9
CPE Provisioning Flows 4-9
Initial Configuration Workflows 4-9
Configuration Update Workflow 4-12
Promiscuous Access for Devices 4-13
Configuring Promiscuous Access 4-14
Promiscuous Access and Property Hierarchy 4-14
Generating Configurations for Promiscuous Devices 4-15
Properties for Configuring Promiscuous Policy 4-15

CHAPTER 5 **Configuration Templates Management** 5-1
Template Files–An Overview 5-1
Template Grammar 5-2
Comments 5-2
Includes 5-3
Options 5-3
Instance Modifier 5-4
OUI Modifier 5-5
SNMP VarBind 5-6
DOCSIS MIBs 5-7
PacketCable MIBs 5-7
CableHome MIBs 5-8
Macro Variables 5-8
SNMP TLVs 5-10
Adding SNMP TLVs Without a MIB 5-10
Adding SNMP TLVs With Vendor-Specific MIBs 5-11
Encoding Types for Defined Options 5-14
BITS Value Syntax 5-21
OCTETSTRING Syntax 5-21
Using the Configuration File Utility 5-21
Running the Configuration File Utility 5-22
Adding a Template to BAC 5-23
Converting a Binary File to a Template File 5-23
Testing Template Processing for a Local Template File 5-25
Testing Template Processing for an External Template File 5-26

Testing Template Processing for a Local Template File and Adding Shared Secret 5-27
Specifying Macro Variables at the Command Line 5-29
Specifying a Device for Macro Variables 5-30
Specifying Output to a Binary File 5-31
Viewing a Local Binary File 5-32
Viewing an External Binary File 5-33
Activating PacketCable Basic Flow 5-34
Generating TLV 43s for Multivendor Support 5-36

CHAPTER 6 **DOCSIS Configuration** 6-1
DOCSIS Workflow 6-1
DOCSIS DHCPv4 Workflow 6-2
DOCSIS DHCPv6 Workflow 6-5
Using MIBs with Dynamic DOCSIS Templates 6-9
BAC Features for DOCSIS Configurations 6-10
Dynamic Configuration TLVs 6-10
DPE TFTP IP Validation 6-11
Support for DOCSIS 1.0, 1.1, 2.0, and 3.0 6-11
Dynamic DOCSIS Version Selection 6-11
IPv6 Support 6-13
IPv6 Addressing 6-14
Single Stack versus Dual Stack 6-15
DHCP Options for IPv6 6-15
Attributes versus Options 6-15
Configuration Workflows for IPv6 6-19
Lease Query 6-19
Lease Query Autoconfiguration 6-19
Lease Query Source IP Address 6-20
Configuring Lease Query 6-20
Configuring BAC as Relay Agent for Lease Query 6-21
For IPv4 Lease Query 6-21
For IPv6 Lease Query 6-21
Enabling AIC Echo 6-22
Debugging Lease Query 6-22
IPv6 Lease Query Use Cases 6-23

CHAPTER 7 **PacketCable Voice Configuration** 7-1
PacketCable Secure eMTA Provisioning 7-1
BAC PacketCable Secure Provisioning Flow 7-1

KDC in Provisioning PacketCable Secure eMTAs 7-6
Default KDC Properties 7-7
KDC Certificates 7-9
KDC Licenses 7-9
Multiple Realm Support 7-10
Configuring the KDC for Multiple Realms 7-11
Authoring Template for Provisioning Devices in Multiple Realms 7-26
Configuring SRV Records in the Network Registrar DNS Server 7-28
Configuring SNMPv3 Cloning on the RDU and DPE for Secure Communication with PacketCable MTAs 7-29
Creating the Key Material and Generating the Key 7-29
PacketCable Basic eMTA Provisioning 7-30
PacketCable TLV 38 and MIB Support 7-31
SNMP v2C Notifications 7-31
Euro PacketCable 7-31
Euro-PacketCable MIBs 7-31
Configuring Euro-PacketCable MIBs 7-32

CHAPTER 8 **CableHome Configuration** 8-1
Non-Secure CableHome Provisioning Flow 8-1
Configuring CableHome 8-3
Configuring Network Registrar 8-3
Configuring the RDU 8-3
Configuring CableHome WAN-MAN 8-4
Configuring CableHome WAN-Data 8-4
Configuring the DPE 8-4

CHAPTER 9 **Managing Broadband Access Center** 9-1
BAC Process Watchdog 9-1
Using the BAC Process Watchdog from the Command Line 9-2
Administrator User Interface 9-3
Command-Line Interface 9-3
Accessing the DPE CLI from a Local Host 9-4
Accessing the DPE CLI from a Remote Host 9-4
SNMP Agent 9-4
BAC Tools 9-4

CHAPTER 10 **Monitoring Broadband Access Center** 10-1

Logging Events 10-1
Log Levels and Structures 10-1
Configuring Severity Levels 10-3
Rotating Log Files 10-3
RDU Logs 10-4
Viewing the *rdu.log* File 10-4
Viewing the *audit.log* File 10-4
Using the RDU Log Level Tool 10-4
Setting the RDU Log Level 10-6
Viewing the Current Log Level of RDU 10-6
DPE Log 10-7
Network Registrar Logs 10-8
Monitoring Servers Using SNMP 10-9
SNMP Agent 10-9
Using the snmpAgentCfgUtil.sh Tool 10-10
Adding a Host 10-10
Deleting a Host 10-11
Adding an SNMP Agent Community 10-11
Deleting an SNMP Agent Community 10-12
Starting the SNMP Agent 10-12
Stopping the SNMP Agent 10-13
Configuring an SNMP Agent Listening Port 10-13
Changing the SNMP Agent Location 10-13
Setting Up SNMP Contacts 10-14
Displaying SNMP Agent Settings 10-14
Specifying SNMP Notification Types 10-14
Monitoring Server Status 10-15
Using the Administrator User Interface 10-15
Using the DPE CLI 10-16

CHAPTER 11 **Understanding the Administrator User Interface** 11-1

Configuring the Administrator User Interface 11-1
Accessing the Administrator User Interface 11-2
Logging In 11-2
Logging Out 11-5
Understanding the Administrator User Interface Icons 11-6

CHAPTER 12 **Using the Administrator User Interface** 12-1

User Management 12-1
Administrator 12-1
Read/Write User 12-2
Read-Only User 12-2
Adding a New User 12-2
Modifying Users 12-3
Deleting Users 12-3
Device Management 12-4
Manage Devices Page 12-4
Searching for Devices 12-4
Device Management Controls 12-8
Viewing Device Details 12-9
Managing Devices 12-12
Adding Device Records 12-13
Modifying Device Records 12-14
Deleting Devices 12-14
Regenerating Device Configurations 12-14
Relating and Unrelating Devices 12-15
Resetting Devices 12-17
Node Management 12-17
Managing Node Types 12-17
Adding a Node Type 12-17
Modifying Node Types 12-18
Deleting Node Types 12-18
Managing Nodes 12-18
Adding a New Node 12-19
Searching for Devices in a Node 12-19
Modifying a Node 12-20
Deleting Nodes 12-20
Relating/Unrelating Node Types to Nodes 12-20
Viewing Node Details 12-21
Viewing Servers 12-21
Viewing Device Provisioning Engines 12-21
Viewing Network Registrar Extension Points 12-25
Viewing Provisioning Groups 12-27
Viewing Regional Distribution Unit Details 12-29

CHAPTER 13 **Configuring Broadband Access Center** 13-1

Configuring Class of Service 13-1
Adding a Class of Service 13-2
Modifying a Class of Service 13-3
Deleting a Class of Service 13-4
Configuring Custom Properties 13-5
Adding a Custom Property 13-5
Deleting a Custom Property 13-6
Configuring Defaults 13-6
CableHome WAN Defaults 13-7
Computer Defaults 13-7
DOCSIS Defaults 13-8
Network Registrar Defaults 13-9
PacketCable Defaults 13-11
RDU Defaults 13-11
System Defaults 13-12
STB Defaults 13-14
Configuring DHCP Criteria 13-15
Adding DHCP Criteria 13-15
Modifying DHCP Criteria 13-16
Deleting DHCP Criteria 13-16
Managing Files 13-17
Adding Files 13-18
Viewing Files 13-19
Replacing Files 13-21
Exporting Files 13-22
Deleting Files 13-22
Managing Licenses 13-22
Adding and Modifying a License 13-24
Deleting a License 13-25
Managing RDU Extensions 13-25
Writing a New Class 13-26
Installing RDU Custom Extension Points 13-28
Viewing RDU Extensions 13-28
Publishing Provisioning Data 13-28
Publishing Datastore Changes 13-29
Modifying Publishing Plug-In Settings 13-29
Automatic FQDN Generation 13-30

Automatically Generated FQDN Format 13-30
Properties for Automatically Generated FQDNs 13-31
FQDN Validation 13-31
Sample Automatic FQDN Generation 13-31

CHAPTER 14 **Support Tools and Advanced Concepts** 14-1
BAC Tools 14-1
Using the PKCert.sh Tool 14-2
Running the PKCert Tool 14-2
Creating a KDC Certificate 14-3
Validating the KDC Certificates 14-4
Setting the Log Level for Debug Output 14-5
Using the KeyGen Tool 14-8
Using the changeNRProperties.sh Tool 14-11
Using the disk_monitor.sh Tool 14-12

CHAPTER 15 **Database Management** 15-1
Understanding Failure Resiliency 15-1
Database Files 15-2
Database Storage File 15-2
Database Transaction Log Files 15-2
Automatic Log Management 15-2
Miscellaneous Database Files 15-3
Disk Space Requirements 15-3
Handling Out of Disk Space Conditions 15-3
Backup and Recovery 15-4
Database Backup 15-4
Database Recovery 15-5
Database Restore 15-6
Changing Database Location 15-7
RDU Database Migration 15-8

CHAPTER 16 **Troubleshooting Broadband Access Center** 16-1
Troubleshooting Checklist 16-1
Troubleshooting Devices by Device ID 16-2
Configuring Devices for Troubleshooting 16-3
Relating a Device to a Node 16-3
Viewing List of Devices in Diagnostics Mode 16-4

Troubleshooting Using the Diagnostics Tool 16-5
Using the startDiagnostics.sh Tool 16-5
Running startDiagnostics.sh in Interactive Mode 16-6
Running startDiagnostics.sh in Noninteractive Mode 16-7
Using the statusDiagnostics.sh Tool 16-8
Using the stopDiagnostics.sh Tool 16-9
Running stopDiagnostics.sh in Interactive Mode 16-9
Running stopDiagnostics.sh in Noninteractive Mode 16-9
Bundling Server State for Support 16-10
Troubleshooting DOCSIS Networks 16-10
Troubleshooting PacketCable eMTA Provisioning 16-11
Components 16-11
eMTA 16-11
DHCP Server 16-12
DNS Server 16-12
KDC 16-12
PacketCable Provisioning Server 16-12
Call Management Server 16-12
Key Variables 16-13
Certificates 16-13
Scope-Selection Tags 16-14
MTA Configuration File 16-14
Troubleshooting Tools 16-14
Logs 16-14
Ethereal, SnifferPro, or Other Packet Capture Tools 16-15
Troubleshooting Scenarios 16-15
Certificate Trust Hierarchy 16-18
Certificate Validation 16-19
MTA Device Certificate Hierarchy 16-20
MTA Root Certificate 16-20
MTA Manufacturer Certificate 16-21
MTA Device Certificate 16-21
MTA Manufacturer Code Verification Certificates 16-22
CableLabs Service Provider Certificate Hierarchy 16-22
CableLabs Service Provider Root Certificate 16-23
Service Provider CA Certificate 16-23
Local System CA Certificates 16-24
Operational Ancillary Certificates 16-25
Certificate Revocation 16-28

Code Verification Certificate Hierarchy 16-28
Common CVC Requirements 16-28
CableLabs Code Verification Root CA Certificate 16-29
CableLabs Code Verification CA Certificate 16-29
Manufacturer Code Verification Certificate 16-30
Service Provider Code Verification Certificate 16-30
Certificate Revocation Lists for CVCs 16-31

APPENDIX A **Alert and Error Messages** A-1
Message Format A-1
RDU Alerts A-2
DPE Alerts A-3
Watchdog Alerts A-4
Network Registrar Extension Point Alerts A-5

APPENDIX B **Option Support** B-1
DOCSIS Option Support B-1
PacketCable Option Support B-22
CableHome Option Support B-22

APPENDIX C **Mapping PacketCable DHCP Options to BAC Properties** C-1
Option 122 and BAC Property Comparison C-2
Option 177 and BAC Property Comparison C-2

APPENDIX D **Provisioning API Use Cases** D-1
How to Create an API Client D-1
Use Cases D-4
Self-Provisioned Modem and Computer in Fixed Standard Mode D-5
Adding a New Computer in Fixed Standard Mode D-8
Disabling a Subscriber D-10
Preprovisioning Modems/Self-Provisioned Computers D-12
Modifying an Existing Modem D-15
Unregistering and Deleting a Subscriber's Devices D-16
Self-Provisioning First-Time Activation in Promiscuous Mode D-20
Bulk Provisioning 100 Modems in Promiscuous Mode D-23
Preprovisioning First-Time Activation in Promiscuous Mode D-25
Replacing an Existing Modem D-27
Adding a Second Computer in Promiscuous Mode D-29

Self-Provisioning First-Time Activation with NAT D-29
Adding a New Computer Behind a Modem with NAT D-30
Move Device to Another DHCP Scope D-30
Log Device Deletions Using Events D-31
Monitoring an RDU Connection Using Events D-32
Logging Batch Completions Using Events D-33
Getting Detailed Device Information D-33
Searching Using the Device Type D-38
Searching for Devices Using Vendor Prefix or Class of Service D-39
Preprovisioning PacketCable eMTA D-40
SNMP Cloning on PacketCable eMTA D-42
Incremental Provisioning of PacketCable eMTA D-44
Preprovisioning DOCSIS Modems with Dynamic Configuration Files D-46
Optimistic Locking D-48
Temporarily Throttling a Subscriber's Bandwidth D-50
Preprovisioning CableHome WAN-MAN D-51
CableHome with Firewall Configuration D-52
Retrieving Device Capabilities for CableHome WAN-MAN D-54
Self-Provisioning CableHome WAN-MAN D-55

APPENDIX E **FAQs on Provisioning Broadband Access Center** E-1

BAC Configuration E-1
How do I enable or disable Network Registrar extensions? E-1
How do I enable tracing for Network Registrar extensions? E-2
Why is my DPE server registration failing? E-2

IPv6 Configuration E-2
How do I enable provisioning in IPv6 for the DPE? E-3
How do I configure an IPv4 interface for provisioning? E-3
DPE is configured for IPv6 provisioning, but BAC does not provision IPv6 DOCSIS 3.0 devices. Why? E-4
When searching for all devices using their MAC address, some IPv6 devices do not show up. Why? E-4
How do I enable IPv6 on an interface? E-4
How do I configure IPv6 on a loopback interface? E-4
How do I disable a stateful DHCPv6 client on Solaris 10? E-5
How do I assign a static IP address to an interface? E-5

CMTS Configuration E-5
How do I know that both cable line cards are using the cable bundle 1? E-5
Is there an IPv6 cable-helper address that I can use? E-5
How do I configure multiple IPv6 subnets similar to IPv4 primary and secondary IPv4 subnets? E-5

How do I view the list of IPv6 modems on the CMTS? **E-6**
How do I configure a CMTS interface to accept only IPv6 single stack? **E-6**
What does the modem state init(x) mean? **E-6**

GLOSSARY

INDEX

FIGURES

Figure 2-1 Deployment Using BAC **2-1**

Figure 4-1 Device Object Model **4-2**

Figure 4-2 Workflow of Initial Device Configuration – Preprovisioned Mode **4-10**

Figure 4-3 Workflow of Initial Device Configuration – Self-Provisioned Mode **4-11**

Figure 4-4 Workflow of Device Configuration Update **4-13**

Figure 6-1 DOCSIS DHCPv4 Provisioning Flow **6-2**

Figure 6-2 DOCSIS DHCPv6 Provisioning Flow **6-5**

Figure 7-1 Embedded-MTA Secure Power-On Provisioning Flow **7-2**

Figure 7-2 Provisioning Motorola MTA–Device Details **7-21**

Figure 7-3 Provisioning Linksys MTA–Device Details **7-22**

Figure 7-4 Provisioning SA MTA–Device Details **7-23**

Figure 8-1 Non-Secure CableHome Flow **8-1**

Figure 11-1 Login Page **11-4**

Figure 11-2 Change Password Screen **11-4**

Figure 11-3 Main Menu Page **11-5**

Figure 12-1 Manage Devices Page **12-7**

Figure 13-1 Sample Binary File Content **13-20**

Figure 13-2 Sample JAR File Content **13-21**

Figure 13-3 Manage License Keys Page **13-24**

Figure 16-1 PacketCable Certificate Hierarchy **16-18**

TABLES

Table 2-1 Reasons for Device Access as Determined by Service-Level Selection Extension **2-5**
Table 2-2 TACACS+ Service Levels **2-7**
Table 2-3 Related DPE States **2-8**
Table 2-4 Static Provisioning versus Dynamic Provisioning **2-15**
Table 3-1 RDU Configuration Workflow **3-1**
Table 3-2 DPE Configuration Workflow for IPv4 **3-2**
Table 3-3 DPE Configuration Workflow for IPv6 **3-3**
Table 3-4 Network Registrar Workflow for DHCPv4 **3-4**
Table 3-5 Network Registrar Workflow for DHCPv6 **3-4**
Table 3-6 DOCSIS Workflow **3-6**
Table 3-7 PacketCable Secure Workflow **3-7**
Table 3-8 PacketCable Basic Workflow **3-9**
Table 3-9 CableHome Workflow **3-11**
Table 4-1 Device Object Relationships **4-3**
Table 4-2 Data Discovered from IPv4 Devices **4-4**
Table 4-3 Data Discovered from IPv6 Devices **4-5**
Table 4-4 Configuring Promiscuous Access for Devices **4-14**
Table 4-5 Properties for Enabling Promiscuous Access **4-16**
Table 4-6 Promiscuous Access–Read-Write Properties for Class of Service **4-18**
Table 4-7 Promiscuous Access–Read-Write Properties for DHCP Criteria **4-18**
Table 4-8 Promiscuous Access–Read-Only Properties **4-19**
Table 5-1 Template Grammar **5-2**
Table 5-2 SNMP Variable Types **5-10**
Table 5-3 Defined Option Encoding Types **5-14**
Table 6-1 DOCSIS DHCPv4 Workflow Description **6-3**
Table 6-2 DOCSIS DHCPv6 Workflow Description **6-6**
Table 6-3 Properties for Network Registrar Extensions Before BAC 4.0 **6-17**
Table 6-4 Properties for DHCPv4 Network Registrar Extensions in BAC 4.0 **6-17**
Table 6-5 Properties for DHCPv6 Network Registrar Extensions in BAC 4.0 **6-18**
Table 6-6 Lease Query Address for Binding **6-20**

Table 7-1 PacketCable Secure eMTA Provisioning **7-3**
Table 7-2 KDC Logging Levels **7-8**
Table 7-3 PacketCable Certificates **7-10**
Table 7-4 Directory Structure for Multiple Realms **7-11**
Table 7-5 PacketCable Basic eMTA Provisioning **7-30**
Table 8-1 CableHome Provisioning Workflow **8-2**
Table 9-1 BAC CLI Commands **9-2**
Table 9-2 BAC Tools **9-5**
Table 10-1 Severity Levels **10-2**
Table 10-2 Logging Levels **10-5**
Table 10-3 DHCP Server Extension Trace Levels **10-8**
Table 10-4 BAC-Supported MIBs **10-9**
Table 11-1 Browser Platform Support **11-3**
Table 11-2 Administrator User Interface Icons **11-6**
Table 12-1 Searches Supported for Device Management **12-6**
Table 12-2 View Device Details Page **12-9**
Table 12-3 View Device Provisioning Engines Details Page **12-21**
Table 12-4 View Network Registrar Extension Point Details Page **12-25**
Table 12-5 View Provisioning Groups Details Page **12-27**
Table 12-6 View Regional Distribution Unit Details Page **12-29**
Table 13-1 Manage Class of Service Page **13-2**
Table 13-2 Configure Defaults–CH WAN-Data/CH WAN-MAN Defaults Page **13-7**
Table 13-3 Configure Defaults–DOCSIS Defaults Page **13-8**
Table 13-4 Configure Defaults–Network Registrar Defaults Page **13-9**
Table 13-5 Configure Defaults–PacketCable Defaults Page **13-11**
Table 13-6 Configure Defaults–RDU Defaults Page **13-12**
Table 13-7 Configure System Defaults Page **13-13**
Table 13-8 Configure Defaults–STB Defaults Page **13-14**
Table 13-9 View Files Page **13-18**
Table 13-10 Required RDU Extension Points **13-26**
Table 13-11 Modify Publishing Plug-Ins Page **13-29**
Table 14-1 BAC Tools **14-1**
Table 16-1 Troubleshooting Checklist **16-1**
Table 16-2 Sample Relate/Unrelate Process **16-3**
Table 16-3 Troubleshooting Scenarios **16-15**

Table 16-4 MTA Root Certificate **16-20**
Table 16-5 MTA Manufacturer Certificates **16-21**
Table 16-6 MTA Device Certificates **16-22**
Table 16-7 CableLabs Service Provider Root Certificates **16-23**
Table 16-8 CableLabs Service Provider CA Certificates **16-24**
Table 16-9 Local System CA Certificates **16-24**
Table 16-10 KDC Certificates **16-25**
Table 16-11 DF Certificates **16-26**
Table 16-12 PacketCable Server Certificates **16-27**
Table 16-13 CableLabs Code Verification Root CA Certificates **16-29**
Table 16-14 CableLabs Code Verification CA Certificates **16-29**
Table 16-15 Manufacturer Code Verification Certificates **16-30**
Table 16-16 Service Provider Code Verification Certificates **16-31**
Table A-1 Severity Levels for Alert Messages **A-1**
Table A-2 RDU Alerts **A-2**
Table A-3 DPE Alerts **A-3**
Table A-4 Process Watchdog Alerts **A-4**
Table A-5 Network Registrar Extension Alerts **A-6**
Table B-1 DOCSIS Options and Version Support **B-1**
Table B-2 PacketCable MTA Options **B-22**
Table B-3 CableHome Options and Version Support **B-22**
Table C-1 DHCP Option 122 to BAC Property Comparison **C-2**
Table C-2 DHCP Option 177 to BAC Property Comparison **C-2**
Table E-1 Cable Modem States **E-6**

Preface

Welcome to the *Cisco Broadband Access Center Administrator Guide 4.0*. This guide describes concepts and configurations related to Cisco Broadband Access Center, referred to as BAC throughout this guide.

The preface provides an outline of other chapters in this guide, details information about related documents that support this BAC release, and demonstrates the styles and conventions used in the guide.

Note Use this guide along with the documentation listed in Product Documentation, page xxvi, and Related Documentation, page xxvi.

This preface describes:

- Audience, page xxiii
- How This Guide Is Organized, page xxiv
- Conventions, page xxv
- Product Documentation, page xxvi
- Related Documentation, page xxvi
- Obtaining Documentation and Submitting a Service Request, page xxvii

Audience

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

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

How This Guide Is Organized

The major sections of this guide are:

Broadband Access Center Overview	Describes BAC, the technologies and standards that this BAC release supports, and features and benefits.
Broadband Access Center Architecture	Describes the system architecture implemented in this BAC release.
Configuration Workflows	Provides workflows to follow when configuring BAC.
CPE Provisioning Overview	Provides an overview of CPE provisioning and describes key concepts that BAC supports.
Configuration Templates Management	Describes the configuration templates that BAC supports and how to develop template files.
DOCSIS Configuration	Describes how to bring a BAC DOCSIS deployment into service.
PacketCable Voice Configuration	Describes how to bring a PacketCable voice deployment into service.
CableHome Configuration	Describes how to bring a CableHome deployment, using the non-secure (DHCP) version, into service.
Managing Broadband Access Center	Describes the various subcomponents that help manage BAC.
Monitoring Broadband Access Center	Describes how to monitor the BAC servers in a deployment.
Understanding the Administrator User Interface	Describes how to access BAC from the administrator user interface.
Using the Administrator User Interface	Describes how to perform administration activities, including searching for and viewing device information, from the administrator user interface.
Configuring Broadband Access Center	Describes how to perform configuration activities from the administrator user interface.
Support Tools and Advanced Concepts	Describes BAC tools intended to help configure, maintain speed, and improve the installation, deployment, and use of BAC.
Database Management	Describes how to manage and maintain the RDU database.
Troubleshooting Broadband Access Center	Describes how to troubleshoot the provisioning process for PacketCable embedded Multimedia Terminal Adapters (eMTAs).
Alert and Error Messages	Lists and describes BAC x messages.
Option Support	Lists the technology-specific options that BAC supports for each technology version.

Mapping PacketCable DHCP Options to BAC Properties	Identifies the mapping of BAC properties to the PacketCable DHCP options used for PacketCable provisioning.
Provisioning API Use Cases	Presents a series of common provisioning API use cases that can be used to model typical service-provider workflows.
FAQs on Provisioning Broadband Access Center	Lists frequently asked questions about BAC configuration and provisioning.
Glossary	Defines terminology used in this guide and generally applicable to the technologies being discussed.

Conventions

This document uses the following conventions:

Item	Convention
Commands and keywords	**boldface** font
Variables for which you supply values	*italic* font
Displayed session and system information	`screen` font
Information you enter	**`boldface screen`** `font`
Variables you enter	*`italic screen`* font
Menu items and button names	**boldface** font
Selecting a menu item in paragraphs	**Option > Network Preferences**
Selecting a menu item in tables	Option > Network Preferences

Caution Means *be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

Note Means *take note*. Notes contain helpful suggestions or references to material not covered in the publication.

Tip Means *a helpful hint*. The description can present an optimum action to take.

Product Documentation

Note We sometimes update the printed and electronic documentation after original publication. Therefore, you should also review the documentation on Cisco.com for any updates.

Table 1 describes the documentation that is available for this BAC release.

Table 1 Product Documentation

Document Title	Available Format
Release Notes for Cisco Broadband Access Center 4.0	• PDF file on the product CD-ROM • On Cisco.com: http://cisco.com/en/US/products/sw/netmgtsw/ps529/prod_release_notes_list.html
Installation and Setup Guide for Cisco Broadband Access Center, Release 4.0	• PDF file on the product CD-ROM • On Cisco.com: http://cisco.com/en/US/products/sw/netmgtsw/ps529/prod_installation_guides_list.html
Cisco Broadband Access Center Administrator Guide, Release 4.0	• PDF file on the product CD-ROM • On Cisco.com: http://cisco.com/en/US/products/sw/netmgtsw/ps529/prod_maintenance_guides_list.html
Cisco Broadband Access Center DPE CLI Reference, Release 4.0	• PDF file on the product CD-ROM • On Cisco.com: http://cisco.com/en/US/products/sw/netmgtsw/ps529/prod_command_reference_list.html

Related Documentation

Note We sometimes update the printed and electronic documentation after original publication. Therefore, you should also review the documentation on Cisco.com for any updates.

Table 2 describes the related documentation that is available for this BAC release.

Table 2 Related Documentation

Document Title	Available Format
Release Notes for Cisco Network Registrar 7.0	On Cisco.com: http://cisco.com/en/US/products/sw/netmgtsw/ps1982/prod_release_notes_list.html
Installation Guide for Cisco Network Registrar, Release 7.0	On Cisco.com http://cisco.com/en/US/products/sw/netmgtsw/ps1982/prod_installation_guides_list.html
User Guide for Cisco Network Registrar, Release 7.0	On Cisco.com: http://cisco.com/en/US/products/sw/netmgtsw/ps1982/products_user_guide_list.html
CLIFrame.html in the */docs* directory	On Cisco.com: http://cisco.com/en/US/products/sw/netmgtsw/ps1982/prod_command_reference_list.html
Quick Start Guide for Cisco Network Registrar, Release 7.0	On Cisco.com: http://cisco.com/en/US/products/sw/netmgtsw/ps1982/prod_installation_guides_list.html

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Subscribe to the *What's New in Cisco Product Documentation* as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS version 2.0.

CHAPTER 1

Broadband Access Center Overview

This chapter gives an overview of Cisco Broadband Access Center, Release 4.0, hereafter referred to as BAC.

This chapter details:

- Introduction to BAC, page 1-1
- Technologies and Features, page 1-1

Introduction to BAC

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

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

BAC 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.

BAC 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 BAC operates. You can use the provisioning API to register devices in BAC, assign device configurations, and configure the entire BAC provisioning system.

The predominant feature of this release is support for provisioning and managing of CPE that is compliant with the DOCSIS 3.0 specification. With IP version 6 (IPv6) being a large subset of DOCSIS 3.0, this release supports DHCPv6 and DNSv6.

Technologies and Features

This section describes the technologies and features that this BAC release supports.

- Supported Technologies and Standards, page 1-2
- Supported Devices, page 1-4
- Features and Benefits, page 1-4

Supported Technologies and Standards

BAC incorporates support for many technologies to provide provisioning services for your network. These technologies include:

- DOCSIS high-speed data
- PacketCable voice service, both Secure and Basic workflows
- Non-secure CableHome provisioning

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, 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.

This BAC release, 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.

BAC supports the Secure and Basic variants of PacketCable. PacketCable Basic and PacketCable Secure are much the same, except for reduced security found in the Basic variant.

Note BAC currently supports versions 1.0, 1.1, and 1.5 of the PacketCable 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

The BAC servers comply with these applicable Requests for Comments (RFCs), protocols, standards, and Internet Engineering Task Force (IETF) drafts.

- IPv6—Complies with RFC 2460 (IPv6 specification), 2461 (Neighbor Discovery protocol), 2462 (Stateless Address Autoconfiguration) 2463 (Internet Control Message Protocol–ICMP), 3513 (Addressing Architecture).
- 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.
- 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) and 2349 (TFTP Block Size Option).

Additionally, BAC complies with these applicable CableLabs and Comcast standards:

- DOCSIS 3.0 Specifications:
  - CM-SP-SECv3.0-I04-070518
  - CM-SP-PHY3.0-I04-070518
  - CM-SP-MULPIv3.0-I04-070518
  - CM-SP-OSSIv3.0-I03-070518
- DOCSIS 2.0 Specification CM-SP-RFI2.0-I12-071206
- DOCSIS L2VPN Specification CM-SP-L2VPN-I06-071206
- PacketCable MTA Device Provisioning Specification PKT-SP-PROV1.5-I03-070412
- CableHome CH-SP-CH1.0-I05-030801
- COMCAST-SP-RNG-200-ProvOSS-I04-070102
- OpenCable specification OC-SP-HOST2.0-CFR-I13-070323

Supported Devices

In this release, BAC supports provisioning and managing of:

- IPv6 devices, which include:
  - Cable modems compliant with DOCSIS 3.0
  - Computers
  - Set-top boxes (STBs)
- Video STBs, specifically the RNG-200 STB, which is based on the evolving 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.

BAC 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

Features and Benefits

BAC lets multiple service operators (MSOs) meet the rapidly changing demands for data over cable services. Using BAC, you can realize these benefits:

- Massive scalability. The BAC Regional Distribution Unit can support up to 60 million devices, and a provisioning group can support 2 million devices, of which 500,000 can be Secure PacketCable devices.
- Easy integration with back-end systems, via BAC mechanisms such as:
  - The BAC Java API, which can be used to perform all provisioning and management operations.
  - The BAC publishing extensions, which are useful in writing RDU data into another database.
  - The SNMP agent, which simplifies integration for monitoring BAC.
  - 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 via:
  - Provisioning group properties on the property hierarchy—Enhances the flexibility that the BAC property hierarchy provides by including the properties of a device’s provisioning group.
  - Provisioning group capabilities—Allows you to control the device type support that must be enabled for the provisioning groups in your deployment.

- Increased security via:
  - 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 BAC 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 password that you use to log in to the administrator user interface must have at least 8 characters.
  - HTTP over SSL (HTTPS), which provides access to the administrator user interface using a secure SSL connection.
- Enhanced troubleshooting and diagnostics for:
  - Device troubleshooting to provide detailed records of device interactions with BAC 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 BAC servers. This release also provides many scripts to collect server and system configuration data that may be required for support escalations. You can use additional scripts to bundle the diagnostics data for support.

CHAPTER 2

Broadband Access Center Architecture

This chapter describes the system architecture implemented in this Cisco Broadband Access Center (BAC) release.

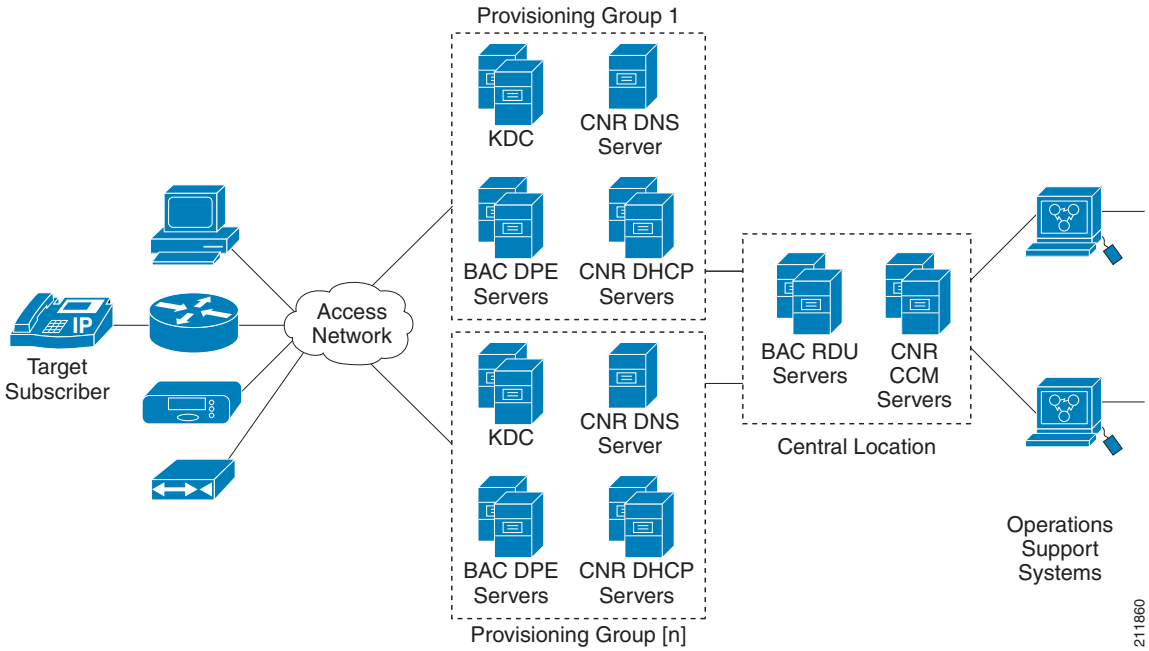
This chapter describes:

- Deployment, page 2-1
- Architecture, page 2-2
- Logging Events, page 2-16

Deployment

Figure 2-1 represents a typical, fully redundant deployment in a BAC network.

Figure 2-1 Deployment Using BAC



Architecture

This section describes the basic BAC architecture, such as:

- Regional Distribution Unit (RDU) that provides:
  - The authoritative data store of the BAC system.
  - Support for processing application programming interface (API) requests.
  - Monitoring of the system's overall status and health.

  See Regional Distribution Unit, page 2-3, 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 Device Provisioning Engines, page 2-5, for additional information.
- BAC API that provides total client control over system capabilities.
- Cisco Network Registrar servers that provide:
  - Dynamic Host Configuration Protocol (DHCP).
  - Domain Name System (DNS).

  See Network Registrar, page 2-11, 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 2-14, for additional information.
- A Kerberos server that authenticates PacketCable Multimedia Terminal Adapters (MTAs). See Key Distribution Center, page 2-12, for additional information.
- The BAC process watchdog that provides:
  - Administrative monitoring of all critical BAC processes.
  - Automated process-restart capability.
  - Ability to start and stop BAC component processes.

  See BAC Process Watchdog, page 9-1, for additional information.

- An SNMP agent that provides:
  - Third-party management systems.
  - SNMP version v2.
  - SNMP Notification.

  See SNMP Agent, page 9-4, for additional information.
- An administrator user interface that supports:
  - Adding, deleting, modifying and searching for devices.
  - Configuring of global defaults and defining of custom properties.

  See Administrator User Interface, page 2-13, for additional information.

Regional Distribution Unit

The RDU is the primary server in the BAC provisioning system. You must install the RDU on a server running the Solaris operating system.

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 BAC functions
- Managing the BAC system

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

Currently, BAC supports one RDU per installation. To provide failover support, we recommend using clustering software from Veritas or Sun. We also recommend using RAID (Redundant Array of Independent Disks) shared storage in such a setup.

The following sections describe these RDU concepts:

- Generating Device Configurations, page 2-3
- Service-Level Selection, page 2-4

Generating Device Configurations

When a device boots, it requests a configuration from BAC 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.

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 BAC 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.

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 when 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, BAC uses the default Class of Service and DHCP Criteria assigned for the device type.
- Promiscuous—For devices granted promiscuous access, BAC obtains the Class of Service and DHCP Criteria from the relay agent that the device is behind.
- Registered—For devices granted registered access, BAC 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**. For additional information, see Configuring Defaults, page 13-6. 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 administrator user interface on the View Device Details page.

Table 2-1 describes these reasons and the type of access granted in that case.

Table 2-1 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.			✓
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.

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.

BAC 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.

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.
- Integrates with Network Registrar.
- Provisions voice-technology services.

You must install the DPE on a server that runs the Solaris operating system. 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, refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

Note During installation, you must configure each DPE for the:

- Name of the provisioning group to which the DPE belongs. This name determines the logical group of devices that the DPE services.
- IP address and port number of the RDU.

For important information related to DPEs, see:

- DPE Licensing, page 2-6
- TACACS+ and DPE Authentication, page 2-7
- DPE-RDU Synchronization, page 2-7
- TFTP Server, page 2-9
- ToD Server, page 2-9

Also, familiarize yourself with the information described in Provisioning Concepts, page 2-14.

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 administrator user interface 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 administrator user interface. 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 13-22.

TACACS+ and DPE 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 can support many users, with each username and login and enable 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. The TACACS+ client specifies a certain service level that is configured for the user.

Table 2-2 identifies the two service levels used to authorize DPE user access.

Table 2-2 TACACS+ Service Levels

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

TACACS+ Client Settings

TACACS+ uses a number of properties that are configured from the CLI. For information on commands related to TACACS+, refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

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

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.

General DPE States

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 2-3 describes some other states that the DPE may be in from time to time.

Table 2-3 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 Viewing Device Provisioning Engines, page 12-21.
- From the DPE CLI using the **show dpe** command. Refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

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.

The TFTP server is located in a home directory that is used for local file-system access. The local files are stored in the *BPR_DATA/dpe/tftp* directory. In this release, all deliverable TFTP files are precached at 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 Broadband Access Center DPE CLI Reference 4.0*.

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 12-9.

ToD Server

The integrated time of day (ToD) server in BAC 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 Broadband Access Center DPE CLI Reference 4.0*.

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 12-9.

DOCSIS Shared Secret

BAC 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 BAC.

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.

Network Registrar

Network Registrar provides the DHCP and DNS functionality in BAC. The DHCP extension points on Network Registrar integrate BAC with Network Registrar. Using these extensions, BAC 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 Network Registrar, refer to the *User Guide for Cisco Network Registrar 7.0*; *CLIFrame.html* in the */docs* directory*;* and *Installation Guide for Cisco Network Registrar*, *7.0*.

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 BAC 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 autoconfiguration, which enables IPv6 hosts to configure their own addresses using a local IPv6 router. DHCPv6 represents the stateful autoconfiguration 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 autoconfiguration protocol
- Improved support for options and extensions
- Relay agent functionality
- 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.

DNS

The DNS server contains information on hosts throughout the network, such as IP address hostnames. 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.

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 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. In this release, you can configure the RDU to use a specific interface and source port.

For detailed information on lease query support in this BAC release, see Lease Query, page 6-19.

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.

You must install the KDC on a server that runs the Solaris operating system.

The certificates used to authenticate the KDC are not shipped with BAC. 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 the PKCert.sh Tool, page 14-2.

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 7-9.

The KDC has several default properties that are populated during a BAC installation into the *BPR_HOME/kdc/solaris/kdc.ini* properties file. You can edit this file to change values as operational requirements dictate. For detailed information, see Default KDC Properties, page 7-7.

The KDC also supports the management of multiple realms. For details on configuring additional realms, see Multiple Realm Support, page 7-10.

BAC Process Watchdog

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

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

See BAC Process Watchdog, page 9-1, for additional information on how to manage the monitored processes.

SNMP Agent

BAC provides basic SNMP v2-based monitoring of the RDU and DPE servers. The BAC 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 10-9) or via the API.
- On the DPE, using the **snmp-server** CLI commands. Refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

Administrator User Interface

The BAC administrator user interface is a web-based application for central management of the BAC 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

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

- Chapter 11, "Understanding the Administrator User Interface," describes how to access and configure the BAC administrator user interface.
- Chapter 12, "Using the Administrator User Interface," provides instructions for performing administrative activities involving the monitoring of various BAC components.
- Chapter 13, "Configuring Broadband Access Center," describes tasks that you perform to configure BAC.

Provisioning Concepts

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

- Provisioning Groups, page 2-14
- Static versus Dynamic Provisioning, page 2-15
- Provisioning Group Capabilities, page 2-15

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 BAC 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

BAC 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 BAC in two ways: static provisioning and dynamic provisioning.

During static provisioning, you enter static configuration files into the BAC system. These configuration files are then delivered via TFTP to the specific device to generate its configuration. BAC treats static configuration files like any other binary file.

During dynamic provisioning, you use templates, 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 2-4 describes the impact of static and dynamic provisioning using the corresponding files.

Table 2-4 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 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, a provisioning group cannot provision a PacketCable MTA in Secure mode if its DPEs are not configured to support this functionality.

In previous BAC releases, each DPE in a provisioning group registered what it was capable of supporting with the RDU at startup. This information was combined with that of other DPEs in the provisioning group to determine the device types that the group could support. The servers registered their low-level capabilities and if those capabilities were enabled or disabled. After server registration, the provisioning group was automatically enabled to support the device types it was capable of supporting. However, in this BAC release, you must enable device support manually:

- From the administrator user interface on the Provisioning Group Details Page. See Viewing Provisioning Groups, page 12-27.
- From the API using the *ProvGroupCapabilitiesKeys* constants. For details, refer to the API Javadoc.

Logging Events

Logging of events is performed at the RDU and the DPE, 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 administrator user interface.

For detailed information on log levels and structures, and how log files are numbered and rotated, see Log Levels and Structures, page 10-1.

CHAPTER 3

Configuration Workflows

This chapter is divided into two sections, each of which defines the process to follow when configuring Cisco Broadband Access Center (BAC) components to support various technologies. These sections are:

- Component Workflows, page 3-1
- Technology Workflows, page 3-5

Note You can also use the application programming interface (API) to perform all the configuration tasks outlined in this chapter. Refer to the 4.0 API Javadoc for more details.

Component Workflows

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

You must configure the BAC components in the order specified below.

1. RDU Workflow, page 3-1
2. DPE Workflow, page 3-2
3. Network Registrar Workflow, page 3-3

RDU Workflow

Table 3-1 identifies the workflow to follow when configuring the RDU.

Table 3-1 RDU Configuration Workflow

	Task	Refer to...
Step 1	Configure the system syslog service for use with BAC.	*Installation and Setup Guide for Cisco Broadband Access Center 4.0*
Step 2	Access the BAC administrator user interface.	Accessing the Administrator User Interface, page 11-2
Step 3	Change the login password.	Accessing the Administrator User Interface, page 11-2
Step 4	Add the license file.	Managing Licenses, page 13-22

Table 3-1 RDU Configuration Workflow (continued)

	Task	Refer to...
Step 5	Back up the RDU database.	Backup and Recovery, page 15-4
Step 6	Configure the RDU SNMP agent.	Using the snmpAgentCfgUtil.sh Tool, page 10-10
Step 7	Configure the default severity log level, which is the Notification level.	Using the RDU Log Level Tool, page 10-4
Step 8	Enable provisioning-group capabilities for IPv4 or IPv6.	Viewing Provisioning Groups, page 12-27

DPE Workflow

You perform the tasks described in this workflow only after configuring the tasks described in Table 3-1. You can configure the DPE to support:

- IPv4. See Table 3-2.
- IPv6. See Table 3-3.

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

Table 3-2 identifies the workflow to follow when configuring the DPE for IPv4.

Table 3-2 DPE Configuration Workflow for IPv4

	Task	Refer to...
Step 1	Configure the system syslog service for use with BAC.	*Installation and Setup Guide for Cisco Broadband Access Center 4.0*
Step 2	Change the passwords.	The **password** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 3	Configure the provisioning interface.*	The **interface ip** *ipv4_address* **provisioning** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 4	Configure the provisioning FQDN.	The **interface ip** *ipv4_address* **provisioning fqdn** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 5	Configure the interface that communicates with Cisco Network Registrar extensions.	The **interface ip** *ipv4_address* **pg-communication** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 6	Configure the BAC shared secret.*	The **dpe shared-secret** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 7	Configure the DPE to connect to the RDU.*	The **dpe rdu-server port** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 8	Configure the Network Time Protocol (NTP).	Solaris documentation for configuration information

Table 3-2 DPE Configuration Workflow for IPv4 (continued)

	Task	Refer to...
Step 9	Configure the primary provisioning group.*	The **dpe provisioning-group primary** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 10	Configure the DPE SNMP agent.	The SNMP agent commands in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
	Note You can configure the SNMP agent using either the DPE command-line interface or the **snmpAgentCfgUtil.sh** tool (see Using the snmpAgentCfgUtil.sh Tool, page 10-10).	
Step 11	Verify that you are connected to RDU.	Viewing Servers, page 12-21
Step 12	Enable provisioning-group capabilities for v4.	Viewing Provisioning Groups, page 12-27

Table 3-3 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 3-2, then additionally complete the steps described in this table.

Table 3-3 DPE Configuration Workflow for IPv6

	Task	Refer to...
Step 1	Configure the provisioning interface.*	The **interface ip** *ipv6_address* **provisioning** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 2	Configure the provisioning FQDN.	The **interface ip** *ipv6_address* **provisioning fqdn** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 3	Enable TFTP.	The **service tftp** *1..1* **ipv6 enabled true** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 4	Enable ToD.	The **service tod** *1..1* **ipv6 enabled true** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 5	Reload the DPE.	The **dpe reload** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 6	Enable provisioning-group capabilities for v6.	Viewing Provisioning Groups, page 12-27

Network Registrar Workflow

You perform the activities described in this workflow only after configuring the tasks described in Table 3-2.

Caution The BAC DHCP option settings always replace any DHCP option values set within Cisco Network Registrar.

To configure Network Registrar for:

- DHCPv4, see Table 3-4.
- DHCPv6, see Table 3-5.

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

Table 3-4 identifies the workflow to follow when configuring Network Registrar for DHCPv4.

Table 3-4 Network Registrar Workflow for DHCPv4

	Task	Refer to...
Step 1	Validate the Network Registrar extensions.	*Installation and Setup Guide for Cisco Broadband Access Center 4.0*
Step 2	Configure the system syslog service for use with BAC.	*Installation and Setup Guide for Cisco Broadband Access Center 4.0*
Step 3	Configure client classes/selection tags that match those defined in the RDU. *	*User Guide for Cisco Network Registrar 7.0*
Step 4	Configure policies.*	*User Guide for Cisco Network Registrar 7.0*
Step 5	Configure scopes.*	*User Guide for Cisco Network Registrar 7.0*
Step 6	Back up the Network Registrar database.	*User Guide for Cisco Network Registrar 7.0*
Step 7	Verify that you are connected to the correct RDU.	Viewing Servers, page 12-21
Step 8	Reload the DHCP server.	*User Guide for Cisco Network Registrar 7.0*

Table 3-5 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 3-5 Network Registrar Workflow for DHCPv6

	Task	Refer to...
Step 1	Validate the Network Registrar extensions.	*Installation and Setup Guide for Cisco Broadband Access Center 4.0*
Step 2	Configure the system syslog service for use with BAC.	*Installation and Setup Guide for Cisco Broadband Access Center 4.0*
Step 3	Configure client classes/selection tags that match those defined in the RDU. *	*User Guide for Cisco Network Registrar 7.0*
Step 4	Configure policies.*	*User Guide for Cisco Network Registrar 7.0*
Step 5	Configure links.*	*User Guide for Cisco Network Registrar 7.0*

Table 3-5 Network Registrar Workflow for DHCPv6 (continued)

	Task	Refer to...
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.	*User Guide for Cisco Network Registrar 7.0*
Step 7	Back up the Network Registrar database.	*User Guide for Cisco Network Registrar 7.0*
Step 8	Verify that you are connected to the correct RDU.	Viewing Servers, page 12-21
Step 9	Reload the DHCP server.	*User Guide for Cisco Network Registrar 7.0*

Technology Workflows

This section describes the tasks that you must perform when configuring BAC to support specific technologies and include:

- DOCSIS Workflow, page 3-5
- PacketCable workflows:
  - PacketCable Secure, page 3-6
  - PacketCable Basic, page 3-8
- CableHome Workflow, page 3-10

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

DOCSIS Workflow

BAC supports these versions of the DOCSIS specifications: 1.0, 1.1, 2.0, and 3.0.

To successfully configure BAC for DOCSIS operations, you must perform the tasks described in Component Workflows, page 3-1, in addition to those described in this section.

Table 3-6 identifies the workflow to follow when configuring BAC to support DOCSIS.

Table 3-6 DOCSIS Workflow

	Task	Refer to...
Step 1	Configure the RDU	
	a. Configure all provisioned DHCP Criteria.	Configuring DHCP Criteria, page 13-15
	b. Configure provisioned Class of Service.	Configuring Class of Service, page 13-1
	c. Configure the promiscuous mode of operation.	System Defaults, page 13-12
Step 2	Configure the DPE	
	a. Enable the TFTP service.	The **service tftp** *1..1* **ipv4 | ipv6 enabled true** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
	b. Optionally, enable the ToD service.	The **service tod** *1..1* **ipv4 | ipv6 enabled true** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 3	Configure Network Registrar	
	Configure client classes/selection tags to match those added for the provisioned DOCSIS modem DHCP Criteria.	*User Guide for Cisco Network Registrar 7.0*

PacketCable Workflows

BAC supports these versions of the PacketCable specifications: 1.0, 1.1, and 1.5.

BAC 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 Secure, page 3-6
- PacketCable Basic, page 3-8

Note The workflows in this section assume that you have loaded an appropriate PacketCable configuration file and the correct MIBs.

PacketCable Secure

BAC 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 7-31.

You perform the PacketCable-related tasks described in this section only after completing the tasks described in Component Workflows, page 3-1.

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 3-7 identifies the workflow to follow when configuring BAC to support PacketCable Secure.

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

Table 3-7 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 13-30
	b. Configure all provisioned DHCP Criteria.	Configuring DHCP Criteria, page 13-15
	c. Configure all provisioned Class of Service.	Configuring Class of Service, page 13-1
	d. Configure an SNMPv3 cloning key.*	Configuring SNMPv3 Cloning on the RDU and DPE for Secure Communication with PacketCable MTAs, page 7-29
	e. If you are using Euro PacketCable, configure the RDU to use Euro-PacketCable MIBs.	Configuring Euro-PacketCable MIBs, page 7-32
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 Broadband Access Center DPE CLI Reference 4.0*
	b. Configure a privacy policy.*	The **service packetcable** *1..1* **registration policy-privacy** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
	c. Configure an SNMPv3 cloning key.*	The **service packetcable** *1..1* **snmp key-material** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
	d. Enable PacketCable.*	The **service packetcable** *1..1* **enable** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
	e. Enable the TFTP service.	The **service tftp ipv4 enabled true** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*

Table 3-7 PacketCable Secure Workflow (continued)

	Task	Refer to...
	f. Optionally, enable the ToD service.	The **service tod ipv4 enabled true** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
	g. Optionally, configure MTA file encryption.	The **service packetcable** *1..1* **registration encryption enable** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 3	Configure the KDC	
	a. Obtain a KDC license from your Cisco representative.	KDC Licenses, page 7-9
	b. Configure a certificate chain using the PKCert.sh tool. For Euro PacketCable, use the **-e** option.	Using the PKCert.sh Tool, page 14-2
	c. Configure a service key pair for each DPE's provisioning FQDN.	Using the KeyGen Tool, page 14-8
	d. Configure service keys for the ticket-granting-ticket (TGT).	Using the KeyGen Tool, page 14-8
	e. Configure Network Time Protocol (NTP).	Solaris documentation for information on configuring NTP for Solaris
Step 4	Configure DHCP	
	a. Configure all necessary PacketCable properties.	Using the KeyGen Tool, page 14-8
	b. Configure dynamic DNS for the MTA scopes.	*User Guide for Cisco Network Registrar 7.0*
	c. Configure client classes/scope-selection tags to match those added for provisioned PacketCable MTA DHCP criteria.*	*User Guide for Cisco Network Registrar 7.0*
Step 5	Configure DNS	
	a. Configure dynamic DNS for each DHCP server.	*User Guide for Cisco Network Registrar 7.0*
	b. Configure a zone for the KDC realm.	*User Guide for Cisco Network Registrar 7.0*

PacketCable Basic

You perform the PacketCable-related tasks described in this section only after completing those described in Component Workflows, page 3-1.

Table 3-8 identifies the workflow to follow when configuring PacketCable Basic on BAC.

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

Table 3-8 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 Broadband Access Center DPE CLI Reference 4.0*
	b. Enable the TFTP service.	The **service tftp** *1..1* **ipv4 enabled true** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
	c. Optionally, enable the ToD service.	The **service tod** *1..1* **ipv4 enabled true** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 2	Configure DHCP	
	a. Configure dynamic DNS for the MTA scopes.	*User Guide for Cisco Network Registrar 7.0*
	b. Configure client classes/scope-selection tags that match those added for provisioned PacketCable MTA DHCP criteria.*	*User Guide for Cisco Network Registrar 7.0*
Step 3	Configure DNS	
	Configure dynamic DNS for each DHCP server.	*User Guide for Cisco Network Registrar 7.0*

Table 3-8 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 13-1
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 BAC. 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 5-34.	Configuring Class of Service, page 13-1

CableHome Workflow

To successfully configure BAC for provisioning using the non-secure CableHome technology, you must perform the tasks described in Component Workflows, page 3-1, in addition to those described in this section.

Table 3-9 describes the tasks you must perform on BAC to support CableHome.

Table 3-9 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 13-15
	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 13-1
	c. Configure the promiscuous mode of operation.	System Defaults, page 13-12
Step 2	Configure the DPE	
	a. Enable the TFTP service.	The **service tftp** *1..1* **ipv4 enabled true** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
	b. Optionally, enable the ToD service.	The **service tod** *1..1* **ipv4 enabled true** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*
Step 3	Configure Network Registrar	
	Configure the client classes/scope-selection tags to match those added for the provisioned non-secure CableHome DHCP Criteria.	*User Guide for Cisco Network Registrar 7.0*

CHAPTER 4

CPE Provisioning Overview

This chapter describes the management of customer premises equipment (CPE) using the technologies that the CPE supports for the Cisco Broadband Access Center (BAC). It features:

- Overview, page 4-1
- Device Object Model, page 4-2
- Discovered Data, page 4-4
- Configuration Generation and Processing, page 4-6
- Device Deployment in BAC, page 4-8
- Promiscuous Access for Devices, page 4-13

Overview

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

BAC 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

This release of BAC supports provisioning and managing of:

- IPv6 devices, which include:
  - Cable modems compliant with DOCSIS 3.0
  - Computers
  - Set-top boxes (STBs)
- Video STBs, specifically the RNG-200 STB, which is based on the evolving 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 BAC 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 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 BAC 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 4-1 illustrates the interaction among the various objects in the device data model.

Figure 4-1 Device Object Model

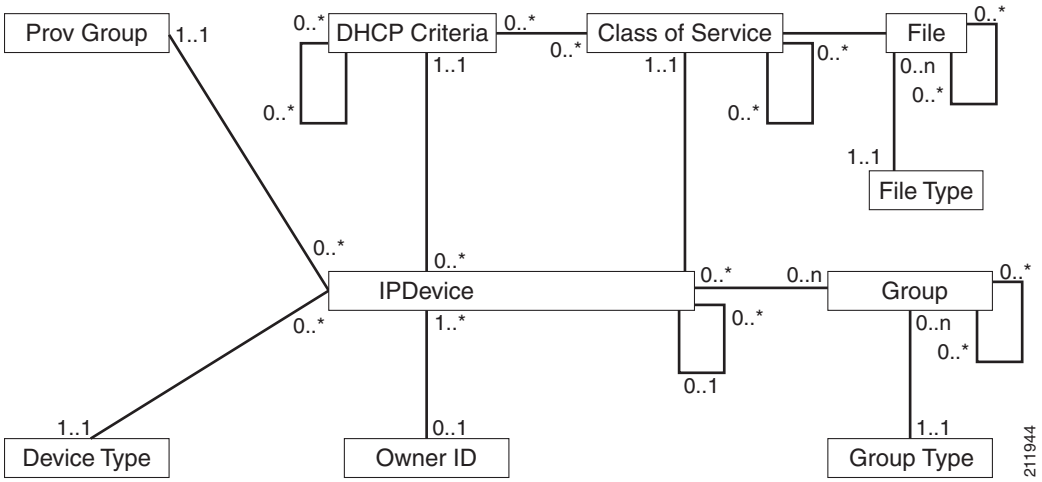


Table 4-1 describes the attributes and relationships unique to each object in the data model.

Table 4-1 Device Object Relationships

Object	Related to...
IPDevice • Could be preprovisioned or self-provisioned (See Device Deployment in BAC, page 4-8). • 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 4-3.)	• 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, BAC 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 BAC discovers using this set of properties is known as discovered data. BAC 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

BAC 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 BAC provisioning.

Table 4-2 lists the data that BAC discovers for IPv4 devices.

Table 4-2 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 4-2 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 4-3 lists the data that BAC discovers for IPv6 devices.

Table 4-3 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 12-9.

For a list of properties that BAC extensions use to discover data for DHCPv4 and DHCPv6, see Attributes versus Options, page 6-15.

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 BAC deployment, it initiates contact with the BAC 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 Template Files, page 4-6
- Property Hierarchy, page 4-7
- Templates and Property Hierarchy, page 4-7
- Custom Properties, page 4-8

Static Files versus Template Files

You can provision devices with BAC using two types of configuration files: static files and template files.

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

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. BAC 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 the Configuration File Utility, page 5-21. Template files are identified by a *.tmpl* extension.

For a summary of static provisioning versus dynamic provisioning, see Table 2-4.

Property Hierarchy

BAC properties provide a means to access and store data in BAC via the API. Preprovisioned, discovered, and status data can be retrieved via properties of corresponding objects via the API. Properties also enable configuration of BAC 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 BAC property hierarchy. For details on whether you can assign the property at any level, refer to the API Javadoc.

The BAC 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. Device-detected properties—Specifies properties that are stored on the device record by the device detection process.
4. Provisioning Group—Specifies properties of a device's provisioning group.
5. 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.
6. 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.
7. 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.
8. 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 BAC 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, BAC uses a single template to generate configurations for anywhere from a few devices to hundreds, thousands, or even millions of devices.

Custom Properties

BAC 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 13-5.

Device Deployment in BAC

A BAC 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 2-14).

BAC 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.

This section describes:

- CPE Registration Modes, page 4-8
- CPE Provisioning Flows, page 4-9

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 4-9
- Configuration Update Workflow, page 4-12

Initial Configuration Workflows

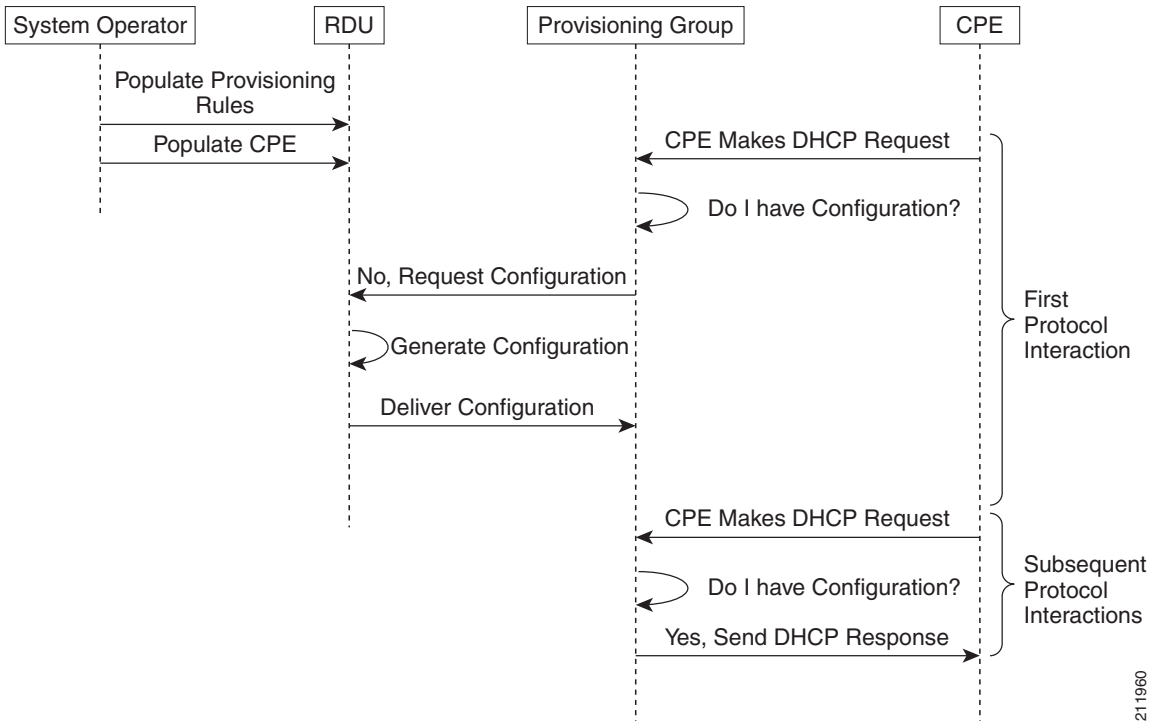
This section describes the configuration workflow when a device is initially installed and booted. The workflows differ based on deployment and registration mode and include:

- Preprovisioned Device Workflow, page 4-10
- Self-Provisioned Device Workflow, page 4-11

Preprovisioned Device Workflow

This section describes the workflow for a preprovisioned device. Figure 4-2 shows a common initial configuration workflow.

Figure 4-2 Workflow of Initial Device Configuration – Preprovisioned Mode



1. From the BAC 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 BAC 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 BAC 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.

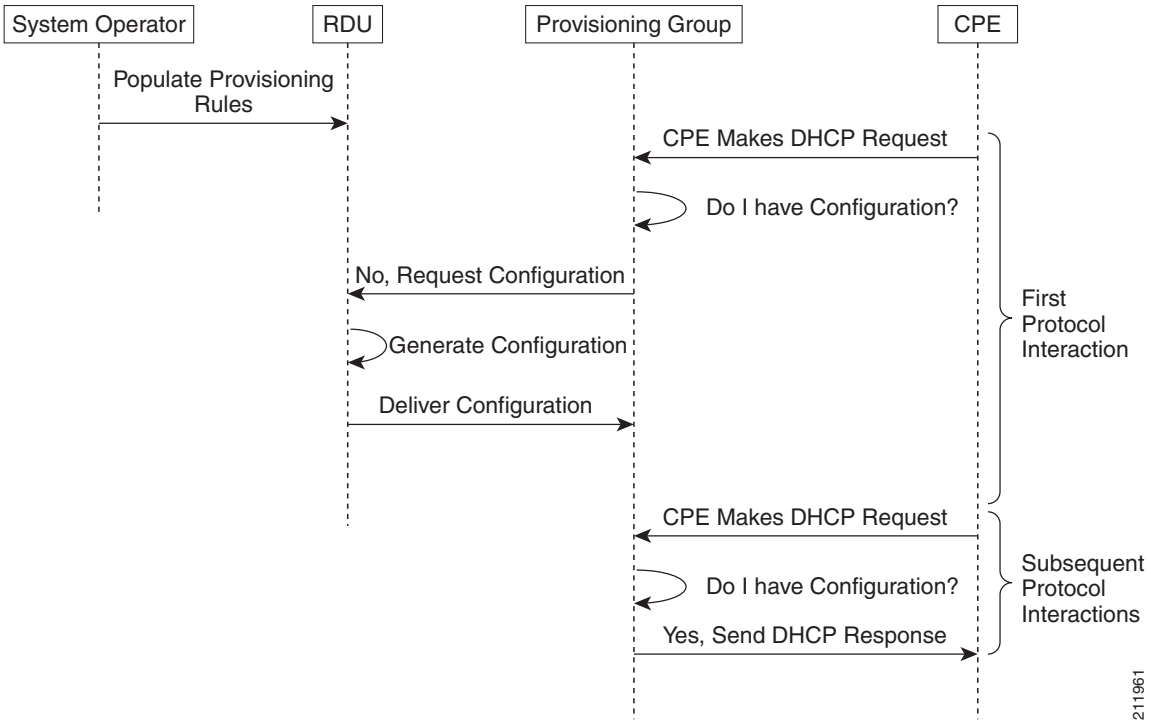
5. The device configuration is forwarded to the DPE and cached there. Now, the DPE is programmed to handle subsequent CPE protocol interactions for the device autonomously from the RDU. Once the device is added to the network and a configuration is generated for the device, the device boots to allow 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 4-3 shows a common initial configuration workflow.

Figure 4-3 Workflow of Initial Device Configuration – Self-Provisioned Mode



1. From the BAC 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 BAC 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 BAC 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.

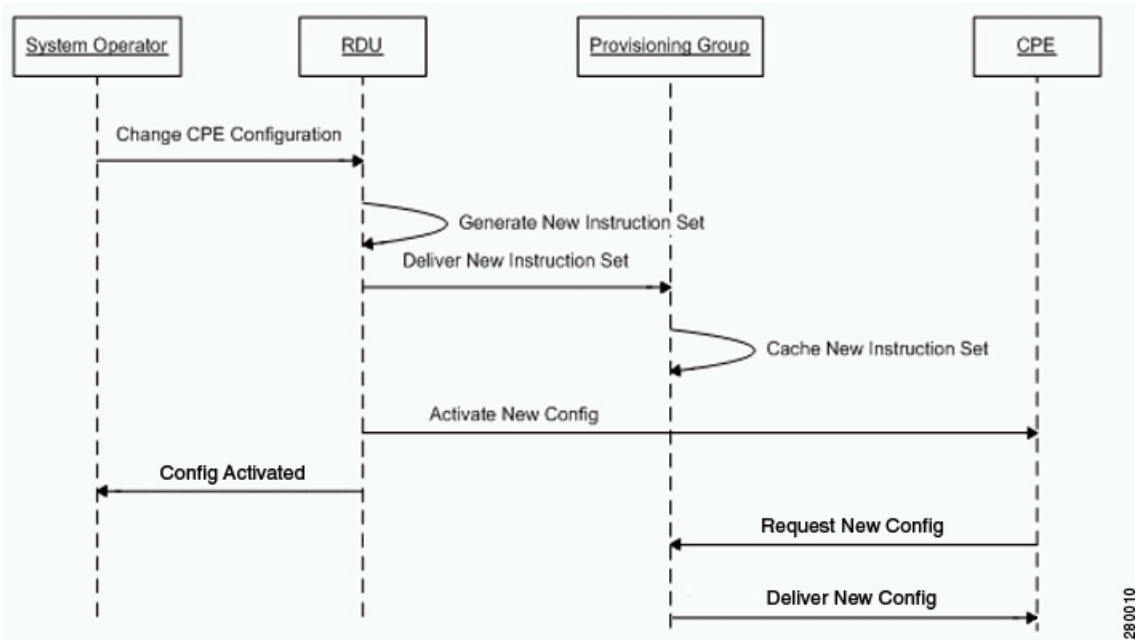
4. The RDU generates a configuration appropriate for the device. The resulting configuration directs DPE responses to various CPE protocol events, such as a DHCP Discover.
5. The device configuration is forwarded to the DPE and cached there. Now, the DPE is programmed to handle subsequent CPE protocol 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 4-4 shows the common configuration workflow when you change the configuration of a device that was previously configured.

Figure 4-4 Workflow of Device Configuration Update



1. From the BAC 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. In the case of cable, the RDU does an SNMP Set on the modem or MTA, causing the device to reboot.

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 BAC. 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 4-4 describes the ways in which you can configure a promiscuous policy for a device.

Table 4-4 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.	*getProvGroupProperties* *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*

Note You cannot configure the promiscuous policy directly on devices, such as specific modems.

Promiscuous Access and Property Hierarchy

You can configure a promiscuous policy on a number of objects in BAC. 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 BAC property hierarchy. The first object in the property hierarchy that has a specific property determines the value that BAC is to use.

BAC looks up the properties of the promiscuous policy in the property hierarchy of the device's relay agent. For example, for a computer, BAC 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 4-7. For more details about promiscuous policy see Properties for Configuring Promiscuous Policy, page 4-15.

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 BAC 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 BAC 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 BAC 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 4-4 describes.
- Disabled—Disables promiscuous access. If the property does not exist, the default is the disabled setting.

See Table 4-5 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 4-4 describes the applicable API calls for all the properties that are described in this section.

Table 4-5 describes the properties that you can use to enable promiscuous access.

Table 4-5 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`
/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`

Table 4-5 Properties for Enabling Promiscuous Access (continued)

Property Name	Description
/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`
/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 4-6 describes the read-write properties that you must configure to select Class of Service for devices granted promiscuous access.

Table 4-6 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`
/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 4-7 describes the read-write properties that you must configure to select DHCP Criteria for devices granted promiscuous access.

Table 4-7 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`

Table 4-7 Promiscuous Access–Read-Write Properties for DHCP Criteria (continued)

DHCP Criteria Property Name	Description
/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`
/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 4-8 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 4-8 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 *getClassOfServiceProperties* *getDHCPCriteriaDetails*	**API Constant** `PolicyKeys.IS_SYSTEM_WIDE_DEFAULT_PROMISCUOUS`

Table 4-8 Promiscuous Access–Read-Only Properties (continued)

Property Name	Description	
/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 *getClassOfServiceProperties* *getDHCPCriteriaDetails*	**API Constant** `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 *getClassOfServiceProperties* *getDHCPCriteriaDetails*	**API Constant** `PolicyKeys.RELATED_CLASS_OF_SERVICE`
/related/dhcpCriteria	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 **Note** 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.	
	Applicable API Calls *getClassOfServiceProperties* *getDHCPCriteriaDetails*	**API Constant** `PolicyKeys.RELATED_DHCP_CRITERIA`

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 5

Configuration Templates Management

This chapter details the templates that Broadband Access Center (BAC) supports for device configuration and device management. This chapter features:

- Template Files–An Overview, page 5-1
- Template Grammar, page 5-2
  - SNMP VarBind, page 5-6
  - Macro Variables, page 5-8
  - SNMP TLVs, page 5-10
  - Encoding Types for Defined Options, page 5-14
- Using the Configuration File Utility, page 5-21

Template Files–An Overview

BAC 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. BAC 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 BAC 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:

- BAC provisioning flows
- DOCSIS 1.0, 1.1, 2.0, and 3.0 RFI specifications
- DOCSIS Layer 2 Virtual Private Networks specification
- PacketCable 1.0, 1.1, and 1.5 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 5-2
- Includes, page 5-3
- Options, page 5-3
- Instance Modifier, page 5-4
- OUI Modifier, page 5-5

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 5-1 describes the available template grammar options.

Table 5-1 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 5-1 shows sample comment usage.

Example 5-1 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 5-2 and Example 5-3 illustrate both correct and incorrect usage of the include option.

Example 5-2 Correct Include Statement Usage

```
# Valid, including common options
include "common_options.tmpl"
```

Example 5-3 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 5-4.

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 5-14, and DOCSIS Option Support, page B-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 5-6, 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 5-4 and Example 5-5 illustrate both correct and incorrect usage of the option statement.

Example 5-4 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 5-5 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 5-6 and Example 5-7 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 5-6 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 5-7 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 5-4, 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 BAC 4.0, you can use a single template to specify various TLV 43s from many vendors. Example 5-8 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 5-8 illustrates BAC 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 5-8 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 5-9 and Example 5-10 illustrate incorrect usage of the OUI modifier.

Example 5-9 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 5-10 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
- DOCS-CABLE-DEVICE-MIB-OBSOLETE
- CISCO-CABLE-MODEM-MIB

Two versions of the DOCS-CABLE-DEVICE MIB are loaded into the RDU:

- DOCS-CABLE-DEVICE-MIB-OBSOLETE (experimental branch)
- DOCS-CABLE-DEVICE-MIB (mib2 branch)

A fully qualified MIB OID (.experimental...) always uniquely identifies a MIB OID.

If you use a nonfully qualified MIB OID from DOCS-CABLE-DEVICE-MIB, it will always default to DOCS-CABLE-DEVICE-MIB and not DOCS-CABLE-DEVICE-MIB-OBSOLETE.

Example 5-11 and Example 5-12 illustrate using a fully qualified MIB OID and a nonfully qualified MIB OID.

Example 5-11 Fully Qualified MIB OID

```
# Valid, uniquely identifying an OID
option 11 .experimental.docsDev.docsDevMIBObjects.docsDevNmAccessTable.docsDevNmAccess
Entry.docsDevNmAccessStatus.1, Integer, 4
```

Example 5-12 Nonfully Qualified MIB OID (Defaults to DOCS-CABLE-DEVICE-MIB)

```
# Valid, NonFully Qualified MIB OID.
option 11 .docsDevNmAccessStatus.1, Integer, 4
```

If no DOCSIS CMs in a deployment require DOCS-CABLE-DEVICE-MIB-OBSOLETE, you can always use the shorter form of the MIB OID.

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 BAC 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 5-13 and Example 5-14 illustrate correct and incorrect usage of Option 11.

Example 5-13 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 5-14 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

BAC supports SNMP TLVs in dynamic template files, using Option 11 and 64, for:

- DOCSIS—From Broadband Access Center for Cable (BACC) version 2.0 onwards.
- PacketCable—From BACC version 2.5 onwards.
- CableHome—From BACC version 2.6 onwards.

To validate the syntax of the SNMP TLVs in these template files, BAC 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 5-2 describes the allowed SNMP variable type names.

Table 5-2 SNMP Variable Types

IETF standard SMI Data Type	SNMP API name
Integer32	INTEGER
Integer (Enumerated)	INTEGER
Unsigned32	UNSIGNED32
Gauge32	GAUGE
Counter32	COUNTER
Counter64	COUNTER64

Table 5-2 SNMP Variable Types (continued)

IETF standard SMI Data Type	SNMP API name
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.

BACC 2.6 or earlier

If you have the MIB corresponding to the SNMP OID that you want to use, you can add the MIB file to the BAC 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 RDU:

Step 1 Copy the new MIB file to the *BPR_HOME/rdu/mibs* directory.

Step 2 Add the */docsis/mibs/custom/mibList* property, whose value contains a comma-separated list of MIB filenames, to the:

a. *rdu.properties* file, which is used by the RDU and the administrator user interface. This file resides in the *BPR_HOME/rdu/conf* directory.

b. *api.properties* file, which the Configuration File Utility (runCfgUtil.sh tool) uses.

Note The *api.properties* file is not created during the BAC installation process. You must manually create this file for initial use, in any text editor. Ensure that you locate this file in the *BPR_HOME/rdu/conf* directory.

The *api.properties* file contains a */docsis/mibs/custom/mibList*, which is configured for a set of MIBs that you can use in Arris embedded MTAs (eMTAs).

Step 3 Restart the RDU via the BAC process watchdog, using the **/etc/init.d/bprAgent restart rdu** command.

The following example describes the addition of ARRIS MIBs for use in templates to configure ARRIS MTAs.

Assume that you want to use an Arris vendor-specific SNMP TLV:

```
option 11 .ppCfgMtaCountryTemplate.0, INTEGER, 9
```

and the following MIB files were made available:

- ARRIS-MIB
- ARRIS-CM-CAPABILITY-MIB
- ARRIS-CM-DEVICE-MIB
- ARRIS-MTA-DEVICE-MIB
- PACKETPORT-MIB

You must copy the MIB files to the *BPR_HOME/rdu/mibs* directory, and insert the following property in the *api.properties* and *rdu.properties* files:

/docsis/mibs/custom/mibList=ARRIS-MIB,ARRIS-CM-CAPABILITY-MIB,ARRIS-CM-DEVICE-MIB,ARRIS-MTA-DEVICE-MIB,PACKETPORT-MIB

BACC 2.7 or later

Note The */docsis/mibs/custom/mibList* property is renamed */snmp/mibs/mibList* from 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 BAC 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 BAC RDU:

Step 1 Launch the BAC 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 BAC debugging information to resolve MIB load-order issues.

Note The MIB load order in BAC is set by the order in which the MIBs are listed in the:

- */docsis/mibs/custom/mibList* property, if you are using BACC 2.6.x or earlier releases.
- */snmp/mibs/MibList* property, if you are using 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 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 BAC 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 BACC version you are using:

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.

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 BAC 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 5-3 identifies the options with defined encoding types.

Table 5-3 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`

Table 5-3 Defined Option Encoding Types (continued)

Encoding	Input	Examples

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`

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`

Table 5-3 Defined Option Encoding Types (continued)

Encoding	Input	Examples
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. BAC automatically calculates the hash for the cable modem and CMTS MIC option.
Bytes	A series of hexadecimal octets. Each octet must be 2 characters.	`00010203040506070 8`

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`

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`

Table 5-3 Defined Option Encoding Types (continued)

Encoding	Input	Examples
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`
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`
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`

Table 5-3 Defined Option Encoding Types (continued)

Encoding	Input	Examples
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 5-3 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 5-3 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 the Configuration File Utility

You use the configuration file utility to test, validate, and view PacketCable 1.0/1.1/1.5, 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 5-1, 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 BAC 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 5-22
- Adding a Template to BAC, page 5-23
- Converting a Binary File to a Template File, page 5-23
- Testing Template Processing for a Local Template File, page 5-25
- Testing Template Processing for an External Template File, page 5-26
- Specifying Macro Variables at the Command Line, page 5-29

- Specifying a Device for Macro Variables, page 5-30
- Specifying Output to a Binary File, page 5-31
- Viewing a Local Binary File, page 5-32
- Viewing an External Binary File, page 5-33
- Activating PacketCable Basic Flow, page 5-34
- Generating TLV 43s for Multivendor Support, page 5-36

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** *secret*—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 2.0. The values that BAC 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**—Specifies the PacketCable locale: **na** (North America) or **euro** (Europe). 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**.

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 BAC 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 BAC

To use the configuration file utility to test BAC templates:

Step 1 Develop the template as described in Template Files–An Overview, page 5-1. 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 5-26.

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. BAC 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: 4.0, 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: 4.0, 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 username.
- **changeme**—Identifies the 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: 4.0, 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** *secret*

- **-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.
- *secret*—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: 4.0, 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.
```

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: 4.0, 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: 4.0, 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: 4.0
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: 4.0, 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      Option    Description                      Value

0       FE0101          254       Telephony Config File Start/End  1

        0B153013060E    11        SNMP MIB Object                  .iso.org.dod.internet.
3       2B06010401A30B                                             private.enterprises.ca
        02020101010700                                             bleLabs.clabProject.cl
        020102                                                     abProjPacketCable.pktc
                                                                   MtaMib.pktcMtaMibObjec
                                                                   ts.pktcMtaDevBase.pktc
                                                                   MtaDevEnabled.0,INTEGE
                                                                   R,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: 4.0, 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: 4.0, 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* **-p** *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: 4.0, 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 03	43.5.1	VPNID Subtype	0234560003
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 5F6D6F6E6974 6F725F696D61 67652E62696E	43.3	Update Boot Monitor Image	boot_monitor_image.bin
66	061071E79068 3DE8B9950536 8936F4C5312F	6	CM MIC Configuration Setting	71E790683DE8B995 05368936F4C5312F
84	0710DB0EED14 B5B3428D2B15 0DA582B41A54	7	CMTS MIC Configuration Setting	DB0EED14B5B3428D 2B150DA582B41A54
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.
```

CHAPTER 6

DOCSIS Configuration

This chapter describes the provisioning flow in a Broadband Access Center (BAC) DOCSIS deployment. It also provides information required before configuration and describes the available tools.

- DOCSIS Workflow, page 6-1
- Using MIBs with Dynamic DOCSIS Templates, page 6-9
- BAC Features for DOCSIS Configurations, page 6-10
- IPv6 Support, page 6-13
- Lease Query, page 6-19

Note See DOCSIS Option Support, page B-1, for information on DOCSIS options supported by this BAC release.

This chapter assumes that you are familiar with the contents of these specifications:

- DOCSIS 3.0:
  - CM-SP-SECv3.0-I04-070518
  - CM-SP-PHY3.0-I04-070518
  - CM-SP-MULPIv3.0-I04-070518
  - CM-SP-OSSIv3.0-I03-070518
- DOCSIS 2.0:
  - CM-SP-RFI2.0-I12-071206
  - L2VPN CM-SP-L2VPN-I06-071206

DOCSIS Workflow

This section describes the provisioning workflow contained in the DOCSIS Provisioning Specification for DHCPv4 and DHCPv6.

- DOCSIS DHCPv4 Workflow, page 6-2
- DOCSIS DHCPv6 Workflow, page 6-5

DOCSIS DHCPv4 Workflow

Figure 6-1 shows the provisioning workflow contained in the DOCSIS Provisioning Specification for DHCPv4. Each step is described subsequently.

Figure 6-1 DOCSIS DHCPv4 Provisioning Flow

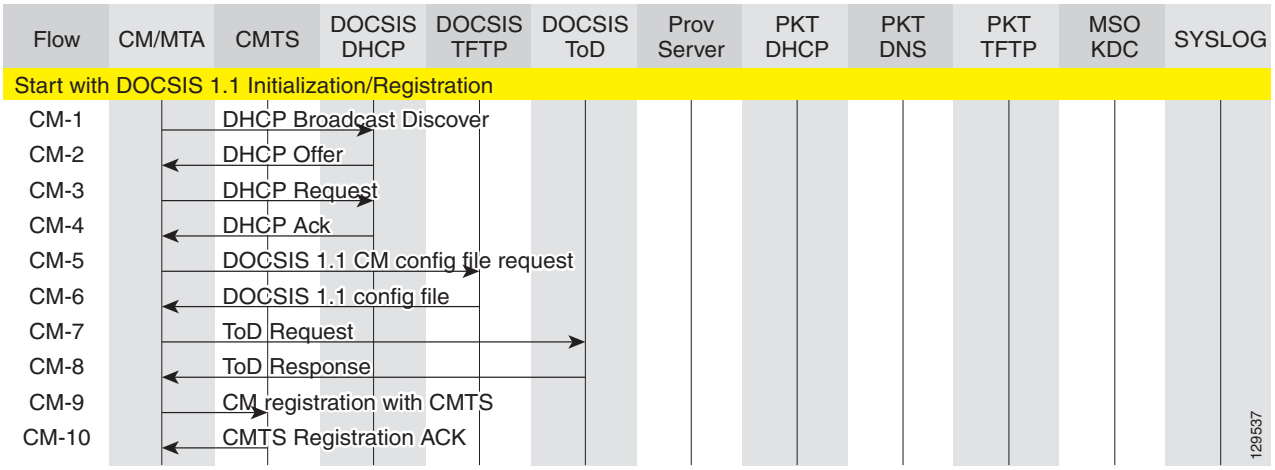


Table 6-1 describes the potential problems that can exist at various DOCSIS provisioning steps illustrated in Figure 6-1.

Table 6-1 DOCSIS DHCPv4 Workflow Description

Step	DOCSIS DHCPv4 Workflow	Potential Problems
CM[1]-1	DHCP Discover	• init(d) state • No addresses available • Incorrect BAC shared secret • Incorrectly configured Class of Service • DOCSIS template parsing errors (invalid option, include file - not found, and so on) **Cisco Network Registrar DHCP** • Incorrect DHCP configuration • DHCP server not there in provisioning group **BAC 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 (default 49187) • 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 6-1 DOCSIS DHCPv4 Workflow Description (continued)

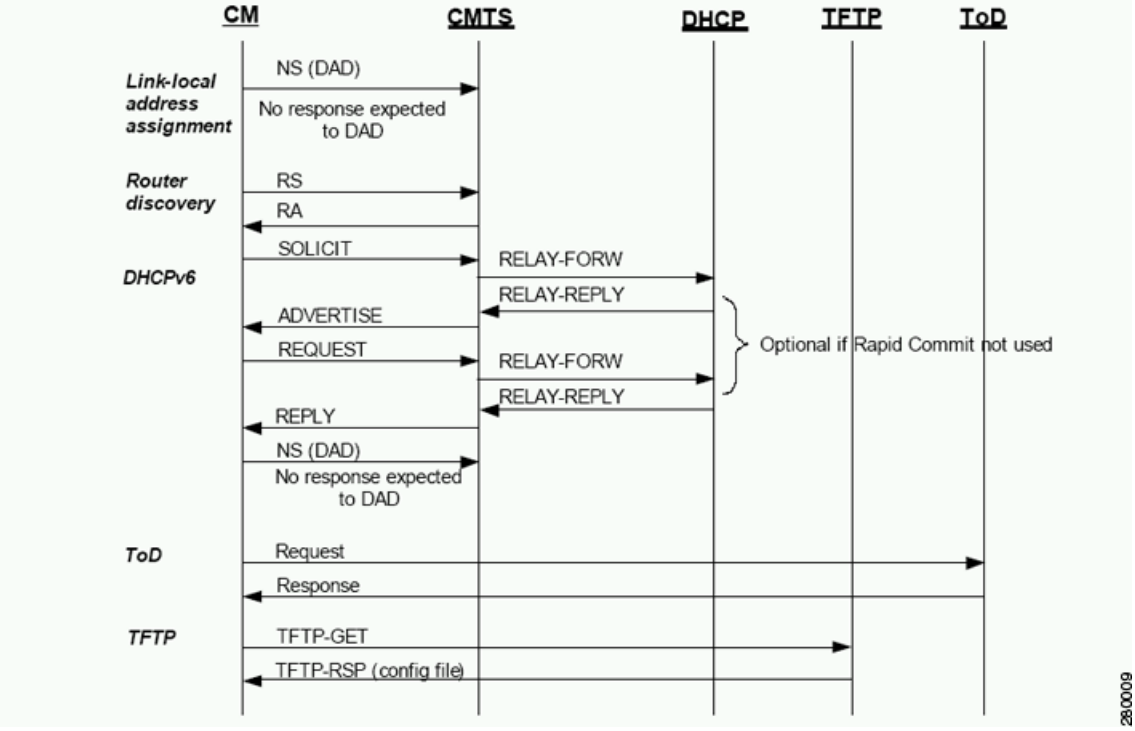
Step	DOCSIS DHCPv4 Workflow	Potential Problems
CM-2	DHCP Offer	Routing issues between DHCP and the cable modem termination system (CMTS)
CM-3	DHCP Request	• init(i) state • DHCP server did not provide all required parameters
CM-4	DHCP Ack	
CM-5	TFTP Request	• Init(o) state • Routing issues between CMTS 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 BAC 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 6-2 shows the provisioning workflow contained in the DOCSIS Provisioning Specification for DHCPv6. Each step is described subsequently.

Figure 6-2 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 6-2 describes the potential problems that can exist at various DOCSIS provisioning steps illustrated in Figure 6-2.

Table 6-2 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 6-2 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 BAC 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 BAC RDU configuration) **BAC 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 (default 49187) • 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 6-2 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 6-2 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.		

Using MIBs with Dynamic DOCSIS Templates

For a full list of MIBs that BAC ships with, see SNMP VarBind, page 5-6.

Two versions of the DOCSIS MIB are loaded into the RDU:

- DOCS-CABLE-DEVICE-MIB-OBSOLETE (experimental branch)
- DOCS-CABLE-DEVICE-MIB (mib2 branch)

For information on how to use them, see DOCSIS MIBs, page 5-7.

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 7-32.

You can add SNMP TLVs to a template:

- When no MIB is available. See Adding SNMP TLVs Without a MIB, page 5-10.
- With vendor-specific MIBs. See Adding SNMP TLVs With Vendor-Specific MIBs, page 5-11.

BAC Features for DOCSIS Configurations

This section describes BAC value-added features as they relate to the DOCSIS technology.

Dynamic Configuration TLVs

The DPE adds the following TLVs 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 13-3 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 13-3 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 6-11. If DSS is set on the Cisco CMTS, you must ensure that the TFTP Server Provisioned Modem Address is disabled.

- 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 13-3 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 13-3 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 BAC:
  - **ip dhcp relay information option**
  - **no ip dhcp relay information check**
  - **cable helper-address** *x.x.x.x*

    where *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, refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

Support for DOCSIS 1.0, 1.1, 2.0, and 3.0

BAC 4.0 supports DOCSIS 1.0, 1.1, 2.0, and 3.0. For information describing the TLVs, see Template Grammar, page 5-2, and for a list of options that this BAC release supports in each DOCSIS version, see DOCSIS Option Support, page B-1.

Dynamic DOCSIS Version Selection

BAC 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, BAC, if so configured, determines the optimum DOCSIS configuration file for the device. This is the highest 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

BAC 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

This BAC release 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 BAC 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 13-18.

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.

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 the Configuration File Utility, page 5-21. 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

BAC 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 BAC 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, BAC 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, BAC uses the DOCSIS configuration filename property without the DOCSIS version suffix and supplies the filename to specify in the device configuration.

IPv6 Support

This BAC release introduces support for the newest revision of 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
  - RNG-200 STBs, which is based on the evolving OpenCable Application Platform
  - Variants of eSAFE (embedded Service/Application Functional Entities) devices, such as mixed-IP mode PacketCable Multimedia Terminal Adapters (MTAs)

  BAC provides services required to provision IPv6 devices, such as IPv6 support for TFTP and ToD services. BAC 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—BAC provisions DOCSIS cable modems in the IPv4, IPv6, or dual stack mode (comprising IPv4 and IPv6 provisioning modes). For details on the various modes, see Single Stack versus Dual Stack, page 6-15.

    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 6-5.

Before you enable BAC to provision devices in IPv6, ensure that you enable IPv6 on your system. To enable your machine to support IPv6, 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
```

Then, also run these commands:

```
# /usr/lib/inet/in.ndpd
# touch /etc/hostname6.intf
```

where *intf* identifies the interface on which you want to enable IPv6.

These sections describe concepts related to IPv6 in BAC:

- IPv6 Addressing, page 6-14
- Single Stack versus Dual Stack, page 6-15
- DHCP Options for IPv6, page 6-15
- Attributes versus Options, page 6-15

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.

BAC 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.

Note BAC 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 The Cisco BAC does not support CM dual stack mode but the CPE dual stack mode is fully supported. 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 BAC supports, refer to the *User Guide for Cisco Network Registrar 7.0*.

Attributes versus Options

This section describes attributes and options as used by BAC 4.0 when communicating with Network Registrar.

BAC uses DHCP extensions installed on Network Registrar to manipulate DHCP messages based on the configuration in its database. Using these extensions, BAC 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, Network Registrar exposes a set of dictionaries to BAC extensions. The BAC extensions use these dictionaries to interact with Network Registrar.

There are three types of dictionaries: the attribute dictionaries that the request and response dictionaries use, the environment dictionary, and the inform dictionary.

- 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 BAC extension and the RDU.

The dictionaries represent various DHCP options and settings as configured on BAC 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.

BAC 4.0, along with Network Registrar 7.0, supports two API versions, each of which BAC 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 BAC 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 BAC extensions discover for DHCPv6:

Note From the administrator user interface, you can view the settings for these properties on the **Configuration > Defaults > NR Defaults** page.

- Properties that control the behavior of Network Registrar extensions in versions earlier than 4.0. See Table 6-3.
- Properties that control the behavior of Network Registrar extensions for DHCPv4 in BAC 4.0. See Table 6-4.

- Properties that control the behavior of Network Registrar extensions for DHCPv6 in BAC 4.0 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 6-5.

Table 6-3 describes the properties that influence the behavior of Network Registrar extensions in BAC versions before 4.0.

Table 6-3 Properties for Network Registrar Extensions Before BAC 4.0

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/attributesToPullFrom RequestAsBytes	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/attributesToPullFrom RequestAsStrings	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`
/cnrExtension/attributesToReadFrom EnvironmentDictionary	Identifies a list of attributes that must be pulled from the Network Registrar environment dictionary
	API Constant `CNR_ATTRIBUTES_TO_READ_FROM_ENVIRONMENT_DICTIONARY`

Table 6-4 describes the properties that control the behavior of Network Registrar extensions for DHCPv4 in BAC 4.0.

Table 6-4 Properties for DHCPv4 Network Registrar Extensions in BAC 4.0

Property Name	Description
/cnrExtension/attributesRequiredIn V4Request	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/attributesToPullFrom V4RequestAsBytes	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`

Table 6-4 Properties for DHCPv4 Network Registrar Extensions in BAC 4.0 (continued)

Property Name	Description
/cnrExtension/attributesToPullFrom V4RequestAsStrings	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 6-5 describes the properties that control the behavior of Network Registrar extensions for DHCPv6 in BAC 4.0.

Table 6-5 Properties for DHCPv6 Network Registrar Extensions in BAC 4.0

Property Name	Description
Client Message	
/cnrExtension/attributesRequiredIn V6Request	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/attributesToPullFrom V6RequestAsBytes	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`
/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`

Table 6-5 Properties for DHCPv6 Network Registrar Extensions in BAC 4.0 (continued)

Property Name	Description
/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`

Configuration Workflows for IPv6

Configuring BAC to support IPv6 involves two distinct workflows:

- Configuring the DPEs in the provisioning group. See Table 3-3.
- Configuring the Network Registrar servers in your network. See Table 3-5.

Lease Query

The BAC RDU queries Network Registrar for the IP address of devices using the DHCP lease query protocol. BAC then uses this information for device disruption and for reporting details of both IPv4 and IPv6 devices.

This BAC release supports the following configurations:

- Lease Query Autoconfiguration, page 6-19
- Lease Query Source IP Address, page 6-20

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 this BAC release, 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 BAC 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 Viewing Provisioning Groups, page 12-27.

Lease Query Source IP Address

In earlier 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

BAC, by default, binds to the IP addresses and ports that are described in Table 6-6.

Table 6-6 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.

Configuring BAC as Relay Agent for Lease Query

You can configure BAC 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 BAC to act as a relay agent for an IPv4 lease query, BAC 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 BAC 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 BAC 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 10-4.)

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. BAC then queries Network Registrar for each individual link and gets all the IP addresses. With this list, BAC 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 7

PacketCable Voice Configuration

This chapter describes the tasks you must perform to bring a PacketCable voice deployment into service.

This chapter contains information on these variants of PacketCable:

- PacketCable Secure eMTA Provisioning, page 7-1
- PacketCable Basic eMTA Provisioning, page 7-30
- Euro PacketCable, page 7-31

For information that will help you solve issues in a PacketCable voice technology deployment, see Troubleshooting PacketCable eMTA Provisioning, page 16-11.

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 Secure eMTA Provisioning

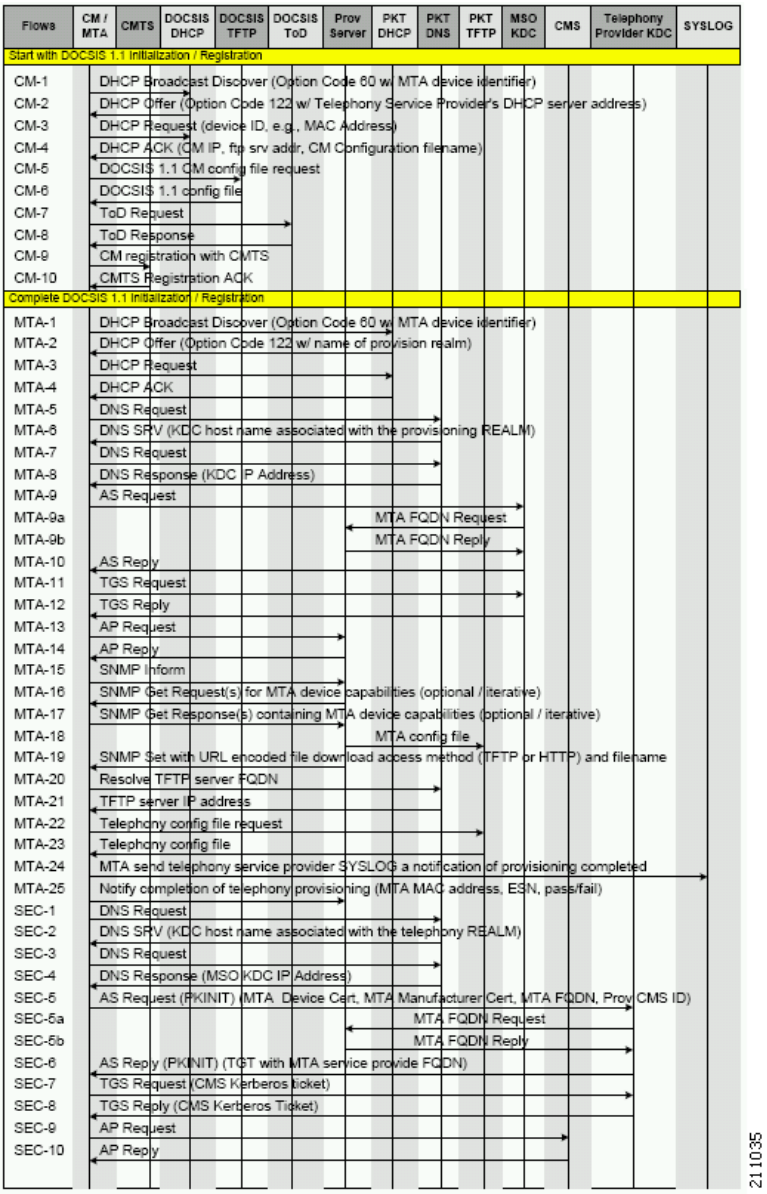
This section deals exclusively with Secure PacketCable voice provisioning. PacketCable Secure is designed to minimize the possibility of theft of telephony service, malicious disruption of service, and so on. PacketCable Secure depends on the Kerberos infrastructure to mutually authenticate the MTA and the provisioning system; in BAC, 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.

BAC 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



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 7-1 helps identify which step in the PacketCable provisioning flow is failing.

Table 7-1 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 7-1 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 Network Registrar DHCP configuration. Verify if the scopes are configured, if IP addresses are available, and if all secondary subnets are configured. • Check the BAC 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 BAC Properties". 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 7-1 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 the PKCert.sh Tool, page 14-2, 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 7-1 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 BAC 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 BAC provisioning. This flow involves getting Kerberos tickets associated with each CMS with which the MTA communicates. For details, refer to the PacketCable Security Specifications.	

KDC in Provisioning PacketCable Secure eMTAs

PacketCable Secure depends on the Kerberos infrastructure to mutually authenticate the MTA and the provisioning system; in BAC, the KDC functions as the Kerberos server. For an overview of the KDC component, see Key Distribution Center, page 2-12.

For important information related to the KDC, see:

- Default KDC Properties, page 7-7
- KDC Certificates, page 7-9
- KDC Licenses, page 7-9
- Multiple Realm Support, page 7-10

Default KDC Properties

The KDC has several default properties that are populated during a BAC installation into the *BPR_HOME/kdc/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, refer to the *Installation and Setup Guide for Cisco Broadband Access Center 4.0*.

- 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 7-2 describes the available logging levels for the KDC log file.

Table 7-2 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 BAC 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 7-1.

Example 7-1 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 BAC. 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 the PKCert.sh Tool, page 14-2.

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 in to the BAC 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 BAC 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/solaris/packetcable/certificates* directory.

BAC supports additional realms by installing subdirectories under the *BPR_HOME/kdc/solaris/packetcable/certificates* directory; each subdirectory is named after a specific realm.

Table 7-3 lists the different certificates, with their corresponding filenames, that must be available in the *BPR_HOME/kdc/solaris/packetcable/certificates* directory.

Table 7-3 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/solaris/packetcable/certificates* directory. Its private key (KDC_private_key.pkcs8) resides in the *BPR_HOME/kdc/solaris/* 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 7-4 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 7-4 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/solaris/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/solaris/packetcable/certificates/CISCO1.COM	For secondary realm CISCO1.COM: • *KDC.cer* • KDC private key
BPR_HOME/kdc/solaris/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, refer to the *Installation and Setup Guide for the Cisco Broadband Access Center 4.0*.

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, refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

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, refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

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, refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

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 BAC 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, refer to the *User Guide for Cisco Network Registrar 7.0.*

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/solaris/packetcable/certificates*.

For example:

```
/opt/CSCObac/kdc/solaris/packetcable/certificates# mkdir CISCO1.COM
/opt/CSCObac/kdc/solaris/packetcable/certificates# mkdir CISCO2.COM
```

For more information on creating directories, refer to 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, refer to Solaris documentation.

c. Copy the certificates mentioned in the previous step into the directory that you created. For information on copying files, refer to Solaris documentation on the **cp** command.

d. Copy the following certificates to the *BPR_HOME/kdc/solaris/packetcable/certificates* directory:

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

For information on copying files, refer to 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/solaris/
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 the PKCert.sh Tool, page 14-2.

f. Copy the *KDC.cer* file to the KDC certificate directory (*BPR_HOME/kdc/solaris/packetcable/certificates*). For information on copying files, refer to 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, refer to 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, refer to 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/solaris/
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 the PKCert.sh Tool, page 14-2.

j. Copy *KDC.cer* to the secondary realm directory; for example, the */CISCO1.COM* directory under *BPR_HOME/kdc/solaris/packetcable/certificates.* For information on copying files, refer to 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/solaris/packetcable/certificates.* For information on copying files, refer to 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/solaris/packetcable/certificates.* For information on copying files, refer to 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/solaris/
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 the PKCert.sh Tool, page 14-2.

n. Copy *KDC.cer* to the secondary realm directory; for example, the */CISCO2.COM* directory under *BPR_HOME/kdc/solaris/packetcable/certificates*. For information on copying files, refer to 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/solaris/packetcable/certificates*. For information on copying files, refer to 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/solaris/packetcable/certificates*. For information on copying files, refer to Solaris 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 the KeyGen Tool, page 14-8.

Step 7 Ensure that the service keys you generated in Step 6 exist in the *BPR_HOME/kdc/solaris/keys* directory.

For example:

```
/opt/CSCObac/kdc/solaris/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, refer to 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/solaris/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 ./solaris/packetcable/certificates
# ls
KDC.cer
Local_System.cer
CableLabs_Service_Provider_Root.cer  MTA_Root.cer
CISCO1.COM                           Service_Provider.cer
CISCO2.COM

# cd ./solaris/packetcable/certificates/CISCO1.COM
# ls
KDC.cer
KDC_private_key_proprietary.

# cd ./solaris/packetcable/certificates/CISCO2.COM:
# ls
KDC.cer
KDC_private_key_proprietary.
```

For more information, refer to Solaris documentation.

Step 9 Restart the KDC.

For example:

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

For more information, see Using the BAC Process Watchdog from the Command Line, page 9-2.

Step 10 Configure the BAC 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 13-15.

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 13-15.

c. Add templates as files to BAC 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 7-2 on page 7-26.
5. Click **Submit**.

For more information, see Managing Files, page 13-17.

d. Add templates as files to BAC 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 7-3 on page 7-27.
5. Click **Submit**.

For more information, see Managing Files, page 13-17.

e. Add templates as files to BAC 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 7-4 on page 7-27.
5. Click **Submit**.

For more information, see Managing Files, page 13-17.

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 13-1.

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 13-1.

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 13-1.

Step 11 Bring the devices online and provision them. Refer to 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.

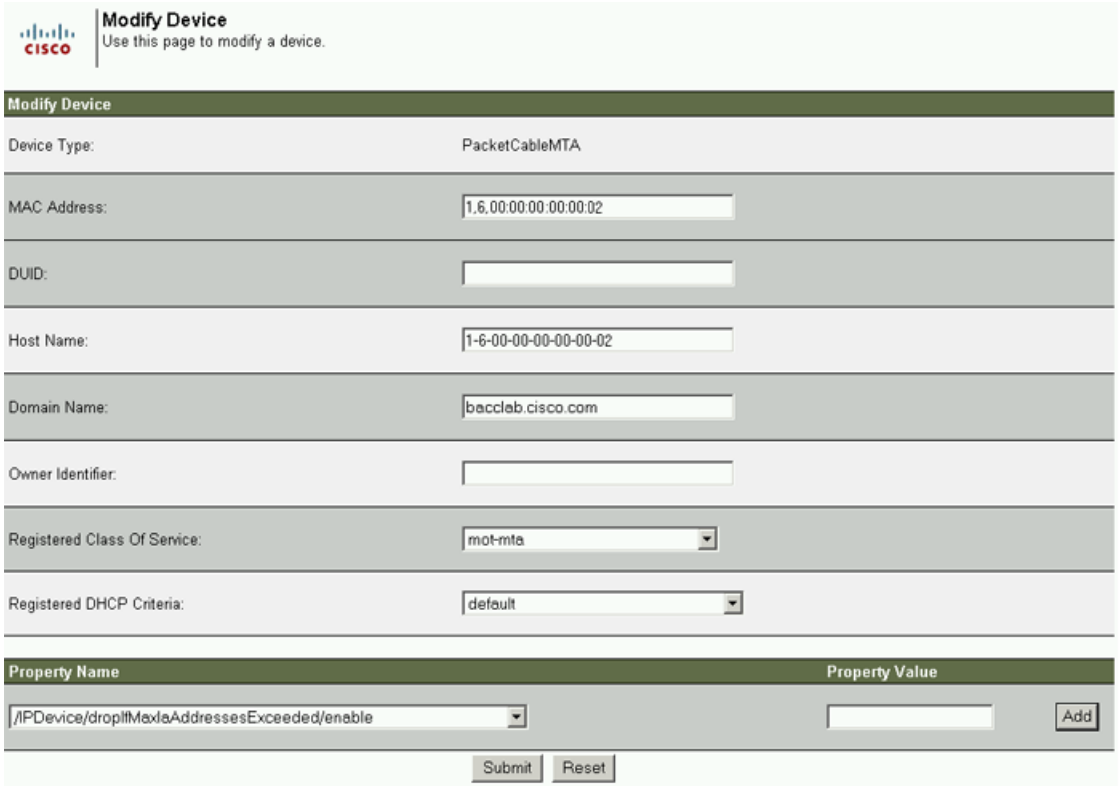
d. From the drop-down list corresponding to Registered Class of Service, select **mot-mta**. This is the Class of Service that you added in Step 10-f.

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

f. Click **Submit**.

Figure 7-2 lists device details for the Motorola MTA.

Figure 7-2 Provisioning Motorola MTA–Device Details



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.

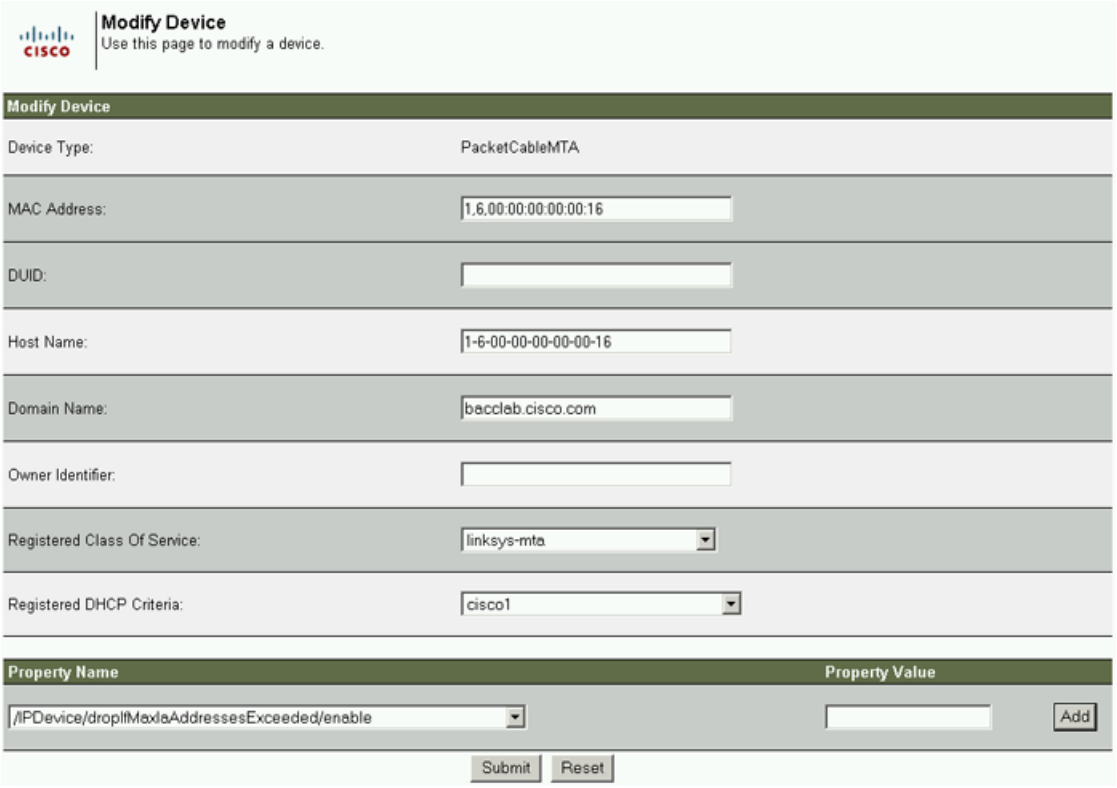
d. From the drop-down list corresponding to Registered Class of Service, select **linksys-mta**. This is the Class of Service that you added in Step 10-g.

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**.

Figure 7-3 lists device details for the Linksys MTA.

Figure 7-3 Provisioning Linksys MTA–Device Details



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.

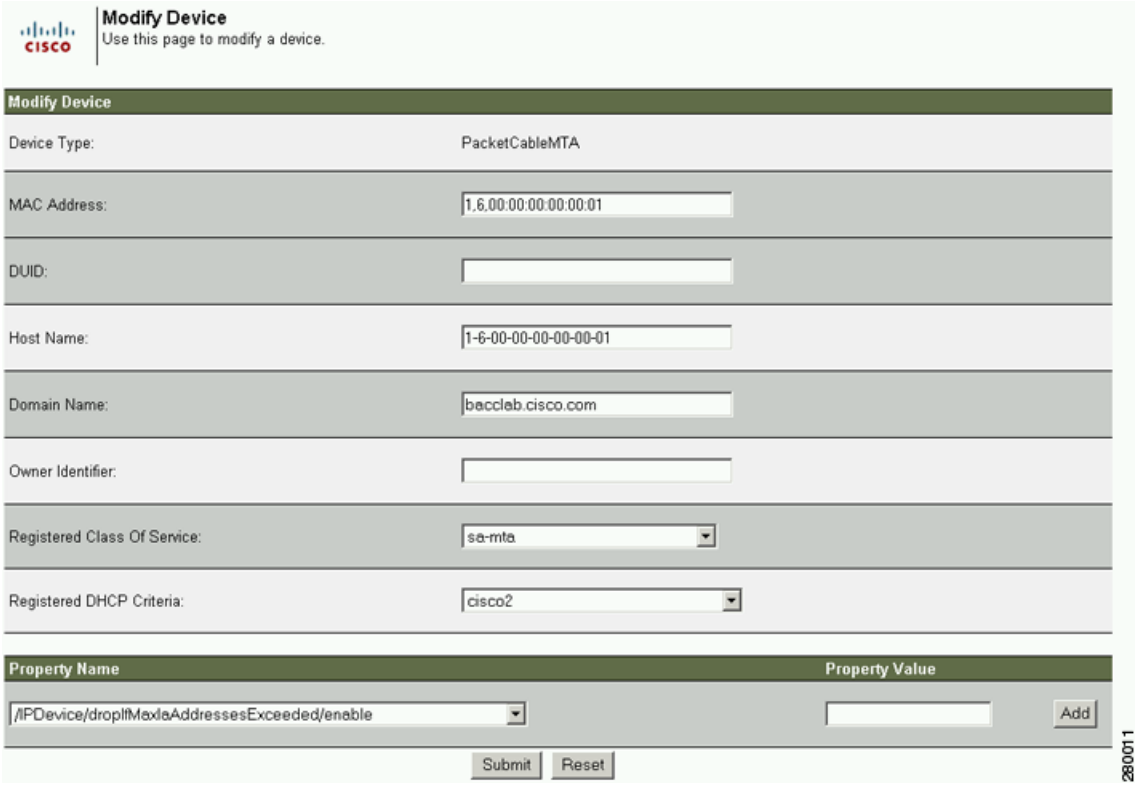
d. From the drop-down list corresponding to Registered Class of Service, select **sa-mta**. This is the Class of Service that you added in Step 10-h.

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**.

Figure 7-4 lists device details for the SA MTA.

Figure 7-4 Provisioning SA MTA–Device Details



Step 12 Verify if multiple realm support is operational by using an ethereal trace. Refer to 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 7-2), the Linksys CM2P2 MTA (Example 7-3), and the SA WebStar DPX2203 MTA (Example 7-4).

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

Example 7-2 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,"'43:
61:62:6C:65:4C:61:62:73:2C:20:49:6E:63:2E'"
#
# 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 7-3 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,"'43
:61:62:6C:65:4C:61:62:73:2C:20:49:6E:63:2E'"
#
# 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 7-4 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,"'43
:61:62:6C:65:4C:61:62:73:2C:20:49:6E:63:2E'"
#
# 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
```

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, refer to 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 SNMPv3 Cloning on the RDU and DPE for Secure Communication with PacketCable MTAs

BAC 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 BAC 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 BAC 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.

PacketCable Basic eMTA Provisioning

BAC also supports PacketCable Basic, which offers a simpler, DOCSIS-like, non-secure provisioning flow. Table 7-5 describes the BASIC.1 flow using the provisioning workflow in Figure 7-1 on page 7-2.

Table 7-5 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	
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 BAC 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 BAC 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 BAC client. This Inform is useful while debugging configuration file issues.

For additional information about the DOCSIS flow, see Chapter 6, “DOCSIS Configuration.”

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

BAC supports the complete set of PacketCable 1.5 MIBs.

BAC 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

BAC supports both SNMP v2C TRAP and INFORM notifications from the PacketCable MTA.

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 the Configuration File Utility, page 5-21, 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.

BAC 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 BAC to use Euro-PacketCable MIBs, you must change the BAC 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-CABLE-DEVICE-MIB-O
BSOLETE,DOCS-QOS-MIB,CISCO-CABLE-MODEM-MIB,DOCS-IETF-BPI2-MIB,INTEGRATED-SERVICES-MIB,DIFF
SERV-DSCP-TC,DIFFSERV-MIB,TCOMLABS-MIB,PKTC-TCOMLABS-MTA-MIB,PKTC-TCOMLABS-SIG-MIB
```

CHAPTER 8

CableHome Configuration

This chapter 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 chapter assumes that you are familiar with the contents of the CableHome Specification CH-SP-CH1.0-I05-030801.

Non-Secure CableHome Provisioning Flow

It is extremely useful to identify which step in the non-secure CableHome provisioning flow is failing before attempting to diagnose other details. Figure 8-1 provides a summary of the key provisioning flows.

Figure 8-1 Non-Secure CableHome Flow

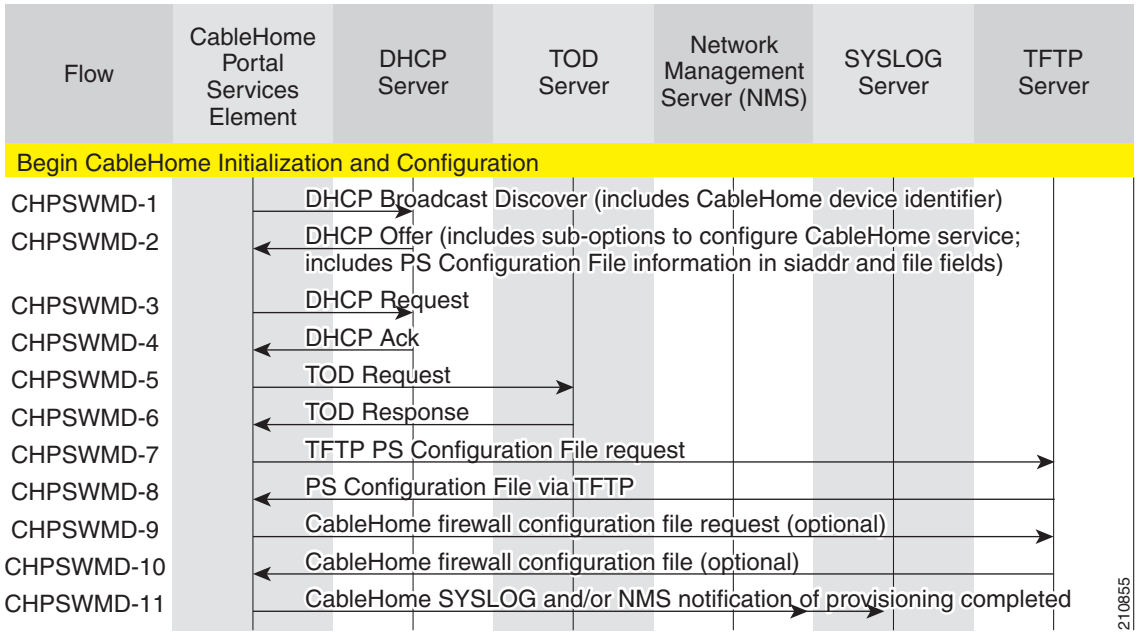


Table 8-1 describes the provisioning flow in a non-secure CableHome deployment.

Table 8-1 CableHome Provisioning Workflow

Step	Workflow	Description
CHPSWMD-1	DHCP Discover	The WAN-MAN obtains its IP lease.
CHPSWMD-2	DHCP Offer	The provisioning system returns a DHCP Offer with CableHome Option 177 suboptions: • 3—Specifies the SNMP Entity Address of the service provider. • 6—Specifies the Kerberos realm name of the provisioning realm. The realm name is required by portal services to permit a DNS lookup for the address of the Key Distribution Center. • 51—Specifies the Kerberos Server IP address, which informs the portal service of the network address of one or more Key Distribution Center servers. This Offer also contains the file information, in the file and siaddr fields, that is required to configure the portal service.
CHPSWMD-3	DHCP Request	The portal service sends the appropriate DHCP server a DHCP Request message to accept the DHCP Offer.
CHPSWMD-4	DHCP Ack	The DHCP server returns a DHCP Ack, which contains the IPv4 address of the portal service. Based on the information received in the DHCP Ack, the portal service modifies the `cabhPsDevProvMode` parameter, which specifies provisioning in the DHCP (non-secure) mode. Also, the Time of Day server address is stored in the `cabhPsDevTimeServerAddr` parameter.
CHPSWMD-5	ToD Request	The portal service initiates Time of Day synchronization with the time servers identified in Option 4 of the DHCP Ack message.
CHPSWMD-6	ToD Response	The Time of Day servers respond with the current time in UTC format.
CHPSWMD-7	PS Configuration File Via TFTP	The portal service sends a TFTP Get Request to obtain a configuration file.
CHPSWMD-8	CableHome Firewall Configuration File Request	The configuration file is downloaded via TFTP. Optionally, if there is a firewall configuration to be loaded and this is the method selected to specify it, the IP address of the name and the hash of the firewall configuration file are included in the configuration file.

Table 8-1 CableHome Provisioning Workflow (continued)

Step	Workflow	Description
CHPSWMD-9	CableHome Firewall Configuration File Request	If the configuration file acquired in step CHPSWMD-8 contains firewall information, portal services may also acquire a firewall configuration file via a TFTP Get Request to the Firewall Configuration TFTP Server. If there is no firewall configuration information in the configuration file, the provisioning process skips steps CHPSWMD-9 and CHPSMWD-10.
CHPSWMD-10	CableHome Firewall Configuration File	The Firewall Configuration TFTP Server sends a TFTP Response containing the firewall configuration file.
CHPSWMD-11	CableHome SYSLOG and/or NMS notification of provisioning completed	Once successfully configured, the portal service sends a syslog message, an SNMP trap, or both, to inform BAC that it has been successfully configured.

Configuring CableHome

This section describes how to configure Cisco Network Registrar, the cable modem configuration system (CMTS).

Configuring Network Registrar

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 *User Guide for Cisco Network Registrar 7.0*.

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 the RDU

To configure CableHome support on the RDU, perform these configurations:

- Configuring CableHome WAN-MAN, page 8-4
- Configuring CableHome WAN-Data, page 8-4

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 the 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 9

Managing Broadband Access Center

This chapter describes the various subcomponents within Broadband Access Center (BAC) that you can use to manage the system. These include:

- BAC Process Watchdog, page 9-1
- Administrator User Interface, page 9-3
- Command-Line Interface, page 9-3
- SNMP Agent, page 9-4
- BAC Tools, page 9-4

BAC Process Watchdog

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

You can use the BAC 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 BAC process watchdog server waits a prescribed length of time before trying to restart.

Note You do not have to use the BAC process watchdog and the SNMP agent to monitor the extensions that are installed on Cisco 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 BAC process watchdog server attempts to restart it and the first restart fails.
3. The BAC process watchdog server waits 2 seconds and attempts to restart the process and the second restart fails.

4. The BAC process watchdog server waits 4 seconds and attempts to restart the process and the third restart fails.
5. The BAC process watchdog server waits 16 seconds and attempts to restart the process.

Using the BAC Process Watchdog from the Command Line

The BAC process watchdog automatically starts whenever the system boots up. Consequently, this watchdog also starts those BAC system components installed on the same system. You can control the BAC watchdog through a simple command-line utility by running the **/etc/init.d/bprAgent** command.

Table 9-1 describes the command-line interface (CLI) commands available for use with the BAC process watchdog.

Table 9-1 BAC CLI Commands

Command	Description
bprAgent start	Starts the BAC process watchdog, including all monitored processes.
bprAgent stop	Stops the BAC process watchdog, including all monitored processes.
bprAgent restart	Restarts the BAC process watchdog, including all monitored processes.
bprAgent status	Gets the status of the BAC 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**—Specifies the RDU server.
- **dpe**—Specifies the DPE server.
- **kdc**—Specifies the KDC server.
- **snmpAgent**—Specifies the SNMP agent.
- **tomcat**—Specifies the administrator.
- **cli**—Specifies the DPE CLI.

Note When the Solaris operating system is rebooted, the BAC process watchdog is first stopped, allowing BAC servers to shut down properly. To shut down or reboot the operating system, use the Solaris **shutdown** command. Remember, the Solaris **reboot** command does not execute application shutdown hooks and kills BAC processes rather than shutting them down. While this action is not harmful to BAC, it may delay server startup and skew certain statistics and performance counters.

The events that trigger an action in the BAC watchdog daemon, including process crashes and restarts, are logged in a log file, *BPR_HOME/agent/logs/agent.log*. The watchdog daemon also logs important events to syslog under the standard `local6` facility.

Administrator User Interface

The BAC administrator user interface is a web-based application for central management of the BAC 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
- Add and edit device information
- Group devices
- View server status and server logs
- Manage users

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

- Chapter 11, "Understanding the Administrator User Interface," describes how to access and configure the BAC administrator user interface.
- Chapter 12, "Using the Administrator User Interface," provides instructions for performing administrative activities involving the monitoring of various BAC components.
- Chapter 13, "Configuring Broadband Access Center," describes tasks that you perform to configure BAC.

Command-Line Interface

The BAC 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 the 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 the 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 Broadband Access Center DPE CLI Reference 4.0* for specific information on the CLI commands that a DPE supports.

SNMP Agent

BAC provides basic SNMP v2-based monitoring of the RDU and DPE servers. The BAC 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 the snmpAgentCfgUtil.sh Tool, page 10-10. For additional information on the DPE CLI, refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0.*

BAC Tools

BAC provides automated tools that you use to perform certain functions more efficiently.

Table 9-2 lists the various tools that this BAC release supports.

Table 9-2 BAC Tools

Tool	Description	Refer...
Configuration File Utility	Used to test, validate, and view BAC template and configuration files.	Using the Configuration File Utility, page 5-21
BAC Process Watchdog	Interacts with the BAC watchdog daemon to observe the status of the BAC system components, and stop or start servers.	Using the BAC Process Watchdog from the Command Line, page 9-2
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 10-4
PacketCable Certificates Tool	Installs, and manages, the KDC certificates that are required by the KDC for its operation.	Using the PKCert.sh Tool, page 14-2
KeyGen Tool	Generates PacketCable service keys.	Using the KeyGen Tool, page 14-8
Changing Network Registrar Properties Tool	Used to change key configuration properties used by BAC extensions that are incorporated into the Network Registrar DHCP server.	Using the changeNRProperties.sh Tool, page 14-11
SNMP Agent Configuration Tool	Manages the SNMP agent.	Using the snmpAgentCfgUtil.sh Tool, page 10-10
Diagnostics Tool	Collects server data related to system performance and troubleshooting.	Troubleshooting Using the Diagnostics Tool, page 16-5
BundleState.sh Tool	Bundles diagnostics data related to server state for support escalations.	Bundling Server State for Support, page 16-10
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 the disk_monitor.sh Tool, page 14-12

CHAPTER 10

Monitoring Broadband Access Center

This chapter describes how you can monitor the central RDU servers and the DPE servers in a Broadband Access Center (BAC) deployment. It describes:

- Logging Events, page 10-1
- Monitoring Servers Using SNMP, page 10-9
- Monitoring Server Status, page 10-15

Logging Events

Logging of events is performed at the RDU and the DPE, 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 administrator user interface.

Log Levels and Structures

The log file structure, illustrated in Example 10-1, 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 BAC.
- Sub-facility—This identifies the BAC subsystem or component.

- Severity Level—The logging system defines seven levels of severity (as described in Table 10-1) 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 Configuring Severity Levels, page 10-3.

Table 10-1 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.
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 10-1 Sample Log File

Domain Name	Data and Time	Facility	Sub-facility	Severity Level	Msg ID	Message
bac.example.com:	2007 9 6 03:06:11 EST:	%BAC-	RDU-	5	0236:	Broadband Access Center Regional Distribution Unit starting up
bac.example.com:	2007 9 6 03:06:15 EST:	%BAC-	RDU-	5	0566:	Initialized API defaults
bac.example.com:	2007 9 6 03:06:15 EST:	%BAC-	RDU-	5	0567:	Initialized Network Registrar defaults
bac.example.com:	2007 9 6 03:06:15 EST:	%BAC-	RDU-	5	0568:	Initialized Server defaults
bac.example.com:	2007 9 6 03:06:18 EST:	%BAC-	RDU-	5	0570:	Initialized DOCSIS defaults
bac.example.com:	2007 9 6 03:06:18 EST:	%BAC-	RDU-	5	0571:	Initialized Computer defaults
bac.example.com:	2007 9 6 03:06:19 EST:	%BAC-	RDU-	5	0573:	Initialized CableHome WAN-MAN defaults
bac.example.com:	2007 9 6 03:06:19 EST:	%BAC-	RDU-	5	0572:	Initialized PacketCable defaults
bac.example.com:	2007 9 6 03:06:19 EST:	%BAC-	RDU-	5	0569:	Created default admin user

Example 10-1 Sample Log File (continued)

Domain Name	Data and Time	Facility	Sub-facility	Severity Level	Msg ID	Message
bac.example.com:	2007 9 6 03:06:20 EST:	%BAC-	RDU-	5	0575:	Database initialization completed in [471] msec
bac.example.com:	2007 9 6 03:06:25 EST:	%BAC-	RDU-	3	0015:	Unable to locate manifest file
bac.example.com:	2007 9 6 03:06:28 EST:	%BAC-	RDU-	3	0280:	Command error

Configuring Severity Levels

You can configure the severity levels of logging for both the RDU and the DPE 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, refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

To configure the log level tool on the RDU, see Using the RDU Log Level Tool, page 10-4.

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*.

RDU 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 10-6.)
- *audit.log*—Records high-level changes to the BAC configuration or functionality including the user who made the change.

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 administrator user interface.

To view the file:

Step 1 Choose the RDU 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 administrator user interface.

To view the file:

Step 1 Choose the RDU 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*.

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 10-2 identifies the available severity levels and the types of messages written to the log file when enabled.

Table 10-2 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 10-2.
- **-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 10-6
- Viewing the Current Log Level of RDU, page 10-6

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.
```

DPE 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, refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

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, refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

You can also view the DPE log file using the BAC administrator user interface.

To view the file:

Step 1 Choose **Servers > DPEs**.

Step 2 Click the link of the DPE whose log file you want to view.

The View Device Provisioning Engines Details page appears.

Step 3 To view the contents of the *dpe.log* file, click the View Details icon against the DPE Log File in the Log Files area.

Network Registrar Logs

BAC generates log messages from Cisco 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 10-3) at the trace level; the number you set makes that number the current setting of the **extension-trace-level** attribute for all extensions.

Table 10-3 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, refer to the *User Guide for Cisco Network Registrar 7.0*.

Monitoring Servers Using SNMP

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

SNMP Agent

BAC provides basic SNMP v2-based monitoring of the RDU and DPE servers. The BAC 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 the snmpAgentCfgUtil.sh Tool, page 10-10. For additional information on the DPE CLI, refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

MIB Support

BAC supports several different MIBs. Table 10-4 summarizes MIB support for each BAC component.

Table 10-4 BAC-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 BAC. This MIB supports the monitoring of multiple BAC 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, refer to the corresponding MIBs files in the *BPR_HOME/rdu/mibs* directory.

Using the snmpAgentCfgUtil.sh Tool

You can use the **snmpAgentCfgUtil.sh** tool to manage the SNMP agent installed on a Solaris 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 computer is 8001.

You can use the RDU SNMP agent for:

- Adding a Host, page 10-10
- Deleting a Host, page 10-11
- Adding an SNMP Agent Community, page 10-11
- Deleting an SNMP Agent Community, page 10-12
- Starting the SNMP Agent, page 10-12
- Stopping the SNMP Agent, page 10-13
- Configuring an SNMP Agent Listening Port, page 10-13
- Changing the SNMP Agent Location, page 10-13
- Setting Up SNMP Contacts, page 10-14
- Displaying SNMP Agent Settings, page 10-14
- Specifying SNMP Notification Types, page 10-14

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 BAC Process Watchdog, page 9-1.

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 BAC Process Watchdog, page 9-1.

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 BAC.

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 BAC Process Watchdog, page 9-1.

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 10-11, 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 BAC Process Watchdog, page 9-1.

Starting the SNMP Agent

You use this command to start the SNMP agent process on a Solaris computer on which BAC is installed.

Note You can also start the SNMP agent by invoking the BAC process watchdog using the **/etc/init.d/bprAgent start snmpAgent** command. For more information, see Using the BAC Process Watchdog from the Command Line, page 9-2.

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 computer on which BAC is installed.

Note You can also stop the SNMP agent by invoking the BAC process watchdog using the **/etc/init.d/bprAgent stop snmpAgent** command. For more information, see Using the BAC Process Watchdog from the Command Line, page 9-2.

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 BAC Process Watchdog, page 9-1.

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 BAC Process Watchdog, page 9-1.

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 BAC Process Watchdog, page 9-1.

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.

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 BAC Process Watchdog, page 9-1.

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
```

Monitoring Server Status

This section describes how you can monitor the performance of the RDU and DPE servers in a BAC deployment. These servers are the central RDU server and the DPE servers.

You can check server statistics from the:

- Administrator user interface
- DPE CLI
- RDU and DPE log files using the administrator user interface or the DPE CLI.

Using the Administrator User Interface

To view server statistics available on the administrator user interface:

1. On the Primary Navigation Bar, click the **Server** tab.

   The Secondary Navigation Bar displays your options: DPEs, NRs, Provisioning Group, RDU.
2. Click the:
   - DPEs tab to monitor all DPEs currently registered in the BAC database.
   - RDU tab to display RDU status and statistics.

If you clicked:

- DPEs—The Manage Device Provisioning Engine page appears.

  Each DPE name on this page is a link to another page that shows the details for that DPE. Click this link to display the details page.

- RDU—The View Regional Distribution Unit Details page appears.

Using the 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 10-2.

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 10-2 show dpe Output

This result occurs when the DPE is running.

```
dpe# show dpe
BAC Agent is running
Process dpe is running
Version BAC 4.0 (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, refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

CHAPTER 11

Understanding the Administrator User Interface

This chapter describes how to access the Broadband Access Center (BAC) administrator user interface and explains the interface. The topics covered are:

- Configuring the Administrator User Interface, page 11-1
- Accessing the Administrator User Interface, page 11-2
- Understanding the Administrator User Interface Icons, page 11-6

Configuring the Administrator User Interface

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 BAC 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. The default value is 1000.
- `/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 BAC is enabled or disabled. You use extensions to augment BAC 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 BAC. The default file size is 4 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 (in seconds) after which an idle session times out. The default period is set as 300 seconds.
- `/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.

Example 11-1 Sample adminui.properties File

```
/adminui/port=49187
/adminui/fqdn=doc.example.com
/adminui/maxReturned=1000
/adminui/pageSize=25
/adminui/refresh=disabled
/adminui/extensions=enabled
/adminui/maxFileSize=40000000
/adminui/refreshRate=90
/adminui/timeout=300
/adminui/noOfLines=250
```

Accessing the Administrator User Interface

You can access the BAC user interface from any computer that can access the URL corresponding to the BAC application.

Logging In

You can log in to the BAC user interface as an administrative user, a Read/Write user, or a Read-Only user. Though each user type has different capabilities, as described in User Management, page 12-1, you access the user interface in the same way.

Complete this procedure to access the BAC administrator user interface:

Step 1 Launch your web browser.

Table 11-1 lists the browsers supported in this BAC release.

Table 11-1 Browser Platform Support

Platform	Supported Browsers
Windows 2000 (Service pack 2)	Internet Explorer 6.0 and higher Firefox 1.5 and higher
Red Hat Linux (7.1)	Firefox 1.5 and higher
Solaris (2.9)	Firefox 1.5 and higher

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

`http://machine_name:port_number/`

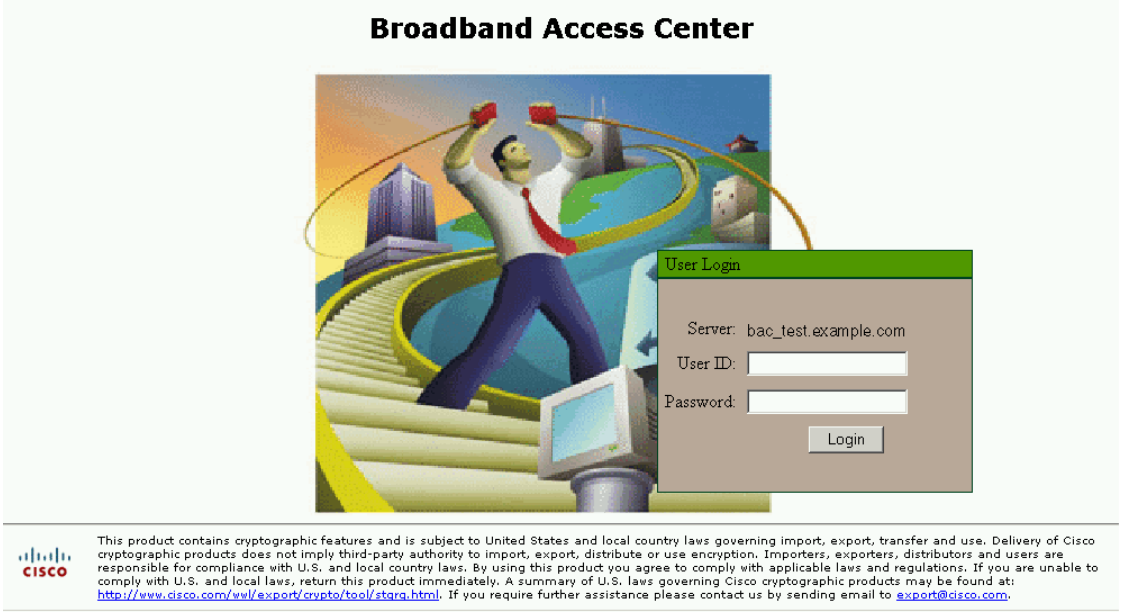
- *machine_name*—Identifies the computer on which the RDU is running.

  Note To access the administrator user interface via HTTP over SSL, also known as HTTPS, enter: `https://machine_name:port_number/`

- *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

The main login page, as shown in Figure 11-1, appears.

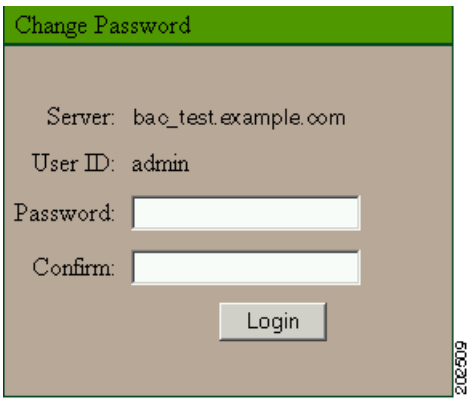
Figure 11-1 Login Page



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

If you are logging in for the first time, the Change Password screen, as shown in Figure 11-2, appears.

Figure 11-2 Change Password Screen

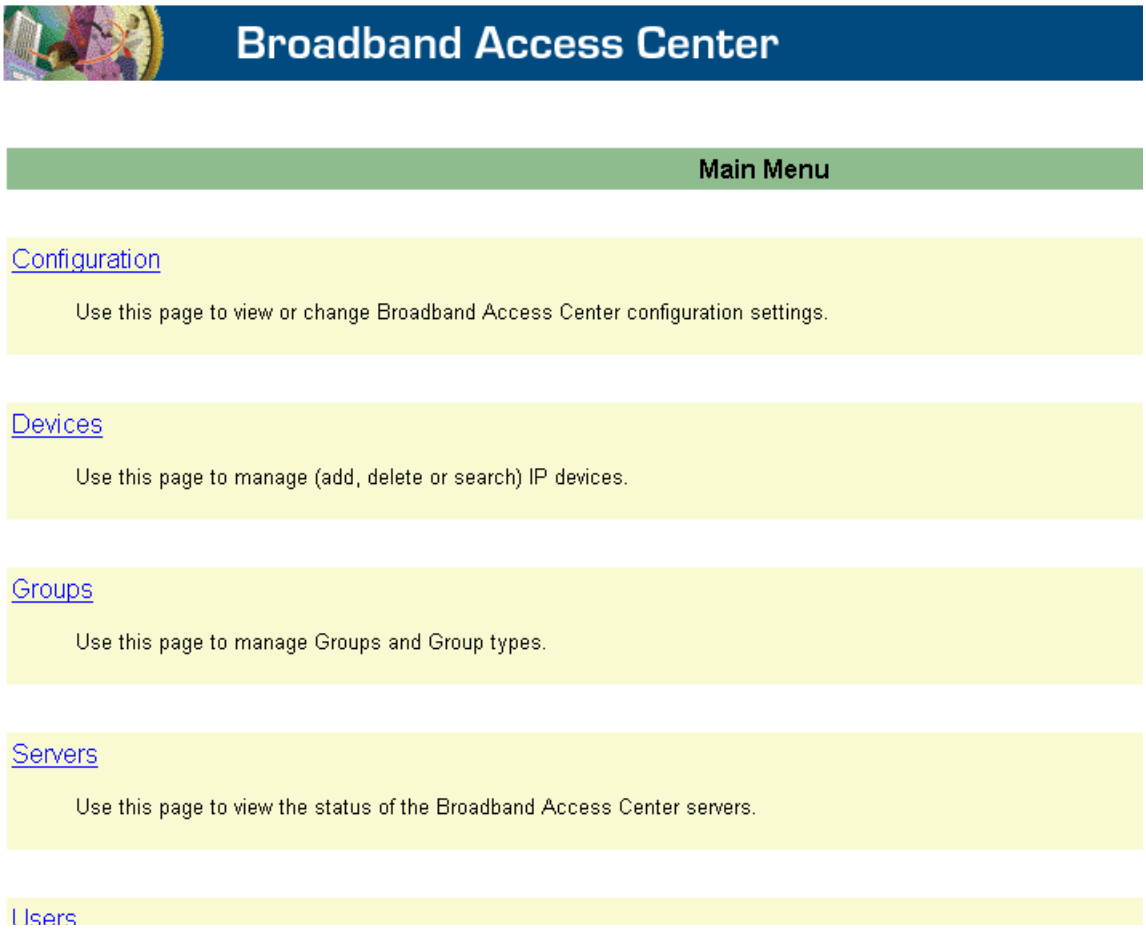


Enter a new password and confirm it. Ensure that the password that you enter has at least 8 characters.

Step 4 Click **Login**.

The Main Menu page, as shown in Figure 11-3, appears.

Figure 11-3 Main Menu Page



Note You can use the link at the top of the page to add the license file. For details, see Managing Licenses, page 13-22.

Logging Out

Complete this procedure to log out of BAC:

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 11-1.

Understanding the Administrator User Interface Icons

The BAC administrator user interface features icons, which you can use to perform specific functions. Table 11-2 defines these icons.

Table 11-2 Administrator User Interface Icons

Icon	Description
	View Details icon—Enables you to view details of a given device or file.
	Delete icon—Deletes a specific object.
	Export icon—Exports the contents of a specific file to the client computer.

These icons are used in sections describing procedures that you perform via the administrator user interface. These sections are found in:

- Chapter 12, "Using the Administrator User Interface."
- Chapter 13, "Configuring Broadband Access Center."

CHAPTER 12

Using the Administrator User Interface

This chapter describes the administration tasks performed from the Broadband Access Center (BAC) administrator user interface. These tasks mainly involve monitoring the actions of various BAC components and include:

- User Management, page 12-1
- Device Management, page 12-4
- Node Management, page 12-17
- Viewing Servers, page 12-21

Note The procedures described in this chapter are presented in a tutorial manner. Wherever possible, examples are included to illustrate the possible results of each procedure.

User Management

Managing users involves adding, modifying, and deleting users who administer BAC. Depending on your user type, you can use this menu to add, modify, and delete users. This menu displays all users configured to use BAC and identifies their user types.

There are three types of BAC users: an Administrator, a Read/Write user, and a Read-Only user. Each has different levels of access, with unique permissions to ensure access control and the integrity of provisioning data.

The assigned user type appears near the top-right corner of every screen on the administrator user interface.

Administrator

BAC recognizes only one administrator and allows this user to view, add, modify, and delete device data, and create other users. As an administrator, you can also change other users' permissions from Read/Write to Read Only, and vice versa. In addition, you can also change the passwords of any other user type.

You cannot delete the administrator user.

Read/Write User

As a Read/Write user, you can perform the same functions as the administrator except creating other users, changing the user types of others, or changing their passwords. Read/Write users can change their own passwords.

Read-Only User

As a Read-Only user, you have basic access including the ability to change your password and to view, but not change, device data. You cannot perform any action that is considered disruptive; you cannot, for example, perform reset or regenerate configurations.

Note During migration from an acceptable previous release to BAC 4.0, all migrated users are assigned Read/Write privileges.

You can add and delete users only if you are logged in as the Administrator.

The following sections contains instructions for managing BAC users including:

- Adding a New User
- Modifying Users
- Deleting Users

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 do have to determine the type of user: a Read/Write user or a Read-Only user.

Note BAC comes with one **Administrator** user already created; you cannot create an administrator as a new user.

To add a new user:

Step 1 Click the **Users** tab.

The Manage Users page appears.

Step 2 Click **Add** to display the Add User page.

Step 3 Enter the new user's username.

Step 4 Enter a password and confirm it. Ensure that the password that you enter has at least 8 characters.

Step 5 Click the appropriate radio button to determine the new user's role. See earlier sections for descriptions of each user type.

Step 6 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 7 Click **Submit**.

The Manage Users page appears with the new user added.

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.

Modifying Users

Although any user type can modify their password and user description, only the administrator can modify another user's information.

To change user properties:

Step 1 Click the **Users** tab.

The Manage User page appears.

Step 2 Click the correct username to access the Modify User page for that user.

Step 3 Make the necessary changes to the password, user type (you must be logged in as the Administrator), and the user's description.

Step 4 Click **Submit**.

The Manage Users page appears with the changed user information.

Deleting Users

Only the administrator can delete any other user that appears in the Manage Users page. You cannot delete the default user, called **admin**. To delete a user:

Step 1 Click **Users.**

The Manage User page appears.

Step 2 Click the **Delete** icon ( ) corresponding to the user you want to delete.

The Delete User dialog box appears.

Step 3 Click **OK**.

The Manage Users page appears without the deleted user.

Device Management

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 12-4.
- Add, modify, or delete devices in the RDU database. See:
  - Adding Device Records, page 12-13
  - Modifying Device Records, page 12-14
  - Deleting Devices, page 12-14
- View device data, such as configuration, and properties. See Viewing Device Details, page 12-9.
- Regenerate device configurations. See Regenerating Device Configurations, page 12-14.
- Relate and unrelate any device to a specific node. See Relating and Unrelating Devices, page 12-15.
- Reset, or reboot, a device. See Resetting Devices, page 12-17.

Manage Devices Page

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 BAC, 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, Node Name (Node Type), and Owner ID.

From the Manage Devices page, you can perform these searches:

- 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 16-2, 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.
- IP Address Search—Searches by returning all devices on the network that currently have the specified DHCP leased IP address.

- 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 16-2, for information on how you can effectively use this search criteria.
- Node search—Searches devices that are part of a particular node or node type.
- 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.
- Provisioning Group Search—Searches by using the provisioning group to which the device belongs.
- Class of Service search includes:
  - Registered Class of Service search—Searches by using the Class of Service that a device has been provisioned with.
  - Related Class of Service search—Searches by using both the registered and selected Class of Service.
  - 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.
- DHCP Criteria search includes:
  - Registered DHCP Criteria search—Searches for devices that belong to certain DHCP Criteria.
  - Related DHCP Criteria search—Searches using both the registered and selected DHCP Criteria.
  - 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.

Some searches that you can perform allow the use of a wildcard character (*) to enhance the search function. BAC provides specific wildcards for each search, as described in Table 12-1.

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 12-1 Searches Supported for Device Management

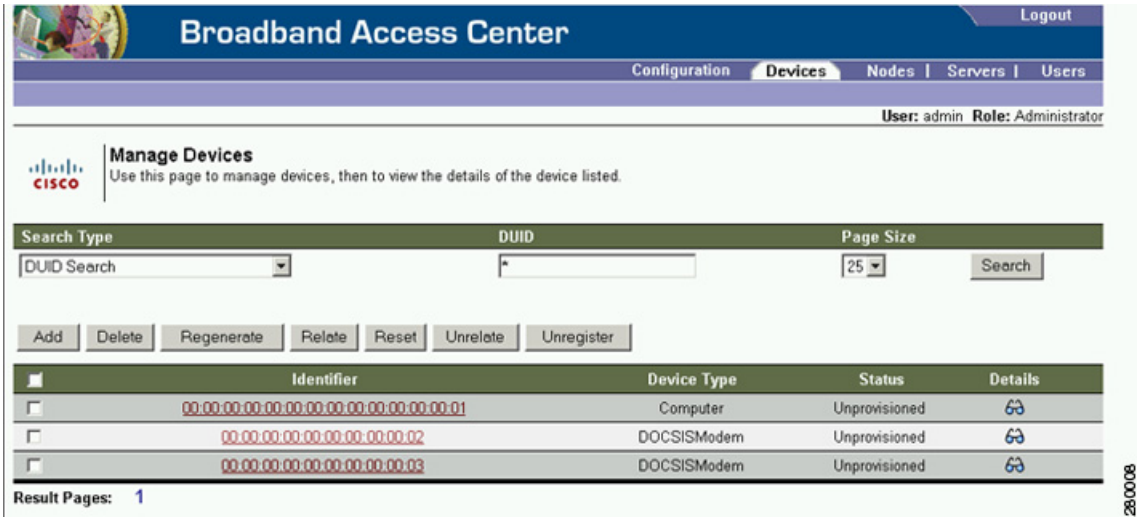
Menu Search	Search Type Option
DUID Search	Complete DUID or partial DUID followed by a wildcard character (*) at the end of the string. For example, to search for a device with DUID 00:03:00:01:02:03:04:05:06:a0, you can try 00:03:*
FQDN Search	Complete FQDN or partial FQDN string beginning with a wildcard (*) character. For example, to search for a device with the FQDN IGW-1234.EXAMPLE.COM, you can try: • *.example.com • *.com • *
IP Address Search	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	Complete MAC address or partial MAC address followed by a wildcard (*) character at the end of the string. For example, to search for a device with the MAC address 1,6,aa:bb:cc:dd:ee:ff, you can try 1,6,*
Node Search	Node Name (Node Type)—Nodes and node type from the drop-down list. The options could be the default **system-diagnostics (system)** option or any other node that you defined.
Owner ID Search	Owner ID Wildcard searches are not supported. 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	Provisioning Group name From the drop-down list, select the **default** option or any other provisioning group that you set up.

Table 12-1 Searches Supported for Device Management (continued)

Menu Search	Search Type Option
Registered Class of Service Search	Class of Service (Type) From the drop-down list, select the default option or any one that you defined as Registered Class of Service.
Registered DHCP Criteria Search	DHCP Criteria (Type) From the drop-down list, select the default option or any one that you defined as Registered DHCP Criteria.
Related Class of Service Search	Class of Service (Type) From the drop-down list, select the default option or any one that you defined as Related Class of Service.
Related DHCP Criteria Search	DHCP Criteria (Type) From the drop-down list, select the default option or any one that you defined as Related DHCP Criteria.
Selected Class of Service Search	Class of Service (Type) From the drop-down list, select the default option or any one that you defined as Selected Class of Service.
Selected DHCP Criteria Search	DHCP Criteria (Type) From the drop-down list, select the default option or any one that you defined as Selected DHCP Criteria.

Figure 12-1 identifies a sample Manage Devices page featuring a search for devices using the DUID search option.

Figure 12-1 Manage Devices Page



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 BAC Tomcat process for the administrator user interface:

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

Device Management Controls

The device management controls are located directly below the search function fields and are generally used with the search function. For example, you might search for devices belonging to a specific group of devices in order to perform some sort of management function.

The following buttons are available, although each management function may not be available depending on the search type you use.

- **Add**—Use the Add button to add a new device to the RDU database. See Adding Device Records, page 12-13.
- **Delete**—Use the Delete button to delete a device from the RDU database. See Deleting Devices, page 12-14.
- **Regenerate**—Use the Regenerate button to force immediate regeneration of configurations for selected devices. See Regenerating Device Configurations, page 12-14.
- **Relate**—Use the Relate button to associate a device (using its MAC address or its DUID) with a specific node. See Relating and Unrelating Devices, page 12-15.
- **Reset**—Use the Reset button to automatically reboot a device.
- **Unrelate**—Use the Unrelate button to cancel the relationship between a selected device and the node that the device is currently related to. See Relating and Unrelating Devices, page 12-15.

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 12-9.

Viewing Device Details

You can view the details of any device identified in the search results. To view any device details, click the **View Details** icon ( ) corresponding to the device you want to view, and the View Device Details page appears.

Note The information that appears in the View Device Details page largely depends on the type of device you choose. The sample figures used in Table 12-2 identify the details that typically appear for most devices.

Table 12-2 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.
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.
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.

Table 12-2 View Device Details Page (continued)

Field or Button	Description
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.
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 BAC 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.

Table 12-2 View Device Details Page (continued)

Field or Button	Description
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 Node Name (Node Type)	Identifies the nodes to which this device is related. This is an active link that, if clicked, displays the appropriate Modify Node page. See Node Management, page 12-17.
DHCPv4 Information	
Note This section does not appear unless the device has discovered DHCPv4 data.	
DHCP Inform Dictionary	Identifies additional information that the Cisco Network Registrar extensions send to the RDU when requesting the generation of a configuration. This is for internal BAC use only.
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 BAC use only; it should always be empty.
DHCP Environment Dictionary	This field is for internal BAC 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 12-2 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 Network Registrar extensions send to the RDU when requesting the generation of a configuration. This is for internal BAC 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 BAC use only; it should always be empty.
DHCPv6 Environment Dictionary	This field is for internal BAC 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 BAC. 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.**

The Add Device page appears.

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**.

Modifying Device Records

To modify a device record:

Step 1 From the Manage Devices page, click the Identifier link corresponding to the device.

The Modify Device page appears.

Step 2 Enter data that you want to add or change. You can modify any existing property name/value pairs by clicking **Add,** or delete any of them by clicking **Delete**.

Step 3 Click **Submit** to save the changes made to this device.

To delete the values that you entered, click **Reset**.

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 readd the device. If restoration of a backed-up database becomes necessary, see Database Restore, page 15-6.

To delete a device record:

Step 1 From the Manage Devices page, locate the device that you want to delete. 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 **Delete**.

The device record stored in the RDU database is removed.

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, BAC 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, 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.

Some configurations cannot be automatically regenerated because BAC 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 nodes and that you use nodes 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 node, which is in turn associated with a specific node type.

By relating a device to a specific node, information indicating that the device is related to a specific node is stored in the database. If you relate the device to the predefined **system-diagnostics (system)** node, you can use available information to troubleshoot potential problems.

Relating a Device to a Node

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 node. 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 Node page appears.

Step 4 Select the node type from the drop-down list and the node from the list of defined nodes.

Note To select multiple groups from the Nodes list, press **Control** or **Shift**.

Step 5 Click **Submit**.

To verify if the device is related to the node you specified, click the View Details icon corresponding to the device. On the Device Details page that appears, check the status against Related Node Name (Node Type).

Unrelating a Device from a Node

To unrelate a device from a node:

Step 1 From the Manage Devices page, locate the device that you want to unrelate from a node.

Step 2 Check the check box corresponding to the device identifier, and click the **Unrelate** button.

The Unrelate Device from Node page appears.

Step 3 From the list of defined nodes, select the node from which you want to unrelate the device.

Note To select multiple groups from the Nodes list, press **Control** or **Shift**.

Step 4 Click **Submit**.

The Manage Devices page appears.

Searching Devices in a Node

To search for devices belonging to a particular node:

Step 1 From the Manage Devices page, select the Node Search option from the drop-down list under Search Type.

The Node Name (Node Type) appear.

Step 2 From the Node Name (Node Type) drop-down list, select the name of the node to which the devices are associated.

Step 3 Click **Search**.

The devices related to the node 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.

Node Management

Node management allows you to create, change, and delete nodes and node types. Within the context of BAC, node types can be considered as groups of nodes, while nodes themselves make up the node type.

Managing Node Types

Access the Manage Nodes page by selecting Nodes from the Main Menu or the primary navigation bar. Node Type is the default setting when this page appears.

Adding a Node Type

To add a new node type:

Step 1 From the Manage Nodes page, click **Add**.

The Add Node Type page appears.

Step 2 Enter a name for the new node type.

Note If you previously added custom properties, you can select 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.

The new node type appears.

Step 3 Click **Submit**.

The new node type is recorded in the RDU, and the Manage Node page appears with the new node type added.

Modifying Node Types

To modify node type properties:

Step 1 From the Manage Nodes page, click the specific node type.

The Modify Node Type page appears.

Note If you previously added custom properties, you can make the necessary changes to the Property Name/Property Value pairs. If you need to delete a specific pair, click **Delete** next to that pair.

Step 2 Click **Submit.**

The Manage Node page appears with the changed description.

Deleting Node Types

To delete node types:

Step 1 From the Manage Nodes page, click the **Delete** icon ( ) corresponding to the node type you want to delete.

Step 2 In the confirmation dialog box that appears, click **OK** to delete the selected node type.

The Manage Nodes page appears without the deleted node type.

Managing Nodes

You can create and modify nodes, delete unwanted nodes, relate and unrelate nodes and node types, and view the devices that you associated with a node.

Adding a New Node

To add a new node:

Step 1 On the Manage Nodes page, select **Nodes** from the Search Type drop-down list.

Step 2 Click **Add.**

The Add Node page appears.

Step 3 Enter the new node name and select the appropriate Node Type for this node.

Note If you previously added custom properties, you can select 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 node is recorded in the RDU, and the Manage Nodes page appears with the new node added.

Searching for Devices in a Node

To view devices associated with a node:

Step 1 From the Manage Nodes page, select the Nodes option from the Search Type drop-down list.

Step 2 You can choose to search either by node type or node name.

- By Node Type—Provides a drop-down list of predefined nodes.
- By Node Name—Provides a Node or Node 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 node.

Doing this displays the Node Search function on the Manage Devices page.

Step 5 From the Manage Devices page, select the appropriate Node Type. See Searching for Devices, page 12-4, for additional information on search functions.

The devices associated with the node appear.

Modifying a Node

To modify node properties:

Step 1 From the Manage Nodes page, click the desired node link.

The Modify Node page appears.

Note If you previously added custom properties, you can make the necessary changes to the Property Name/Property Value pairs. If you need to delete a specific pair, click **Delete** next to that pair.

Step 2 Click **Submit.**

The Manage Nodes page appears with the changed description.

Deleting Nodes

You can delete any node that appears in the Manage Nodes page by checking the check box corresponding to the node and clicking **Delete.**

The node is deleted from the database.

Relating/Unrelating Node Types to Nodes

The relate and unrelate functions are used to establish a relationship between specific nodes and node types.

To either relate or unrelate this relationship:

Step 1 From the Manage Nodes page, choose Nodes from the Search Type drop-down list.

Step 2 Choose the node type for which you want to relate or unrelate nodes using the node type or node name search criteria.

Step 3 Click **Search**.

The specified node appears.

Step 4 Click the Relate to Node or Unrelate from Node link.

Depending on the link you clicked, either the Relate Node or the Unrelate Node page appears.

Step 5 Select the appropriate Node Type from the drop-down list, and select the node to which the node is to be related or unrelated.

Step 6 Click **Submit**.

The Manage Nodes page appears.

Viewing Node Details

To view details relating to a node:

Step 1 From the Manage Nodes page, select the Nodes option from the Search Type drop-down list.

Step 2 Using the node type or node name search criteria, choose the node whose details you want to view.

Step 3 Click **Search**.

Step 4 Click the link corresponding to the Node whose details you want to view.

The Modify Node page appears, with details of the Node Name and Node Type.

Viewing Servers

This section describes the BAC server pages:

- Viewing Device Provisioning Engines, page 12-21
- Viewing Network Registrar Extension Points, page 12-25
- Viewing Provisioning Groups, page 12-27
- Viewing Regional Distribution Unit Details, page 12-29

Viewing Device Provisioning Engines

The Manage Device Provisioning Engines page (**Servers > DPEs**) lets you monitor the list of all DPEs currently registered with the BAC 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 12-3.

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 12-3 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.

Table 12-3 View Device Provisioning Engines Details Page (continued)

Field or Button	Description
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 2-7. **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.
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.
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.

Table 12-3 View Device Provisioning Engines Details Page (continued)

Field or Button	Description
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 2-15.
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.
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.

Table 12-3 View Device Provisioning Engines Details Page (continued)

Field or Button	Description
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	
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.

Table 12-3 View Device Provisioning Engines Details Page (continued)

Field or Button	Description
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.

Viewing Network Registrar Extension Points

The Manage Network Registrar Extension Points page (**Servers > NRs**) lists the extension points for all Network Registrar servers that have been registered with the RDU, and are configured for use with BAC. 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 12-4.

Table 12-4 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.

Table 12-4 View Network Registrar Extension Point Details Page (continued)

Field or Button	Description
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 2-7. **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.
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.
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.

Table 12-4 View Network Registrar Extension Point Details Page (continued)

Field or Button	Description
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.

Viewing 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 12-5.

Table 12-5 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.
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.

Table 12-5 View Provisioning Groups Details Page (continued)

Field or Button	Description
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 the device type support that DPEs in the provisioning group, based on their capabilities, register with the RDU at startup. If the field is Disabled, it means that the provisioning group is not capable of supporting a given device type or feature. See Provisioning Group Capabilities, page 2-15. 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 BAC 4.0 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 12-5 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 BAC 4.0.
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 BAC 4.0.

Viewing Regional Distribution Unit Details

The RDU option, from the Servers menu, displays details of the RDU as described in Table 12-6.

Table 12-6 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.
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.

Table 12-6 View Regional Distribution Unit Details Page (continued)

Field or Button	Description
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	
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.

Table 12-6 View Regional Distribution Unit Details Page (continued)

Field or Button	Description
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.	

CHAPTER 13

Configuring Broadband Access Center

This chapter describes the Broadband Access Center (BAC) configuration tasks that you perform by selecting the options in the Configuration menu:

- Configuring Class of Service, page 13-1
- Configuring Custom Properties, page 13-5
- Configuring Defaults, page 13-6
- Configuring DHCP Criteria, page 13-15
- Managing Files, page 13-17
- Managing Licenses, page 13-22
- Managing RDU Extensions, page 13-25
- Publishing Provisioning Data, page 13-28
- Automatic FQDN Generation, page 13-30

Configuring Class of Service

Using the BAC administrator user interface, 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 BAC administrator user interface to add, modify, view, or delete any selected Class of Service.

Table 13-1 identifies the fields and buttons that appear when you click **Configuration > Class of Service > Manage Class of Service**.

Table 13-1 Manage Class of Service Page

Field or Button	Description
Class of Service	
Class of Service	A drop-down list that identifies the technology Class of Service that you can search for. Available options are: • CableHome WAN-Data • CableHome WAN-MAN • Computer • DOCSIS Modem • PacketCable Multimedia Terminal Adapter (MTA) • STB **Note** For additional information on these areas of technology, see Configuring Defaults, page 13-6.
Class of Service	Displays the names of Class of Service objects.

Adding a Class of Service

To add a specific Class of Service:

Step 1 From the Manage Class of Service page, select the device type for which you want to add a Class of Service using the Class of Service drop-down list.

Step 2 Click **Add**.

The Add Class of Service page appears. This page identifies the various settings for the selected Class of Service.

Step 3 Enter the name for the new Class of Service, and choose the device type from the Class of Service 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 4 Choose a property and enter its corresponding value in the Property Value field. For example, if you choose as 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 DOCSISModem 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.

BAC 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 5 Click **Add** to add the property to the Class of Service.

Step 6 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 add, delete, or modify Class of Service properties:

Step 1 From the Manage Class of Service page, select the Class of Service for the specific device type.

The Modify Class of Service page appears.

- To add a new property to the selected Class of Service:
  - Select the first property that you want assigned to the selected Class of Service from the Property Name drop-down list and, after choosing the appropriate value for that property, click **Add**.
  - Repeat for any other properties that you want to assign to the selected Class of Service.
- To delete a property for the selected Class of Service:
  - Locate the unwanted property in the list immediately above the Property Name drop-down list.
  - Click **Delete**.
- To modify the value currently assigned to a property:
  - Delete the appropriate property as described above.
  - Add the same property again with the new property value.

Note If you delete a property that is required for your business process, add the property again and select the appropriate value before you submit the change.

Step 2 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.

Step 3 Click **OK.**

The modified Class of Service is available in the Manage Class of Service page.

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.

Tip When large numbers of devices associated with a Class of Service need to be deleted, use the BAC 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. If you try to delete a Class of Service with devices associated with it, this error message appears:

```
The following error(s) occurred while processing your request.
Error: Class Of Service [sample-CoS] has devices associated with it, unable to delete

Please correct the error(s) and resubmit your request.
```

The specific Class of Service is specified within the error message. This example uses *sample-CoS*.

To delete a Class of Service:

Step 1 From the Manage Class of Service page, select the Class of Service for the specific device type that you want to delete.

Step 2 Click the **Delete** icon ( ) for that Class of Service.

A confirmation dialog box appears.

Step 3 Click **OK**.

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 BAC Custom Properties**. You use this page to add or delete custom properties.

Caution Although you can delete custom properties if they are currently in use, doing so could cause extreme difficulty to other areas where the properties are in use.

After the custom property is defined, you can use it in the property hierarchy. See Property Hierarchy, page 4-7.

Additionally, properties can be configured on Node and Node Type objects, but they will not be part of the property hierarchy.

Adding a Custom Property

To add a custom property:

Step 1 From the Manage BAC Custom Properties page, click **Add**.

The Add Custom Property page appears.

Step 2 Enter the name of the new custom property.

Step 3 Choose a type for the custom property from the options available in the drop-down list.

Step 4 Click **Submit**.

After the property is added to the database, the Manage BAC Custom Properties page appears.

Deleting a Custom Property

To delete a custom property:

Step 1 From the Manage BAC Custom Properties page, identify the custom property to be deleted.

Step 2 Click the **Delete** icon corresponding to the custom property.

The confirmation dialog box appears.

Step 3 Click **OK**.

The Manage BAC Custom Properties page appears with the custom property deleted from the database.

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 13-7
- Computer Defaults, page 13-7
- DOCSIS Defaults, page 13-8
- Network Registrar Defaults, page 13-9
- PacketCable Defaults, page 13-11
- RDU Defaults, page 13-11
- System Defaults, page 13-12
- STB Defaults, page 13-14

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 13-2.

Table 13-2 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 Automatic FQDN Generation, page 13-30, for additional information.

Computer Defaults

When you select the Computer Defaults link, the list of default values currently applied to the computers supported by BAC appears. Refer to Table 13-2 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 BAC appears. Refer to Table 13-3 for the description of all fields and buttons that appear on this page.

Table 13-3 Configure Defaults–DOCSIS Defaults Page

Field or Button	Description
Extension Point	Identifies the extension point to execute when generating a configuration for a DOCSIS device.
Disruption Extension Point	Identifies the extension point to be executed to disrupt a DOCSIS 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 device. New, unrecognized 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 devices are assigned this default DHCP Criteria. Use the drop-down list to select a new default value.
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.	
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—Automatic FQDN generation is disabled. **Note** See Automatic FQDN Generation, page 13-30, for additional information.
CMTS Shared Secret	Identifies the character string that BAC 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.

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

BAC provides Cisco Network Registrar (NR) extension points that allow BAC to pull information from incoming DHCP packets to detect a device's technology. The extension points also let BAC 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 13-4 identifies the fields that appear on this page.

Table 13-4 Configure Defaults–Network Registrar Defaults Page

Field or Button	Description
NR Extension Point Settings (BAC 2.6, 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 (BAC 4.0)	
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**.

Table 13-4 Configure Defaults–Network Registrar Defaults Page (continued)

Field or Button	Description
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.
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. Table 13-5 identifies the fields that are unique to this defaults page.

Table 13-5 Configure Defaults–PacketCable Defaults Page

Field or Button	Description
Extension Point	Identifies the extension point to execute when generating a configuration for a device of this technology.
Disruption Extension Point	Identifies the extension point to be executed to disrupt a device of this technology.
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 device. New, unrecognized 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 devices are assigned this default DHCP Criteria. Use the drop-down list to select a new default value.
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
Automatic FQDN Generation	Identifies whether a fully qualified domain name (FQDN) will be generated.

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, refer to the *User Guide for Cisco Network Registrar 7.0*.

Table 13-6 describes the fields that appear on the RDU Defaults page.

Table 13-6 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.
CCM Server IP Address	Identifies the IP address of the CCM server.
CCM Server Port	Identifies the CCM server port on which BAC communicates.
CCM Server User	Identifies the CCM server username and is used in conjunction with the password fields.
CCM Server Password	Identifies the password used to authenticate the CCM Server User.
CCM Server Confirm Password	Authenticates the CCM Server Password.
CCM Server	Specifies whether the BAC interface to the CCM Server is enabled or disabled.
CCM Server Timeout	Specifies the length of time, in seconds, that BAC attempts to connect with the CCM Server until BAC declares the connection down.
CRS	Identifies whether the Configuration Regeneration Service (CRS) is enabled. There are two options: • Enable—Enables the CRS within BAC. • Disable—Disables the CRS within BAC.

Note See Managing RDU Extensions, page 13-25, for information on RDU extension points.

System Defaults

When you select the Systems Defaults link, the System Defaults page appears. Table 13-7 describes the fields that appear on this page.

Note You can configure the default values using the BAC API.

Table 13-7 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**.
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.

Table 13-7 Configure System Defaults Page (continued)

Field or Button	Description
System Defaults	
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.
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.

STB Defaults

The STB Defaults page identifies those defaults necessary to support the video set-top box (STB), specifically the RNG-200 STB, which is based on the evolving OpenCable Application Platform. Table 13-8 describes the fields that appear on this page.

Table 13-8 Configure Defaults–STB Defaults Page

Field or Button	Description
Extension Point	Identifies the extension point to execute when generating a configuration for an STB.
Disruption Extension Point	Identifies the extension point to be executed to disrupt an STB.
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 an STB. New, unrecognized STB devices are assigned to this Class of Service. Use the drop-down list to select a new default value.

Table 13-8 Configure Defaults–STB Defaults Page (continued)

Field or Button	Description
Default DHCP Criteria	Identifies the current default DHCP Criteria for a specific device technology. New, unrecognized STBs 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—Automatic FQDN generation is disabled. **Note** See Automatic FQDN Generation, page 13-30, for additional information.

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 BAC, 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, refer to the *User Guide for Cisco Network Registrar 7.0,* 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.

Step 5 You can add properties that are added on the DHCP Criteria. Select a Property Name, and enter the appropriate Property Value.

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:

Step 1 On the Manage DHCP Criteria page, click the DHCP Criteria link that you want to modify.

The Modify DHCP Criteria page appears.

Step 2 Make the desired changes to the client class, inclusion and exclusion selection tags, and the property value settings.

Step 3 Click **Submit**.

After successful modification of the DHCP Criteria in the RDU database, the Manage DHCP Criteria page appears.

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:

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.

Step 1 On the Manage DHCP Criteria page, click the **Delete** icon corresponding to the DHCP Criteria you want to delete.

A confirmation dialog box appears.

Step 2 Click **OK**.

The Manage DHCP Criteria page appears.

Managing Files

Using the BAC 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, refer to Template Files–An Overview, page 5-1.

- 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. BAC 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. BAC treats this file type like any other binary file. These firmware images can also include IOS images for Cisco devices.

Table 13-9 describes the fields that appear on the View Files page.

Table 13-9 View Files Page

Field or Button	Description
Search Type	Identifies the types of searches that you can perform for files using the BAC 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 Template—Specifies a configuration template file for CableLabs. – Generic File—Specifies a generic file. – JAR File—Specifies a JAR file. – MIB File—Specifies a MIB 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* extension. An example of an invalid wildcard is bronze*.
Page Size	Identifies the number of results that must appear on a single page.
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 binary 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**.

The Add Files page appears.

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 Template or a Firmware File, you must also complete these steps, otherwise go to Step 6.

a. When adding a CableLabs Configuration Template 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 BAC sets a deliverable status for each file type, you can change the default setting only for a CableLabs Config Template 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

b. In the case of a Firmware File, additionally enter the file version and a suitable description for that version.

Step 5 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.

Viewing Files

To view the contents of a DOCSIS or PacketCable voice technology file:

Step 1 From the View Files page, search for the required file using the file type or file name search options.

Step 2 Click the **View Details** icon ( ) corresponding to the file.

The View File page appears with details of the file contents.

Figure 13-1 identifies sample binary file content.

Figure 13-1 Sample Binary File Content

View File

File name: unprov_packet_cable.bin.

File type: CABLELABS_STATIC_CONFIGURATION.

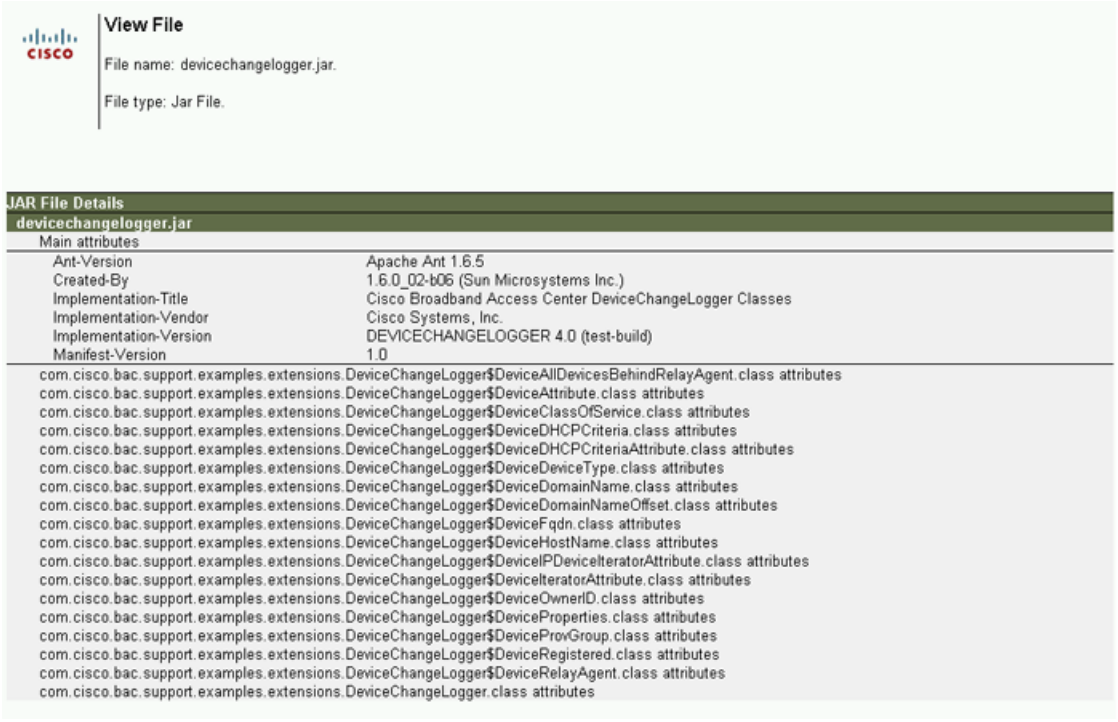
Off.	File bytes.	Option.	Description.	Value.
0	FE0101	254	Telephony Config File Start/End	1
3	0B153013060E 2B06010401A3 0B0202010101 0700020102	11	SNMP MIB Object	.iso.org.dod.internet.private. enterprises.cableLabs.clabProj ect.clabProjPacketCable.pktcMt aMib.pktcMtaMibObjects.pktcMta DevBase.pktcMtaDevEnabled.0,IN TEGER,false(2)
26	0B2E302C0610 2B06010401A3 0B0202020102 010101010418 6D6763702D73 797343412E70 637465737462 65642E636F6D	11	SNMP MIB Object	.iso.org.dod.internet.private. enterprises.cableLabs.clabProj ect.clabProjPacketCable.pktcSi gMib.pktcSigMibObjects.pktcNcs EndPntConfigObjects.pktcNcsEnd PntConfigTable.pktcNcsEndPntCo nfigEntry.pktcNcsEndPntConfigC allAgentId.1,STRING,mgcp-sysCA .pctestbed.com
74	0B2E302C0610 2B06010401A3 0B0202020102 010101020418 6D6763702D73 797343412E70 637465737462 65642E636F6D	11	SNMP MIB Object	.iso.org.dod.internet.private. enterprises.cableLabs.clabProj ect.clabProjPacketCable.pktcSi gMib.pktcSigMibObjects.pktcNcs EndPntConfigObjects.pktcNcsEnd PntConfigTable.pktcNcsEndPntCo nfigEntry.pktcNcsEndPntConfigC allAgentId.2,STRING,mgcp-sysCA .pctestbed.com
122	0B40303E061A 2B06010401A3 0B0202010103 100104495046 4F4E49582E43 4F4D04205265 616C6C792041 6D617A696E67 2054656C6570 686F6E652043 6F6D70616E79	11	SNMP MIB Object	.iso.org.dod.internet.private. enterprises.cableLabs.clabProj ect.clabProjPacketCable.pktcMt aMib.pktcMtaMibObjects.pktcMta DevSecurity.pktcMtaDevRealmTab le.pktcMtaDevRealmEntry.pktcMt aDevRealmOrgName.73.80.70.79.7 8.73.88.46.67.79.77,STRING,Rea lly Amazing Telephone Company

280014

Note The output featured in this graphic has been trimmed.

Figure 13-2 identifies sample JAR file content.

Figure 13-2 Sample JAR File Content



Replacing Files

To replace an existing file:

Step 1 From the View Files page, click the Files link that corresponds to the file you want to replace.

The Replace File page appears. Note that the selected filename already appears on this page.

Step 2 Enter the path and filename of the source file that is to replace the file present in the RDU database. If you do not know the exact name or location of the source file, use **Browse** to navigate to the desired directory and select the file.

Step 3 Click **Submit**.

After submitting the replacement file, a confirmation page appears to indicate that, after replacement, BAC will regenerate configurations for the affected devices.

Step 4 Click **OK**.

The View Files page appears.

The configuration for all devices using this file through a Class of Service is regenerated after the replacement is finished.

Exporting Files

You can copy files to your local hard drive using the export function.

Note The procedure described below assumes that you are using Internet Explorer. This procedure is different if you are using Netscape Navigator.

To export a file:

Step 1 From the View Files page, click the Files link that corresponds to the file you want to export.

Step 2 Identify the file that you want to export.

Step 3 Click the **Export** icon ( ).

You are prompted to either open the file or save it.

Step 4 Return to the BAC user interface.

Deleting Files

To delete an existing file:

Step 1 From the View Files page, locate the file you want to delete using the search option.

The appropriate files appear in the Files list.

Step 2 Choose the appropriate file or files.

Step 3 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 13-1, for additional information.

Managing Licenses

Software licenses are used to activate specific features or to increase the functionality of your installation. This section describes BAC handling of different licenses. For details on the licensing changes in this release and how to obtain your license file, refer to the *Release Notes for Cisco Broadband Access Center 4.0*.

BAC 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.

Caution Do not attempt to deploy into a fully operational network using an evaluation license. When the evaluation license expires, you will not be able to use BAC to provision the devices in your network.

BAC 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.

This BAC release enables licensing using a service file. These licenses allow you to provision a set number of services using BAC. Each service translates to three IP addresses provisioned in the system; thus, a 10,000 service license equates to 30,000 IP addresses. The license file that you receive will contain the number of IP addresses that are licensed, not the number of services that you purchased.

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

Using the service license, you can provision any device type in any combination up to the maximum number that is stated in the license file. The device types that BAC supports in this release are:

- DOCSIS cable modems
- PacketCable MTAs
- CableHome WAN-MAN and WAN-Data devices
- Computers
- Custom CPE

Note You still require separate licenses for the following BAC components:

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

While you must install the DPE license from the administrator user interface, the KDC license continues to be proprietary as in previous BAC releases, and is licensed during BAC installation.

While you cannot install more than one evaluation license, you can replace expired or even valid evaluation licenses with a new evaluation license (with a later expiry date), or a permanent license. While replacing the existing license, ensure that the device limit of the new license at least equals the number of devices that is currently stored in the database.

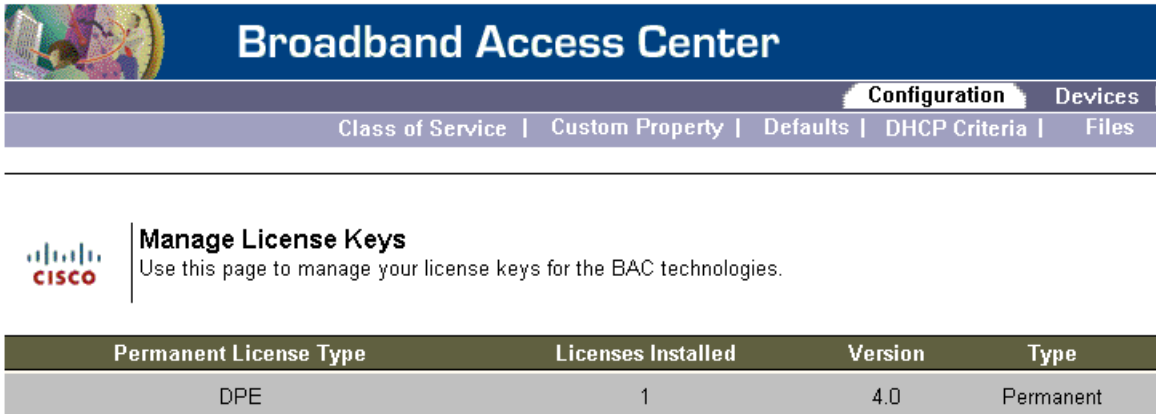
When you upgrade from an evaluation license to a permanent license, you do not have to reinstall the software or reconfigure BAC. You simply have to install the permanent license using the BAC administrator user interface.

Note If you have already installed a permanent license, you cannot install an evaluation license.

You can upgrade a permanent license to increase the number of authorized devices by adding an additional license. When you reach the limit of your number of licensed devices you cannot provision new devices, but existing devices that are already provisioned continue to receive service.

Figure 13-3 identifies a sample Manage License Keys page, which displays a list of service licenses that have been entered for your implementation.

Figure 13-3 Manage License Keys Page



Adding and Modifying a License

Obtain your new license file via the claim process described in the *Release Notes for Cisco Broadband Access Center 4.0*. After you receive your license file, save each file to the system on which you plan to launch the BAC administrator user interface.

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

To add, modify, or upgrade a license:

Step 1 Choose **Configuration > License Keys**.

Note If you are uploading a license for the first time, you can use the license link that appears on the Main Menu.

The Manage License Keys page appears.

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

Or, click **Browse** and navigate to the license file. Remember to include the name of the license file while specifying the pathname.

Step 3 Click **Add/Upgrade**.

The details regarding the number of services and the DPEs that you are licensed to use appear.

Deleting a License

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

The RDU automatically deletes evaluation licenses when:

- You install an evaluation license on an RDU that has an evaluation license already installed. If the evaluation license to be added is valid and will expire after the existing license expires, the installed license is replaced with the new one.
- You upload a permanent license that currently has an evaluation license. If the permanent license is valid, the evaluation license is replaced with the permanent license.

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

To delete a license:

Step 1 Choose **Configuration > License Keys**.

The Manage License Keys page appears.

Step 2 Click the **Delete** button corresponding to the license that you want to delete.

A confirmation dialog box appears.

Step 3 To confirm deleting the license, click **Yes**.

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*.

Managing RDU Extensions

Creating a custom extension point is a programming activity that can, when used with the BAC administrator user interface, allow you to augment BAC behavior or add support for new device technologies.

Before familiarizing yourself with managing extensions, you should know the RDU extension points that BAC 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 13-10 lists the RDU extension points that BAC requires to execute extensions.

Table 13-10 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 BAC do not include any publishing plug-ins.	Optional	No
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

Managing extensions includes:

- Writing a New Class, page 13-26
- Installing RDU Custom Extension Points, page 13-28
- Viewing RDU Extensions, page 13-28

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 the 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, refer to 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 BAC 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 > RDU > 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 the 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 administrator user interface to install it:

Step 1 To add the new JAR file, see Adding Files, page 13-18.

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 13-26, 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 Viewing Regional Distribution Unit Details, page 12-29.

Publishing Provisioning Data

BAC 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 BAC does not ship with any publishing plug-ins. You must create your own plug-ins and load them into BAC in the same way as JAR files are (see Adding Files, page 13-18). 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** on the Primary Navigation bar.

Step 2 Choose **Publishing** on the Secondary Navigation bar.

The Manage Publishing page appears. This page displays a list of all available database plug-ins and identifies the current status of each.

Step 3 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** on the Primary Navigation bar.

Step 2 Choose **Publishing** on the Secondary Navigation bar, and the Manage Publishing page appears.

Step 3 Click the link corresponding to the plug-in you want to modify. The Modify Publishing Plug-Ins page appears.

Table 13-11 identifies the fields shown in the Modify Publishing Plug-Ins page.

Table 13-11 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 4 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 5 Click **Submit** to make the changes to the selected plug-in.

Automatic FQDN Generation

When configuring the PacketCable voice technology, a fully qualified domain name (FQDN) must reside in the BAC database for each voice device, because the KDC queries the registration server for that FQDN. The BAC automatic FQDN generation feature is not limited to any single voice technology; it can be used by any BAC technology.

Automatically Generated FQDN Format

BAC 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 BAC 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 BAC 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-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 BAC 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 BAC property hierarchy.

Properties for Automatically Generated FQDNs

Properties can be defined at any acceptable point in the BAC 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 BAC.

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 Configuring Defaults, page 13-6.

CHAPTER 14

Support Tools and Advanced Concepts

This chapter contains information on, and explains the use of, tools that help you maintain Broadband Access Center (BAC) as well as speed and improve the installation, deployment, and use of this product.

This chapter discusses:

- BAC Tools, page 14-1
- Using the PKCert.sh Tool, page 14-2
- Using the KeyGen Tool, page 14-8
- Using the changeNRProperties.sh Tool, page 14-11
- Using the disk_monitor.sh Tool, page 14-12

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.

BAC Tools

BAC provides automated tools that you use to perform certain functions more efficiently. Table 14-1 lists the various tools that this BAC release supports.

Table 14-1 BAC Tools

Tool	Description	Refer...
Configuration File Utility	Used to test, validate, and view BAC template and configuration files.	Using the Configuration File Utility, page 5-21
BAC Process Watchdog	Interacts with the BAC watchdog daemon to observe the status of the BAC system components, and stop or start servers.	Using the BAC Process Watchdog from the Command Line, page 9-2
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 10-4

Table 14-1 BAC Tools (continued)

Tool	Description	Refer...
PacketCable Certificates Tool	Installs, and manages, the KDC certificates that are required by the KDC for its operation.	Using the PKCert.sh Tool, page 14-2
KeyGen Tool	Generates PacketCable service keys.	Using the KeyGen Tool, page 14-8
Changing Network Registrar Properties Tool	Used to change key configuration properties used by BAC extensions that are incorporated into the Cisco Network Registrar DHCP server.	Using the changeNRProperties.sh Tool, page 14-11
SNMP Agent Configuration Tool	Manages the SNMP agent.	Using the snmpAgentCfgUtil.sh Tool, page 10-10
Diagnostics Tool	Collects server data related to system performance and troubleshooting.	Troubleshooting Using the Diagnostics Tool, page 16-5
BundleState.sh Tool	Bundles diagnostics data related to server state for support escalations.	Bundling Server State for Support, page 16-10
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 the disk_monitor.sh Tool, page 14-12

Using the PKCert.sh Tool

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 the 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 14-3.
  - **-v**—Verifies and normalizes the PacketCable certificate set. See Validating the KDC Certificates, page 14-4.
  - **-z**—Sets the log level for debug output that is stored in the *pkcert.log* file. See Setting the Log Level for Debug Output, page 14-5.

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/solaris/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/solaris/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/solaris/packetcable/certificates/KDC_private_key.pkcs8
File written:
/opt/CSCObac/kdc/solaris/packetcable/certificates/KDC_private_key_proprietary.
File written: /opt/CSCObac/kdc/solaris/packetcable/certificates/KDC_PublicKey.der
File written: /opt/CSCObac/kdc/solaris/packetcable/certificates/KDC.cer
KDC Certificate Successfully Created at
/opt/CSCObac/kdc/solaris/packetcable/certificates/KDC.cer
```

Using this command creates the files */opt/CSCObac/kdc/solaris/packetcable/certificates/KDC.cer* and */opt/CSCObac/kdc/solaris/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 the 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/solaris/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/solaris/packetcable/certificates/CableLabs_Service_Provider_Root.cer
File written: /opt/CSCObac/kdc/solaris/packetcable/certificates/Service_Provider.cer
File written: /opt/CSCObac/kdc/solaris/packetcable/certificates/Local_System.cer
File written: /opt/CSCObac/kdc/solaris/packetcable/certificates/KDC.cer

Service Provider Certificate Chain Written to Destination Directory
/opt/CSCObac/kdc/solaris/packetcable/certificates


File written: /opt/CSCObac/kdc/solaris/packetcable/certificates/MTA_Root.cer

Device Certificate Chain Written to Destination Directory
/opt/CSCObac/kdc/solaris/packetcable/certificates
```

Setting the 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/solaris/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/solaris/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/solaris/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 the 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, refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

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/solaris/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 BAC 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/solaris/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/solaris/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 the changeNRProperties.sh Tool

The BAC installation program establishes values for configuration properties used by BAC 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 BAC 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**.

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 the disk_monitor.sh Tool

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
```

CHAPTER 15

Database Management

This chapter contains information on RDU database management and maintenance. The RDU database is the Broadband Access Center (BAC) central database. The BAC 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 15-7, 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 15-7, 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 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 a number of factors:

- Device objects that the RDU manages
- 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 numbers 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 the disk_monitor.sh Tool, page 14-12, 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 15-7.

Note It is a good practice to monitor disk space utilization to prevent failure. See Using the disk_monitor.sh Tool, page 14-12, 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.

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 /var/backup

Database backup started
Back up to: /var/backup/rdu-backup-20070316-031028

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
```

In this example, all backup database files are stored in a directory called */var/backup/rdu-backup-20070316-031028*. The last subdirectory (*rdu-backup-20070316-031028*) is automatically created with a current timestamp.

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.

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 BAC 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 BAC 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_DATA/ bpr_definitions.sh*. Any change to this file requires that you restart all BAC 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 BAC process watchdog and all BAC 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 BAC process watchdog and all BAC 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, refer to the *Installation and Setup Guide for Cisco Broadband Access Center 4.0*.

CHAPTER 16

Troubleshooting Broadband Access Center

This chapter provides details on how to troubleshoot with Broadband Access Center (BAC). This chapter describes:

- Troubleshooting Checklist, page 16-1
- Troubleshooting Devices by Device ID, page 16-2
- Troubleshooting Using the Diagnostics Tool, page 16-5
- Bundling Server State for Support, page 16-10
- Troubleshooting DOCSIS Networks, page 16-10
- Troubleshooting PacketCable eMTA Provisioning, page 16-11

For a list of FAQs related to BAC provisioning, see Appendix E, "FAQs on Provisioning Broadband Access Center."

Troubleshooting Checklist

While troubleshooting with BAC, use the checklist described in Table 16-1.

Table 16-1 Troubleshooting Checklist

Procedure	Refer to...	Check Off
1. Check if the BAC processes are up on all systems on which BAC components are installed.	Using the BAC Process Watchdog from the Command Line, page 9-2	❑
2. Check the BAC component logs for indications of high-severity errors. These include the information logged for: – RDU – DPE	 RDU Logs, page 10-4 DPE Log, page 10-7	❑
3. View server uptime from the administrator user interface to confirm that the servers are not bouncing.	Viewing Servers, page 12-21	❑
4. View the RDU and DPE service performance statistics from the administrator user interface. Observe any abnormal numbers, such as extended transaction times.	Viewing Servers, page 12-21	❑

Table 16-1 Troubleshooting Checklist (continued)

Procedure	Refer to...	Check Off
5. Check the syslog alerts log.	Appendix A, "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 Broadband Access Center DPE CLI Reference 4.0*.	❑
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 16-3	❑
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 10-4 The **log level** command described in the *Cisco Broadband Access Center DPE CLI Reference 4.0*.	❑

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 node management for one or more specific devices without turning logging on, and without searching through log files for specific device information.

BAC 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 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 BAC 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 12-13.

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, this number is set at 25. To configure the maximum number of devices allowed in troubleshooting mode from the administrator user interface, click the Systems Defaults page via the **Configuration > Defaults** tabs. Enter a value in the Maximum Diagnostics Device Count field.

Relating a Device to a Node

You can troubleshoot a device by relating it to a specific node. Use the Relate function to associate a device, using its MAC address or its DUID, to a specific node, which is in turn associated with a specific node type. (See Relating and Unrelating Devices, page 12-15.) Doing so records an extraordinarily large volume of information for a device; you can then use the information to troubleshoot potential problems.

Table 16-2 identifies a possible workflow using the Relate and Unrelate functions.

Table 16-2 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 node.
3.	Wait a few minutes to ensure that device traffic is passing, or perform a hard boot of the device.
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 node.

Viewing List of Devices in Diagnostics Mode

When you enable troubleshooting for a device, the device is automatically added to a special device node that contains a list of devices in troubleshooting mode. The node type is **system** and the node 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 Node Search.

Step 2 From the Node Name (Node 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 Logging Events, page 10-1.

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 the Diagnostics Tool

You can use the diagnostics tool to collect performance statistics—down to a specific type of statistic—for BAC 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 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 16-10.

Using the 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 16-10.

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 the 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 the 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 the Diagnostics Tool, page 16-5.) 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 BAC 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 the startDiagnostics.sh Tool, page 16-5.

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 BAC and the Cisco uBR7246 CMTS, refer to *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 16-14
- Troubleshooting Scenarios, page 16-15
- Certificate Trust Hierarchy, page 16-18

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 BAC 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 BAC (Class of Service, DHCP Criteria, files, and BAC 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 Chapter 7, "PacketCable Voice Configuration."

This section contains the following topics:

- Components, page 16-11
- Key Variables, page 16-13

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 BAC with PacketCable support, be aware that a correctly configured BAC 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 BAC 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. Refer to Cisco 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 BAC 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 16-13
- Scope-Selection Tags, page 16-14
- MTA Configuration File, page 16-14

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/solaris/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 Refer to Using the PKCert.sh Tool, page 14-2, 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 BAC (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, BAC 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 BAC administrator user interface. Client class can also be used to tie scopes to BAC 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 16-14
- Ethereal, SnifferPro, or Other Packet Capture Tools, page 16-15

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, refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

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 16-3 are possible failures involving eMTAs.

Table 16-3 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.
The MTA device does not appear in the BAC 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 BAC user interface.	Verify that the MTA scope-selection tags match those in the PacketCable DHCP Criteria created, in BAC, for the relevant MTAs.
	The Network Registrar extension point is not properly installed.	Reinstall the Network Registrar extension point. Refer to the *Installation and Setup Guide for Cisco Broadband Access Center 4.0*.
	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 BAC.
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.

Table 16-3 Troubleshooting Scenarios (continued)

If this problem occurs...	Which indicates this potential cause...	To correct it, you should...
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.
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 BAC.	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 3-6, for additional information.	Ensure that all BAC network elements are clock-synced via NTP. Refer to the *Broadband Access Center DPE CLI Reference 4.0*.
	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/solaris/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.

Table 16-3 Troubleshooting Scenarios (continued)

If this problem occurs...	Which indicates this potential cause...	To correct it, you should...
Failure at AP Request/Reply (step 14 in Figure 7-1).	A clock skew error. See PacketCable Workflows, page 3-6, for additional information.	Ensure that all BAC network elements are clock-synced via NTP. Refer to the *Broadband Access Center DPE CLI Reference*.
	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.
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 BAC 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.

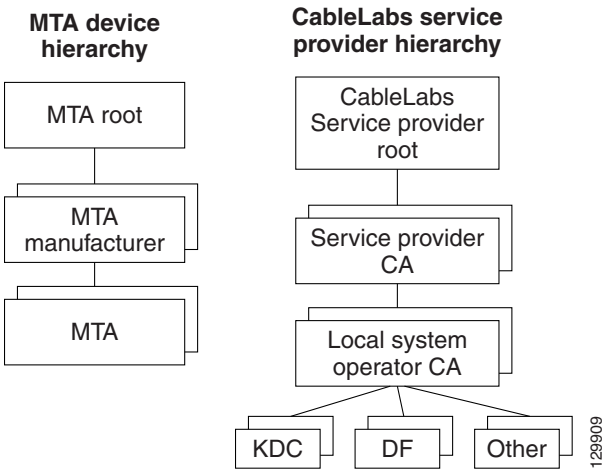
Table 16-3 Troubleshooting Scenarios (continued)

If this problem occurs...	Which indicates this potential cause...	To correct it, you should...
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. Refer to Cisco Network Registrar documentation for information on enabling DDNS.

Certificate Trust Hierarchy

There are two certificate hierarchies affiliated with BAC PacketCable, the MTA Device Certificate Hierarchy and the CableLabs Service Provider Certificate Hierarchy, as shown in Figure 16-1.

Figure 16-1 PacketCable Certificate Hierarchy



Before implementing PacketCable in BAC, 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 node (at the bottom) in the PacketCable certificate trust hierarchy illustrated in Figure 16-1.

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 16-4, 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 16-4 lists the values relevant to the MTA Root Certificate.

Table 16-4 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 16-5 lists the values relevant to the MTA Manufacturer Certificate.

Table 16-5 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 16-6 lists the values relevant to the MTA Device Certificate.

Table 16-6 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 16-7 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 16-7 lists the values relevant to the CableLabs Service Provider Root Certificate.

Table 16-7 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 16-8 lists the values relevant to the CableLabs Service Provider CA Certificate.

Table 16-8 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 16-9 lists the values relevant to the Local System CA Certificate.

Table 16-9 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 16-9 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 16-10 lists the values relevant to the KDC Certificate.

Table 16-10 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 16-11 lists the values relevant to the DF Certificate.

Table 16-11 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 16-12 lists the values relevant to the PacketCable Server Certificate.

Table 16-12 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*] Refer to [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 16-12 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 16-19, for additional information on how to validate certificates.

Table 16-13 lists the values relevant to the CableLabs Code Verification Root CA Certificate.

Table 16-13 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 16-19, 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 16-14 lists the values relevant to the CableLabs Code Verification CA Certificate.

Table 16-14 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 16-14 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 16-15 lists the values relevant to the Manufacturer Code Verification Certificate.

Table 16-15 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. Refer to Certificate Validation, page 16-19, for additional information on how to validate certificates.

Table 16-16 lists the values relevant to the Service Provider Code Verification Certificate.

Table 16-16 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.

APPENDIX A

Alert and Error Messages

This appendix identifies all alert and error messages that Broadband Access Center (BAC) generates, specifically:

- RDU Alerts, page A-2
- DPE Alerts, page A-3
- Watchdog Alerts, page A-4
- Network Registrar Extension Point Alerts, page A-5

BAC generates alerts through the Syslog service. Syslog is a client-server protocol that manages the logging of information on Solaris. BAC 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 BAC log files.

Message Format

When BAC 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 Network Registrar extension points)
  - KDC (Key Distribution Center)
- #—Identifies the severity level in use. Table A-1 describes the different levels.

Table A-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.

RDU Alerts

Table A-2 identifies the RDU alerts.

Table A-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 14, "Support Tools and Advanced Concepts" 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 BAC 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 BAC 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, refer to the *Installation and Setup Guide for the Cisco Broadband Access Center 4.0*.

Table A-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 BAC 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, refer to the *Installation and Setup Guide for the Cisco Broadband Access Center 4.0*.

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*.

DPE 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, refer to the *Cisco Broadband Access Center DPE CLI Reference 4.0*.

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 A-3 identifies the DPE alerts.

Table A-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 A-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 BAC 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, refer to the *Installation and Setup Guide for the Cisco Broadband Access Center 4.0*.
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 A-4 identifies the process watchdog alerts.

Table A-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 A-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 A-4 represents any of these component values:

- rdu
- dpe
- tomcat
- cli
- snmpAgent
- kdc

Network Registrar Extension Point Alerts

Whenever a BAC Network Registrar extension point syslog alert is sent, you can find additional details in the Network Registrar log file.

Table A-5 identifies the process watchdog alerts.

Table A-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 BAC NR extensions have started	The Network Registrar extensions have been started.
NR_EP-6-109: The BAC 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.

APPENDIX B

Option Support

This appendix identifies the technology-specific options that BAC 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 5-14.
- 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 B-1
- PacketCable Option Support, page B-22
- CableHome Option Support, page B-22

DOCSIS Option Support

Table B-1 describes DOCSIS options and identifies the specific version support for each option.

Table B-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	✓	✓	✓	✓
4	Class of Service	Compound	None	True	✓	✓	✓	✓

Table B-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
4.1	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	✓	✓	✓	✓
15.11	RADIUS Realm	NVTASCII	None	False	✓	✓	✓	✓

Table B-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
15.12	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	✓	✓	✓	✓
22	Upstream Packet Classification Encoding	Compound	None	True		✓	✓	✓

Table B-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
22.1	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		✓	✓	✓
22.10.2	Source MAC Address	MAC Address	None	False		✓	✓	✓

Table B-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
22.10.3	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 B-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 B-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					1.0	1.1	2.0	3.0
23.9.1	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 B-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 B-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 B-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 B-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 B-1 DOCSIS Options and Version Support (continued)

Option No.	Description	Encoding	Validation	Multi-valued	DOCSIS Version			
					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 B-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 B-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 B-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 B-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 B-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 B-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 B-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 B-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 B-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	✓	✓	✓	✓

PacketCable Option Support

Table B-2 identifies the PacketCable MTA options that BAC supports.

Table B-2 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 B-3 identifies the non-secure CableHome options that BAC supports.

Table B-3 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 B-3 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 C

Mapping PacketCable DHCP Options to BAC Properties

This appendix identifies the mapping of BAC properties to the PacketCable DHCP options used for PacketCable provisioning and includes:

- Option 122 and BAC Property Comparison, page C-2
- Option 177 and BAC Property Comparison, page C-2

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 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 BAC 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 the KeyGen Tool, page 14-8, for information on changing these key configuration properties.

BAC supports both PacketCable DHCP Option 122 (as specified in RFC 3495 and 3594) and the deprecated PacketCable DHCP Option 177. BAC 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 BAC receives a DHCP request asking for both options 122 and 177, BAC 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 BAC Property Comparison

Table C-1 identifies the BAC properties as they apply to the definition of Option 122 in RFC 3495 and RFC 3594.

Table C-1 DHCP Option 122 to BAC Property Comparison

DHCP Option	Type	BAC 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 BAC; 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 BAC 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 C-2 identifies the BAC properties as they apply to the definition of Option 177.

Table C-2 DHCP Option 177 to BAC Property Comparison

Option 177	Type	BAC Property Names
177.1	ip addr	*/pktcbl/dhcp/primary*
177.2	ip addr	*/pktcbl/dhcp/secondary*

Table C-2 DHCP Option 177 to BAC Property Comparison (continued)

Option 177	Type	BAC Property Names
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*

APPENDIX D

Provisioning API Use Cases

This appendix presents a series of the most common provisioning application programming interface (API) use cases. Refer to the Cisco Broadband Access Center (BAC) 4.0 API Javadoc for more details and sample Java code segments explaining individual API calls and features.

These use cases are directly related to device provisioning, service provisioning, or both. Many administrative operations, such as managing Class of Service, DHCP Criteria, and licenses are not addressed here. We recommend that you go through the API javadoc for more details on the related API calls. You can also use the administrator user interface to perform most of these activities.

This appendix describes:

- How to Create an API Client, page D-1
- Use Cases, page D-4

How to Create an API Client

Before going through the use cases, you must familiarize yourself with how to create an API client. Use the workflow described in this section to create the API client.

Step 1 Create a connection to the Provisioning API Command Engine (PACE).

```
// The PACE connection to use throughout the example. When
// executing multiple batches in a single process, it is advisable
// to use a single PACE connection that is retrieved at the start
// of the application. When done with the connection, YOU MUST
// explicitly close the connection with the releaseConnection()
// method call.

PACEConnection connection = null;

// Connect to the Regional Distribution Unit (RDU).
//
// The parameters defined at the beginning of this class are
// used here to establish the connection. Connections are
// maintained until releaseConnection() is called. If
// multiple calls to getInstance() are called with the same
// arguments, you must still call releaseConnection() on each
// connection you received.
//
// The call can fail for one of the following reasons:
// - The hostname / port is incorrect.
// - The authentication credentials are invalid.
```

```
try
{
    connection = PACEConnectionFactory.getInstance(

         // RDU host      rduHost,
         // RDU port      rduPort,
         // User name     userName,
         // Password       password);
}
catch (PACEConnectionException e)
{
// Handle connection error:

System.out.println("Failed to establish a PACEConnection to [" +
    userName + "@" + rduHost + ":" + rduPort + "]; " +
    e.getMessage());

System.exit(1);
}
```

Step 2 Create a new instance of a batch.

```
// To perform any operations in the Provisioning API, you must
// first create a batch. As you add commands to the batch,
// nothing gets executed until you post the batch.
// Multiple batches can be started concurrently against a
// single connection to the RDU.

Batch myBatch = connection.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);
```

Step 3 Register an API command with the batch. The example featured in this step uses the *getDetails(...)* call.

```
// Use the Provisioning API to get all of the information for
// the specified MAC address. Since methods aren't actually
// executed until the batch is posted, the results are not
// returned until after post() completes. The getCommandStatus()
// followed by getData() calls must be used to access the results
// once the batch is posted.

final DeviceID modemMACAddress = DeviceID.getInstance("1,6,00:11:22:33:44:55",
    KeyType.MAC_ADDRESS);

List options = new ArrayList();
options.add(DeviceDetailsOption.INCLUDE_LEASE_INFO);

myBatch.getDetails(modemMACAddress, options);
```

Step 4 Post the batch to the Regional Distribution Unit (RDU) server.

```
// Executes the batch against the RDU. All of the
// methods are executed in the order entered.

BatchStatus bStatus = null;
```

```
try
{
// Post batch in synchronous fashion without a timeout. This method will block until
// results are returned. Other API calls are available to submit a batch with timeout
// or in asynchronous (non-blocking) fashion.

bStatus = myBatch.post();
}
catch (ProvisioningException pe)
{
System.out.println("Failed to query for modem with MAC address [" +
    modemMACAddress + "]; " + pe.getMessage());

System.exit(2);
}
```

Step 5 Check the status of the batch.

```
// Check if any errors occurred during the execution of the
// batch. Exceptions occur during post() for truly exceptional
// situations such as failure of connectivity to RDU.
// Batch errors occur for inconsistencies such as no lease
// information for a device requiring activation. Command
// errors occur when a particular method has problems, such as
// trying to add a device that already exists.

//check batchStatus and commandStatus
//for any error

CommandStatus commandStatus = null;
if (batchStatus.getCommandCount() > 0)
{
     commandStatus = batchStatus.getCommandStatus(0);
}
if (batchStatus.isError()
    || commandStatus == null
    || commandStatus.isError())
{
        System.out.println("Failed to query for modem with MAC address [" +
           modemMACAddress + "]; " + bs.getStatusCode().toString() + ", "
           + bs.getErrorMessage());
        for (int i = 0; i < bs.getCommandCount(); i++)
        {
           CommandStatus cs = bs.getCommandStatus(i);
           System.out.println("Cmd " + i + ": status code "
                   + cs.getStatusCode().toString() + ", " + cs.getErrorMessage());
        }
}
```

If there is no error, the batch call returns a successful result.

```
// Successfully queried for device.

System.out.println("Queried for DOCSIS modem with MAC address ["+
    modemMACAddress + "]");

// Display the results of the command (TreeMap is sorted). The
// data returned from the batch call is stored on a per-command
// basis. In this example, there is only one command, but if
// you had multiple commands all possibly returning results, you
// could access each result by the index of when it was added.
// The first method added is always index 0. From the status of
// each command, you can then access the accompanying data by
// using the getData() call. Since methods can return data of
```

```
// different types, you will have to cast the response to the
// type indicated in the Provisioning API documentation.

Map deviceData = (Map)bStatus.getCommandStatus(0).getData();

// Created a sorted map view

Map<String, Object> deviceDetails = new TreeMap(deviceData);
    for(String key: deviceDetails.keySet())
{
    System.out.println(" " + key + "=" + deviceDetails.get(key));
}
```

Step 6 Release the connection.

```
// Once the last batch has been executed, the connection can
// be closed to the RDU. It is important to explicitly
// close connections since it helps ensure clean shutdown of
// the Java virtual machine.

connection.releaseConnection();
```

Use Cases

This section includes these use cases:

- Self-Provisioned Modem and Computer in Fixed Standard Mode, page D-5
- Adding a New Computer in Fixed Standard Mode, page D-8
- Disabling a Subscriber, page D-10
- Preprovisioning Modems/Self-Provisioned Computers, page D-12
- Modifying an Existing Modem, page D-15
- Unregistering and Deleting a Subscriber's Devices, page D-16
- Self-Provisioning First-Time Activation in Promiscuous Mode, page D-20
- Bulk Provisioning 100 Modems in Promiscuous Mode, page D-23
- Preprovisioning First-Time Activation in Promiscuous Mode, page D-25
- Replacing an Existing Modem, page D-27
- Adding a Second Computer in Promiscuous Mode, page D-29
- Self-Provisioning First-Time Activation with NAT, page D-29
- Adding a New Computer Behind a Modem with NAT, page D-30
- Move Device to Another DHCP Scope, page D-30
- Log Device Deletions Using Events, page D-31
- Monitoring an RDU Connection Using Events, page D-32
- Logging Batch Completions Using Events, page D-33
- Getting Detailed Device Information, page D-33
- Searching Using the Device Type, page D-38
- Searching for Devices Using Vendor Prefix or Class of Service, page D-39

- Preprovisioning PacketCable eMTA, page D-40
- SNMP Cloning on PacketCable eMTA, page D-42
- Incremental Provisioning of PacketCable eMTA, page D-44
- Preprovisioning DOCSIS Modems with Dynamic Configuration Files, page D-46
- Optimistic Locking, page D-48
- Temporarily Throttling a Subscriber's Bandwidth, page D-50
- Preprovisioning CableHome WAN-MAN, page D-51
- CableHome with Firewall Configuration, page D-52
- Retrieving Device Capabilities for CableHome WAN-MAN, page D-54
- Self-Provisioning CableHome WAN-MAN, page D-55

Self-Provisioned Modem and Computer in Fixed Standard Mode

The subscriber has a computer installed in a single-dwelling unit and has purchased a DOCSIS cable modem. The computer has a web browser installed.

Desired Outcome

Use this workflow to bring a new unprovisioned DOCSIS cable modem and computer online with the appropriate level of service.

Step 1 The subscriber purchases and installs a DOCSIS cable modem at home and connects a computer to it.

Step 2 The subscriber powers on the modem and the computer, and BAC gives the modem restricted access. The computer and modem are assigned IP addresses from restricted access pools.

Step 3 The subscriber starts a web browser, and a spoofing DNS server points the browser to a service provider's registration server (for example, an OSS user interface or a mediator).

Step 4 The subscriber uses the service provider's user interface to complete the steps required for registration, including selecting Class of Service.

Step 5 The service provider's user interface passes the subscriber's information, such as the selected Class of Service and computer IP address, to BAC, which then registers the subscriber's modem and computer.

```
// First we query the computer's information to find the
// modem's MAC Address. We use the computer IP Address (the web browser
// received this when the subscriber opened the service provider's
// web interface

PACEConnection connection = PACEConnectionFactory.getInstance(
            "localhost", 49187, "admin", "changeme");

// NO_ACTIVATION is the activation mode because this is a query.
// NO_CONFIRMATION is the confirmation mode because we are not
// attempting to reset the device.
// NO_PUBLISHING is the publishing mode because we are not attempting
// to publish to external database.

Batch batch = connection.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,
```

```
    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);

// register getAllForIPAddress to the batch

batch.getAllForIPAddress("10.0.14.38");

BatchStatus batchStatus = null;

// post the batch to RDU server

try
{
batchStatus = batch.post();
}
catch(ProvisioningException e)
{
e.printStackTrace();
}
// Get the LeaseResults object after posting a batch.

CommandStatus commandStatus = batchStatus.getCommandStatus(0);

LeaseResults computerLease = (LeaseResults)commandStatus.getData();

// Derive the modem MAC address from computer's network
// information. The "1,6" is a standard prefix for an Ethernet
// device. The fully qualify MAC Address is required by BACC

StringBuffer modemMACAddress = new StringBuffer();
modemMACAddress.append("1,6,");
modemMACAddress.append(computerLease.getSingleLease().get("relay-agent-remote-id"));

// Create MacAddress object from the string

MACAddress modemMACAddressObject = new MACAddress(modemMACAddress.toString());

List<DeviceID> modemDeviceIDList = new ArrayList<DeviceID>();
    modemDeviceIDList.add(modemMACAddressObject);

// Create a new batch to add modem device

batch = connection.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);

// Register add API to the batch
```

```
    batch.add(DeviceType.DOCSIS, modemDeviceIDList,
        null, null, "0123-45-6789", "silver", "provisioned-cm", null);

    // post the batch to RDU server

    // Derive computer MAC address from computer's network information.

    String computerMACAddress =
        (String)computerLease.getSingleLease().get(DeviceDetailsKeys.MAC_ADDRESS);

    // Create a map for computer property.

    Map<String, Object> properties = new HashMap<String, Object>();
    properties.put(IPDeviceKeys.MUST_BE_BEHIND_DEVICE, modemMACAddress.toString());

    List<DeviceID> compDeviceIDList = new ArrayList<DeviceID>();
    MACAddress computerMACAddressObject = new MACAddress(computerMACAddress);
    compDeviceIDList.add(computerMACAddressObject);

    // Register add API to the batch

    batch.add(DeviceType.COMPUTER, compDeviceIDList,
        null, null, "0123-45-6789", null, "provisioned-cpe", properties);
    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
```

Step 6 The provisioning client calls *performOperation(DeviceOperation deviceOperation, DeviceID deviceID, Map<String,Object> parameters* to reboot the modem and gives the modem provisioned access.

```
    // Reset the computer
    // Create a new batch

    batch = connection.newBatch(

        // No reset

        ActivationMode.AUTOMATIC,

        // No need to confirm activation

        ConfirmationMode.NO_CONFIRMATION);

    // Register performOperation command to the batch

    batch.performOperation(DeviceOperation.RESET, modemMACAddressObject, null);

    // Post the batch to RDU server

    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
        }
    }
}
```

Step 7 The user interface prompts the subscriber to reboot the computer.

After rebooting, the computer receives a new IP address, and both cable modem and computer are now provisioned devices. The computer has access to the Internet through the service provider's network.

Adding a New Computer in Fixed Standard Mode

A multiple system operator (MSO) lets a subscriber have two computers behind a cable modem. The subscriber has one computer already registered and then brings home a laptop from work and wants access. The subscriber installs a hub and connects the laptop to it.

Desired Outcome

Use this workflow to bring a new unprovisioned computer online with a previously provisioned cable modem so that the new computer has the appropriate level of service.

Step 1 The subscriber powers on the new computer and BAC gives it restricted access.

Step 2 The subscriber starts a web browser on the new computer and a spoofing DNS server points it to the service provider's registration server (for example, an OSS user interface or a mediator).

Step 3 The subscriber uses the service provider's user interface to complete the steps required to add a new computer.

Step 4 The service provider's user interface passes the subscriber's information, such as the selected Class of Service and computer IP address, to BAC, which then registers the subscriber's modem and computer.

```
// First we query the computer's information to find the
// modem's MAC Address. We use the computer IP address (the web browser
// received this when the subscriber opened the service provider's
// web interface.
PACEConnection connection = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "changeme");

// NO_ACTIVATION is the activation mode because this is a query
// NO_CONFIRMATION is the confirmation mode because we are not
// attempting to reset the device
// NO_PUBLISHING is the publishing mode because we are not attempting
// to publish to external database.

Batch batch = connection.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);

// register getAllForIPAddress to the batch

batch.getAllForIPAddress("10.0.14.39");
```

```
BatchStatus batchStatus = null;

// post the batch to RDU server

try
{
    batchStatus = batch.post();
}
catch(ProvisioningException e)
{
    e.printStackTrace();
}
// Get the LeaseResults object after posting a batch.

CommandStatus commandStatus = batchStatus.getCommandStatus(0);

LeaseResults computerLease = (LeaseResults)commandStatus.getData();

// derive the modem MAC address from computer's network
// information. The "1,6" is a standard prefix for an Ethernet
// device. The fully qualify MAC Address is required by BACC

StringBuffer modemMACAddress = new StringBuffer();
modemMACAddress.append("1,6,");
modemMACAddress.append(computerLease.getSingleLease().get("relay-agent-remote-id"));

// derive computer MAC address from computer's network information.

String computerMACAddress =
    (String)computerLease.getSingleLease().get(DeviceDetailsKeys.MAC_ADDRESS);

//Create a map for computer property.

Map<String, Object> properties = new HashMap<String, Object>();

// setting IPDeviceKeys.MUST_BE_BEHIND_DEVICE on the computer ensures
// that when the computer boots, it will only receive its provisioned
// access when it is behind the given device. If it is not behind
// the given device, it will receive default access (unprovisioned)
// and hence fixed mode.

properties.put(IPDeviceKeys.MUST_BE_BEHIND_DEVICE, modemMACAddress);

// the IPDeviceKeys.MUST_BE_IN_PROV_GROUP ensures that the computer
// will receive its provisioned access only when it is brought up in
// the specified provisioning group. This prevents the computer
// (and/or) the modem from moving from one locality to another
// locality.

properties.put(IPDeviceKeys.MUST_BE_IN_PROV_GROUP, "bostonProvGroup");

List<DeviceID> compDeviceIDList = new ArrayList<DeviceID>();
MACAddress computerMACAddressObject = new MACAddress(computerMACAddress);
compDeviceIDList.add(computerMACAddressObject);

batch = connection.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,
```

```
        // No publishing to external database

        PublishingMode.NO_PUBLISHING);

    // register add API to the batch

    batch.add(
        DeviceType.COMPUTER,    // deviceType: Computer
        compDeviceIDList,       // compDeviceIDList: the list of DeviceIDs derived from
                                // computerLease
        null,                   // hostName: not used in this example
        null,                   // domainName: not used in this example
        "0123-45-6789",         // ownerName
        null,                   // class of service: get the default COS
        "provisionedCPE",       // dhpcCriteria: Network Registrar uses this to
                                  // select a modem lease granting provisioned IP address
        properties                // device properties
    );


    // post the batch to RDU server

    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
        }
    }
}
```

Step 5 The user interface prompts the subscriber to reboot the new computer so that BAC can give the computer its registered service level.

The computer is now a provisioned device with access to the appropriate level of service.

Disabling a Subscriber

A service provider needs to disable a subscriber from accessing the Internet due to recurring nonpayment.

Desired Outcome

Use this workflow to disable an operational cable modem and computer, so that the devices temporarily restrict Internet access for the user. Additionally, this use case can redirect the user's browser to a special page that could announce:

```
You haven't paid your bill so your Internet access has been disabled.
```

Step 1 The service provider's application uses a provisioning client program to request a list of all of the subscriber's devices from BAC.

Step 2 The service provider's application then uses a provisioning client to individually disable or restrict each of the subscriber's devices.

```
PACEConnection conn = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "admin123");

//get all for owner ID

Batch batch = conn.newBatch();
batch.getAllForOwnerID("0123-45-6789");

BatchStatus batchStatus = null;
try
{
    batchStatus = batch.post();
}
catch(Exception e)
{
    e.printStackTrace();
}
CommandStatus commandStatus = batchStatus.getCommandStatus(0);


//batch success without error, retrieve the result

RecordSearchResults rcSearchResult = (RecordSearchResults)commandStatus.getData();
List<RecordData> resultList = rcSearchResult.getRecordData();

if (resultList != null)
{
// getting the data

for (int i=0; i<resultList.size(); i++)
{
    RecordData rd = resultList.get(i);
    Map<String, Object> detailMap = rd.getDetails();

    //get the deviceType from the detail map

    String deviceType =
        (String)detailMap.get(DeviceDetailsKeys.DEVICE_TYPE);

    Key primaryKey = rd.getPrimaryKey();

    //only interest in DOCSIS
    if (DeviceType.getDeviceType(deviceType)
        .equals(DeviceType.DOCSIS))
    {

    //change COS

    batch = conn.newBatch();
    batch.changeClassOfService((DeviceID)primaryKey, "DisabledCOS");

    //change DHCPCriteria

    batch.changeDHCPCriteria((DeviceID)primaryKey, "DisabledDHCPCriteria");

    batchStatus = null;
    try
```

```
            {
                 batchStatus = batch.post();
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }

        }
        //disable computer

        else if (DeviceType.getDeviceType(deviceType)
            .equals(DeviceType.COMPUTER))
        {
            //change DHCPCriteria

            batch = conn.newBatch();
            batch.changeClassOfService((DeviceID)primaryKey,
                "DisabledComputerCOS");
            batch.changeDHCPCriteria((DeviceID)primaryKey,
                "DisabledComputerDHCPCriteria");

            batchStatus = null;
            try
            {
                batchStatus = batch.post();
            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
        }
    }
```

Note You may need to consider the impact on the CPE behind the modem when defining the characteristics of DisabledCOS and resetting the modem. This is especially important if you have voice end points behind the modem, because disrupting the cable modem might affect the telephone conversation in progress at that time.

The subscriber is now disabled.

Preprovisioning Modems/Self-Provisioned Computers

A new subscriber contacts the service provider and requests service. The subscriber has a computer installed in a single-dwelling unit. The service provider preprovisions all its cable modems in bulk.

Desired Outcome

Use this workflow to bring a preprovisioned cable modem, and an unprovisioned computer, online in the roaming standard mode. This must be done so that both devices have the appropriate level of service and are registered.

Step 1 The service provider chooses a subscriber username and password for the billing system.

Step 2 The service provider selects services that the subscriber can access.

Step 3 The service provider's field technician installs the physical cable to the new subscriber's house and installs the preprovisioned device, connecting it to the subscriber's computer.

Step 4 The technician turns on the modem and BAC gives it a provisioned IP address.

Step 5 The technician turns on the computer and BAC gives it a private IP address.

Step 6 The technician starts a browser application on the computer and points the browser to the service provider's user interface.

Step 7 The technician accesses the service provider's user interface to complete the steps required for registering the computer behind the provisioned cable modem.

```
// First we query the computer's information to find the
// modem's MAC Address. We use the computer IP address (the web browser
// received this when the subscriber opened the service provider's
// web interface

PACEConnection connection = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "changeme");

// NO_ACTIVATION is the activation mode because this is a query
// NO_CONFIRMATION is the confirmation mode because we are not
// attempting to reset the device
// NO_PUBLISHING is the publishing mode because we are not attempting
// to publish to external database.

Batch batch = connection.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);

// register getAllForIPAddress to the batch

batch.getAllForIPAddress("10.0.14.38");

BatchStatus batchStatus = null;

// post the batch to RDU server

try
{
    batchStatus = batch.post();
}
catch(ProvisioningException e)
{
    e.printStackTrace();
}

// Get the LeaseResults object after posting a batch.

CommandStatus commandStatus = batchStatus.getCommandStatus(0);

LeaseResults computerLease = (LeaseResults)commandStatus.getData();

// derive computer MAC address from computer's network information.
```

```
    String computerMACAddress =
        (String)computerLease.getSingleLease().get(DeviceDetailsKeys.MAC_ADDRESS);


        List<DeviceID> compDeviceIDList = new ArrayList<DeviceID>();
        MACAddress computerMACAddressObject = new MACAddress(computerMACAddress);
        compDeviceIDList.add(computerMACAddressObject);


    // NO_ACTIVATION will generate new configuration for the computer,
    // however it will not attempt to reset it.
    // NO_CONFIRMATION is the confirmation mode because we are not
    // attempting to reset the computer because this cannot be done.

    batch = connection.newBatch(
        // No reset

        ActivationMode.NO_ACTIVATION,

        // No need to confirm activation

        ConfirmationMode.NO_CONFIRMATION,

        // No publishing to external database

        PublishingMode.NO_PUBLISHING);

    // register add API to the batch

    batch.add(
        DeviceType.COMPUTER,  // deviceType: Computer
        compDeviceIDList,     // compDeviceIDList: the list of DeviceIDs derived from
                              // computerLease
        null,                 // hostName: not used in this example
        null,                 // domainName: not used in this example
        "0123-45-6789",       // ownerName
        null,                 // class of service: get the default COS
        "provisionedCPE",     // dhpcCriteria: Network Registrar uses this to
                              // select a modem lease granting provisioned IP address
        null                  // properties: not used
    );

    // post the batch to RDU server

    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
}
```

Note The *IPDeviceKeys.MUST_BE_BEHIND_DEVICE* property is not set on the computer and this allows roaming from behind one cable modem to another.

Step 8 The technician restarts the computer and the computer receives a new provisioned IP address.

The cable modem and the computer are now both provisioned devices. The computer has access to the Internet through the service provider's network.

Modifying an Existing Modem

A service provider's subscriber currently has a level of service known as **Silver** and has decided to upgrade to **Gold** service. The subscriber has a computer installed at home.

Note The intent of this use case is to show how to modify a device. You can apply this example to devices provisioned in modes other than roaming standard.

Desired Outcome

Use this workflow to modify an existing modem's Class of Service and pass that change of service to the service provider's external systems.

Step 1 The subscriber phones the service provider and requests to have service upgraded. The service provider uses its user interface to change the Class of Service from **Silver** to **Gold**.

Step 2 The service provider's application makes these API calls in BAC:

```
// NO_ACTIVATION is the activation mode because this is a query
// NO_CONFIRMATION is the confirmation mode because we are not
// attempting to reset the device
// NO_PUBLISHING is the publishing mode because we are not attempting
// to publish to external database.

Batch batch = connection.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);

// replace changeClassOfService to this. Make sure the comment
// on top of this line is still there.

batch.changeClassOfService(new MACAddress("1,6,00:11:22:33:44:55")

    // the MACAddress object

    , "Gold");


// post the batch to the RDU

BatchStatus batchStatus = null;
try
{
```

```
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
}
```

The subscriber can now access the service provider's network with the *Gold* service.

Unregistering and Deleting a Subscriber's Devices

A service provider needs to delete a subscriber who has discontinued service.

Desired Outcome

Use this workflow to permanently remove all the subscriber's devices from the service provider's network.

Step 1 The service provider's user interface discontinues service to the subscriber.

Step 2 This step describes how to unregister a subscriber's device and delete a subscriber's device. Deleting a device is optional because some service providers prefer to keep the cable modem in the database unless it is broken. Note however that if you unregister a device using Step 2-a, you cannot delete the device using Step 2-b.

a. To unregister a device, the service provider's application uses a provisioning client program to request a list of all the subscriber's devices from BAC, and unregisters and resets each device so that it is brought down to the default (unprovisioned) service level.

Note If the device specified as the parameter to the "unregister" API is already in unregistered state then the status code from the API call will be set to `CommandStatusCodes.CMD_ERROR_DEVICE_UNRE- GISTER_UNREGISTERED_ERROR`. This is normal/expected behavior.

```
// MSO admin UI calls the provisioning API to get a list of
// all the subscriber's devices.

// Create a new connection

PACEConnection conn = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "admin123");


// NO_ACTIVATION is the activation mode because this is a query
// NO_CONFIRMATION is the confirmation mode because we are not
// attempting to reset the device
// NO_PUBLISHING is the publishing mode because we are not attempting
// to publish to external database.

Batch batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,
```

```
    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);

batch.getAllForOwnerID("0123-45-6789"

//  query all the devices for this account number
);

BatchStatus batchStatus = null;
try
{
    batchStatus = batch.post();
}
catch(Exception e)
{
    e.printStackTrace();
}
CommandStatus commandStatus = batchStatus.getCommandStatus(0);


//batch success without error, retrieve the result

RecordSearchResults rcSearchResult = (RecordSearchResults)commandStatus.getData();
List<RecordData> resultList = rcSearchResult.getRecordData();

//  We need to unregister all the devices behind each modem(s) or else the
//  unregister call for that modem will fail.

if (resultList != null)
{
  //Unregister the COMPUTER
  for (int i=0; i<resultList.size(); i++)
{
    RecordData rd = resultList.get(i);
    Map<String, Object> detailMap = rd.getDetails();

    //get the deviceType from the detail map

    String deviceType = (String)detailMap.get(DeviceDetailsKeys.DEVICE_TYPE);


    //only interest in DOCSIS

    if (DeviceType.getDeviceType(deviceType) .equals(DeviceType.COMPUTER))
    {
        Key primaryKey = rd.getPrimaryKey();

        batch = conn.newBatch();
        batch.unregister((DeviceID)primaryKey);

        batchStatus = null;
        try
        {
           batchStatus = batch.post();
        }
        catch(ProvisioningException e)
        {
           e.printStackTrace();
```

```
            }
         }
    }
}
    //  for each modem in the retrieved list:

    for (int i=0; i<resultList.size(); i++)
    {
    RecordData rd = resultList.get(i);
    Map<String, Object> detailMap = rd.getDetails();

    //get the deviceType from the detail map

    String deviceType = (String)detailMap.get(DeviceDetailsKeys.DEVICE_TYPE);


    //only interest in DOCSIS

    if (DeviceType.getDeviceType(deviceType) .equals(DeviceType.DOCSIS))
    {
        Key primaryKey = rd.getPrimaryKey();
        batch = conn.newBatch();
        batch.unregister((DeviceID)primaryKey);
        batchStatus = null;
        try
        {
            batchStatus = batch.post();
        }
        catch(ProvisioningException e)
        {
            e.printStackTrace();
        }
    }
}
```

b. To delete a device, the service provider's application uses a provisioning client program to delete each of the subscriber's remaining devices individually from the database.

```
// Create a new connection

PACEConnection conn =
    PACEConnectionFactory.getInstance("localhost", 49187, "admin", "admin123");


// NO_ACTIVATION is the activation mode because this is a query
// NO_CONFIRMATION is the confirmation mode because we are not
// attempting to reset the device
// NO_PUBLISHING is the publishing mode because we are not attempting
// to publish to external database.

Batch batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publisining to external database

    PublishingMode.NO_PUBLISHING);
```

```
        batch.getAllForOwnerID("0123-45-6789"   //  query all the devices for this account
                                                // number
        );

        BatchStatus batchStatus = null;
        try
        {
            batchStatus = batch.post();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }

        CommandStatus commandStatus = batchStatus.getCommandStatus(0);


        //batch success without error, retrieve the result

        RecordSearchResults rcSearchResult = (RecordSearchResults)commandStatus.getData();
        List<RecordData> resultList = rcSearchResult.getRecordData();

        if (resultList != null)
        {

        //  for each modem in the retrieved list, delete it

        for (int i=0; i<resultList.size(); i++)
        {
            RecordData rd = resultList.get(i);
            Map<String, Object> detailMap = rd.getDetails();

            //get the deviceType from the detail map

            String deviceType = (String)detailMap.get(DeviceDetailsKeys.DEVICE_TYPE);


            //only interest in DOCSIS

            if (DeviceType.getDeviceType(deviceType) .equals(DeviceType.DOCSIS))
            {
                 Key primaryKey = rd.getPrimaryKey();

                 //change COS

                 batch = conn.newBatch();
                 batch.delete((DeviceID)primaryKey, true);

                 batchStatus = null;
                 try
                {
                     batchStatus = batch.post();
                }
                 catch(ProvisioningException e)
                {
                     e.printStackTrace();
                 }
           }
    }
}
```

Self-Provisioning First-Time Activation in Promiscuous Mode

The subscriber has a computer (with a browser application) installed in a single-dwelling unit and has purchased a DOCSIS cable modem.

Desired Outcome

Use this workflow to bring a new unprovisioned DOCSIS cable modem and computer online with the appropriate level of service.

Step 1 The subscriber purchases a DOCSIS cable modem and installs it at home.

Step 2 The subscriber powers on the modem, and BAC gives it restricted access.

Step 3 The subscriber starts a browser application on the computer and a spoofing DNS server points the browser to the service provider's registration server (for example, an OSS user interface or a mediator).

Step 4 The subscriber uses the service provider's user interface to complete the steps required for registration, including selecting a Class of Service.

The service provider's user interface passes the subscriber's information to BAC, including the selected Class of Service and computer IP address. The subscriber's cable modem and computer are then registered with BAC.

Step 5 The user interface prompts the subscriber to reboot the computer.

```
// Create a new connection
PACEConnection conn = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "admin123");


Batch batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);


// NO_ACTIVATION is the activation mode because this is a
// query. NO_CONFIRMATION is the confirmation mode because
// we are not attempting to reset the device.
// First we query the computer's information to find the
// modem's MAC address.
// We use the computer's IP address (the web browser
// received this when the subscriber opened the service
// provider's web interface).
// We also assume that "bostonProvGroup"
// is the provisioning group used in that locality.

List<String> provGroupList = new ArrayList<String>();

provGroupList.add("bostonProvGroup");
```

```
batch.getAllForIPAddress("10.0.14.38",

    // ipAddress: restricted access computer lease
    provGroupList
    // provGroups: List containing provgroup
    );

BatchStatus batchStatus = null;

// post the batch to RDU server

try
{
    batchStatus = batch.post();
}
catch(ProvisioningException e)
{
    e.printStackTrace();
}

// Get the LeaseResults object after posting a batch.

CommandStatus commandStatus = batchStatus.getCommandStatus(0);

LeaseResults computerLease = (LeaseResults)commandStatus.getData();

// Derive the modem MAC address from the computer's network
// information. The 1,6, is a standard prefix for an Ethernet
// device. The fully qualified MAC address is required by BACC

StringBuffer modemMACAddress = new StringBuffer();
modemMACAddress.append("1,6,");
modemMACAddress.append(computerLease.getSingleLease().get("relay-agent-remote-id"));


//create MacAddress object from the string

MACAddress modemMACAddressObject = new MACAddress(modemMACAddress.toString());

List<DeviceID> modemDeviceIDList = new ArrayList<DeviceID>();
modemDeviceIDList.add(modemMACAddressObject);


// NO_ACTIVATION is the activation mode because this is a query
// NO_CONFIRMATION is the confirmation mode because we are not
// attempting to reset the device
// NO_PUBLISHING is the publishing mode because we are not attempting
// to publish to external database.

batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);

Map<String, Object> properties = new HashMap<String, Object>();
```

```
// Set the property PolicyKeys.COMPUTER_PROMISCUOUS_MODE_ENABLED
// to enable promiscuous mode on modem

properties.put(PolicyKeys.COMPUTER_PROMISCUOUS_MODE_ENABLED, Boolean.TRUE);

properties.put(PolicyKeys.COMPUTER_DHCP_CRITERIA, "provisionedCPE");


// enable promiscuous mode by changing the technology default

batch.changeDefaults(DeviceType.DOCSIS,
    properties, null);


// post the batch to RDU server

try
{
    batchStatus = batch.post();
}
catch(ProvisioningException e)
{
    e.printStackTrace();
}
batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);


batch.add(
    DeviceType.DOCSIS,    // deviceType: DOCSIS
    modemDeviceIDList,    // macAddress: derived from computer lease
    null,                 // hostName: not used in this example
    null,                 // domainName: not used in this example
    "0123-45-6789",       // ownerID: here, account number from billing system
    "Silver",             // ClassOfService
    "provisionedCM",      // DHCP Criteria: Network Registrar uses this to
                          // select a modem lease granting provisioned IP address
    null                  // properties:
);
```

Step 6 The provisioning client calls *performOperation(…)* to reboot the modem and gives the modem provisioned access.

```
// Reset the computer
// create a new batch
batch = conn.newBatch(

    // No reset

    ActivationMode.AUTOMATIC,

    // No need to confirm activation
```

```
        ConfirmationMode.NO_CONFIRMATION);

    // register performOperation command to the batch

    batch.performOperation(DeviceOperation.RESET,
        modemMACAddressObject, null);

    // post the batch to RDU server

    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
}
```

Step 7 When the computer is rebooted, it receives a new IP address.

The cable modem is now a provisioned device. The computer is not registered with BAC, but it gains access to the Internet through the service provider's network. Computers that are online behind promiscuous modems are still available using the provisioning API.

Bulk Provisioning 100 Modems in Promiscuous Mode

A service provider wants to preprovision 100 cable modems for distribution by a customer service representative at a service kiosk.

Desired Outcome

Use this workflow to distribute modem data for all modems to new subscribers. The customer service representative has a list of modems available for assignment.

Step 1 The cable modem's MAC address data for new or recycled cable modems is collected into a list at the service provider's loading dock.

Step 2 Modems that are assigned to a particular kiosk are bulk-loaded into BAC and are flagged with the identifier for that kiosk.

Step 3 When the modems are distributed to new subscribers at the kiosk, the customer service representative enters new service parameters, and changes the Owner ID field on the modem to reflect the new subscriber's account number.

```
// Create a new connection
PACEConnection conn = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "admin123");


Batch batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation
```

```
    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);


// The activation mode for this batch should be NO_ACTIVATION.
// NO_ACTIVATION should be used in this situation because no
// network information exists for the devices because they
// have not booted yet. A configuration can't be generated if no
// network information is present. And because the devices
// have not booted, they are not online and therefore cannot
// be reset. NO_CONFIRMATION is the confirmation mode because
// we are not attempting to reset the devices.
// Create a Map for the properties of the modem

Map properties;

// Set the property PolicyKeys.COMPUTER_PROMISCUOUS_MODE_ENABLED to
// enable promiscuous mode on modem.
// This could be done at a system level if promiscuous mode
// is your default provisioning mode.

properties.put(PolicyKeys.COMPUTER_PROMISCUOUS_MODE_ENABLED, Boolean.TRUE);

// The PolicyKeys.CPE_DHCP_CRITERIA is used to specify the DHCP
// Criteria to be used while selecting IP address scopes for
// CPE behind this modem in the promiscuous mode.

properties.put(PolicyKeys.COMPUTER_DHCP_CRITERIA, "provisionedCPE");

// enable promiscuous mode by changing the technology default

batch.changeDefaults(DeviceType.DOCSIS,properties, null);

BatchStatus batchStatus = null;

// post the batch to RDU server

try
{
    batchStatus = batch.post();
}
catch(ProvisioningException e)
{
    e.printStackTrace();
        }

// for each modem MAC-address in list:

ModemLoop:
{
    batch = conn.newBatch(

       // No reset

       ActivationMode.NO_ACTIVATION,

       // No need to confirm activation

       ConfirmationMode.NO_CONFIRMATION,
```

```
            // No publishing to external database

            PublishingMode.NO_PUBLISHING);


    batch.add(
        DeviceType.DOCSIS,   // deviceType: DOCSIS
        modemMACAddressList, // modemMACAddressList: the list of deviceID
        null,                // hostName: not used in this example
        null,                // domainName: not used in this example
        "0123-45-6789",      // ownerID: here, account number from billing system
        "Silver",            // ClassOfService
        "provisionedCM",     // DHCP Criteria: Network Registrar uses this to
                             // select a modem lease granting provisioned IP address
        properties           // properties:
    );

    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
    // end ModemLoop.
}
```

Preprovisioning First-Time Activation in Promiscuous Mode

A new subscriber contacts the service provider and requests service. The subscriber has a computer installed in a single-dwelling unit.

Desired Outcome

Use this workflow to bring a new unprovisioned cable modem and computer online with the appropriate level of service.

Step 1 The service provider chooses a subscriber username and password for the billing system.

Step 2 The service provider selects the services that the subscriber can access.

Step 3 The service provider registers the device using its own user interface.

Step 4 The service provider's user interface passes information, such as the modem's MAC address and the Class of Service, to BAC. Additionally, the modem gets a CPE DHCP Criteria setting that lets Network Registrar select a provisioned address for any computers to be connected behind the modem. The new modem is then registered with BAC.

Step 5 The service provider's field technician installs the physical cable to the new subscriber's house and installs the preprovisioned device, connecting it to the subscriber's computer.

```
//  MSO admin UI calls the provisioning API to pre-provision
//  an HSD modem.

// Create a new connection
PACEConnection conn = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "admin123");
```

```
Batch batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);


// The activation mode for this batch should be NO_ACTIVATION.
// NO_ACTIVATION should be used in this situation because no
// network information exists for the modem because it has not
// booted. A configuration cannot be generated if no network
// information is present. And because the modem has not booted,
// it is not online and therefore cannot be reset.
// NO_CONFIRMATION is the confirmation mode because we are not
// attempting to reset the modem.
// Create a map for the properties of the modem.


Map<String, Object> properties = new HashMap<String, Object>();

// Set the property PolicyKeys.COMPUTER_PROMISCUOUS_MODE_ENABLED
// to enable promiscuous mode on modem

properties.put(PolicyKeys.COMPUTER_PROMISCUOUS_MODE_ENABLED, Boolean.TRUE);

properties.put(PolicyKeys.COMPUTER_DHCP_CRITERIA, "provisionedCPE");


// enable promiscuous mode by changing the technology default

batch.changeDefaults(DeviceType.DOCSIS, properties, null);

BatchStatus batchStatus = null;

// post the batch to RDU server

try
{
    batchStatus = batch.post();
}
catch(ProvisioningException e)
{
    e.printStackTrace();
}
batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,
```

```
        // No publishing to external database

        PublishingMode.NO_PUBLISHING);


    MACAddress macAddressObject = new MACAddress("1,6,00:11:22:33:44:55");
    List<DeviceID> modemDeviceIDList = new ArrayList<DeviceID>();
    modemDeviceIDList.add(macAddressObject);

    batch.add(
        DeviceType.DOCSIS,       // deviceType: DOCSIS
        modemDeviceIDList,       // macAddress: derived from computer lease
        null,                    // hostName: not used in this example
        null,                    // domainName: not used in this example
        "0123-45-6789",      // ownerID: here, account number from billing system
        "Silver",            // ClassOfService
        "provisionedCM",     // DHCP Criteria: Network Registrar uses this to
                             // select a modem lease granting provisioned IP address
        null                 // properties:
    );


    // post the batch to RDU server

    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
}
```

Step 6 The technician powers on the cable modem and BAC gives it provisioned access.

Step 7 The technician powers on the computer and BAC gives it provisioned access.

The cable modem and the computer are now both provisioned devices. The computer has access to the Internet through the service provider's network.

Replacing an Existing Modem

A service provider wants to replace a broken modem.

Note If the computer has the option restricting roaming from one modem to another, and the modem is replaced, the computer's MAC address for the modem must also be changed.

Desired Outcome

Use this workflow to physically replace an existing cable modem with a new modem without changing the level of service provided to the subscriber.

Step 1 The service provider changes the MAC address of the existing modem to that of the new modem.

```
    // Create a new connection
    PACEConnection conn = PACEConnectionFactory.getInstance(
        "localhost", 49187, "admin", "admin123");

    batch = conn.newBatch(

        // No reset

        ActivationMode.NO_ACTIVATION,

        // No need to confirm activation

        ConfirmationMode.NO_CONFIRMATION,

        // No publishing to external database

        PublishingMode.NO_PUBLISHING);


    MACAddress macAddressObject = new MACAddress("1,6,00:11:22:33:44:55");
    List<DeviceID> modemDeviceIDList = new ArrayList<DeviceID>();
    modemDeviceIDList.add(macAddressObject);

    //  old macAddress: unique identifier for the old modem
    MACAddress oldMacAddress = new MACAddress("1,6,00:11:22:33:44:55");

    // new macAddress: unique identifier for the new modem
    MACAddress newMacAddress = new MACAddress("1,6,00:11:22:33:44:66");
    List<DeviceID> newDeviceIDs = new ArrayList<DeviceID>();
    newDeviceIDs.add(newMacAddress);

    batch.changeDeviceID(oldMacAddress, newDeviceIDs);

    // post the batch to RDU server

    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
}
```

Step 2 The service provider replaces the cable modem and turns it on.

Step 3 The computer must also be turned on.

The cable modem is now a fully provisioned device with the appropriate level of service, as is the computer behind the cable modem.

Adding a Second Computer in Promiscuous Mode

A subscriber wants to connect a second computer behind an installed cable modem. This case does not require calls to the provisioning API.

Desired Outcome

Use this workflow to ensure that the subscriber's selected service permits the connection of multiple sets of CPE, and that the subscriber has network access from both connected computers.

Step 1 The subscriber connects a second computer behind the cable modem.

Step 2 The subscriber turns on the computer.

If the subscriber's selected service permits connecting multiple sets of CPE, BAC gives the second computer access to the Internet.

Self-Provisioning First-Time Activation with NAT

A university has purchased a DOCSIS cable modem with network address translation (NAT) and DHCP capability. The five occupants of the unit each have a computer installed with a browser application.

Desired Outcome

Use this workflow to bring a new unprovisioned cable modem (with NAT) and the computers behind it online with the appropriate level of service.

Step 1 The subscriber purchases a cable modem with NAT and DHCP capability and installs it in a multiple-dwelling unit.

Step 2 The subscriber turns on the modem and BAC gives it restricted access.

Step 3 The subscriber connects a laptop computer to the cable modem, and the DHCP server in the modem provides an IP address to the laptop.

Step 4 The subscriber starts a browser application on the computer and a spoofing DNS server points the browser to the service provider's registration server (for example, an OSS user interface or a mediator).

Step 5 The subscriber uses the service provider's user interface to complete the steps required for cable modem registration of the modem. The registration user interface detects that the modem is using NAT and registers the modem, making sure that the modem gets a Class of Service that is compatible with NAT. For details, see Self-Provisioned Modem and Computer in Fixed Standard Mode, page D-5.

Note Certain cable modems with NAT may require you to reboot the computer to get the new Class of Service settings. If the cable modem and NAT device are separate devices, the NAT device must also be registered similarly to registering a computer.

Adding a New Computer Behind a Modem with NAT

The landlord of an apartment building has four tenants sharing a modem and accessing the service provider's network. The landlord wants to provide Internet access to a new tenant, sharing the building's modem. The modem has NAT and DHCP capability. The new tenant has a computer connected to the modem.

Note This case does not require calls to the provisioning API.

Desired Outcome

Use this workflow to bring a new unprovisioned computer online with a previously provisioned cable modem so that the new computer has the appropriate level of service.

Step 1 The subscriber turns on the computer.

Step 2 The computer is now a provisioned device with access to the appropriate level of service.

The provisioned NAT modem hides the computers behind it from the network.

Move Device to Another DHCP Scope

A service provider is renumbering its network causing a registered cable modem to require an IP address from a different Network Registrar scope.

Desired Outcome

A provisioning client changes the DHCP Criteria, and the cable modem receives an IP address from the corresponding DHCP scope.

Step 1 Change the DOCSIS modem's DHCP Criteria to "newmodemCriteria".

```
// Create a new connection
PACEConnection conn = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "admin123");

Batch batch = conn.newBatch(

    // No reset

    ActivationMode.AUTOMATIC,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);

// AUTOMATIC is the Activation mode because we are attempting
// to reset the modem so that a phone line is disabled
// NO_CONFIRMATION is the Confirmation mode because we don't
// want the batch to fail if we can't reset the modem.
```

```
    // This use case assumes that the DOCSIS modem has been
    // previously added to the database

    batch.changeDHCPCriteria(
        new MACAddress("1,6,ff:00:ee:11:dd:22"),  // Modem's MAC address or FQDN
              "newmodemCriteria"
    );

    // post the batch to RDU server

    BatchStatus batchStatus = null;
    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
}
```

Step 2 The modem gets an IP address from the scope targeted by "newmodemCriteria."

Log Device Deletions Using Events

A service provider has multiple provisioning clients and wants to log device deletions.

Desired Outcome

When any provisioning client deletes a device, the provisioning client logs an event in one place.

Step 1 Create a listener for the device deletion event. This class must extend the *DeviceAdapter* abstract class or, alternatively, implement the *DeviceListener* interface. This class must also override the *deletedDevice(DeviceEvent ev)* method in order to log the event.

```
public DeviceDeletionLogger
        extends DeviceAdapter

        //Extend the DeviceAdapter class.
{
    public void deletedDevice(DeviceEvent ev)

    //Override deletedDevice.
    {
        logDeviceDeletion(ev.getDeviceID());

        //Log the deletion.
    }
}
```

Step 2 Register the listener and the qualifier for the events using the *PACEConnection* interface.

```
DeviceDeletionLogger deviceDeletionLogger =
    new DeviceDeletionLogger();

        // Modem's MAC address or FQDN "newmodemCriteria"

DeviceEventQualifier qualifier = new DeviceEventQualifier();
```

```
// We are interested only in device deletion.

qualifier.setDeletedDevice ();

// Add device listener using PACEConnection

connection.addDeviceListener(deviceDeletionLogger, qualifier
);
```

Step 3 When a device is deleted from the system, the event is generated, and the listener is notified.

Monitoring an RDU Connection Using Events

A service provider is running a single provisioning client and wants notification if the connection between the provisioning client and the RDU breaks.

Desired Outcome

Use this workflow to have the event interface notify the service provider if the connection breaks.

Step 1 Create a listener for the messaging event. This class must extend the *MessagingAdapter* abstract class or, alternatively, implement the *MessagingListener* interface. This class must override the *connectionStopped(MessagingEvent ev)* method.

```
// Extend the service provider's Java program using the
// provisioning client to receive Messaging events.
public MessagingNotifier
    extends MessagingAdapter

    //Extend the MessagingAdapter class.
{
    public void connectionStopped(MessagingEvent ev)

    //Override connectionStopped.
    {
        doNotification(ev.getAddress(), ev.getPort());

        //Do the notification.
    }
}
```

Step 2 Register the listener and the qualifier for the events using the *PACEConnection* interface.

```
MessagingQualifier qualifier =new MessagingQualifier();
qualifier.setConnectionDown();
MessagingNotifier messagingNotifier = new MessagingNotifier();
connnection.addMessagingListener(messagingNotifier, qualifier
);
```

Step 3 If a connection breaks, the event is generated, and the listener is notified. Whenever connectivity is interrupted, the PACE Connection automatically reconnects to the RDU.

Logging Batch Completions Using Events

A service provider has multiple provisioning clients and wants to log batch completions.

Desired Outcome

When any provisioning client completes a batch, an event is logged in one place.

Step 1 Create a listener for the event. This class must extend the *BatchAdapter* abstract class or implement the *BatchListener* interface. This class must override the *completion(BatchEvent ev)* method in order to log the event.

```
public BatchCompletionLogger
    extends BatchAdapter

    //Extend the BatchAdapterclass.
{
    public void completion(BatchEvent ev)
    //Override completion.
    {
        logBatchCompletion(ev.BatchStatus().getBatchID());
        //Log the completion.
    }
}
```

Step 2 Register the listener and the qualifier for the events using the *PACEConnection* interface.

```
BatchCompletionLogger batchCompletionLogger = new BatchCompletionLogger();
BatchEventQualifier qualifier = new BatchEventQualifier();
connection.addBatchListener(batchCompletionLogger , qualifier
);
```

Step 3 When a batch completes, the event is generated, and the listener is notified.

Getting Detailed Device Information

A service provider wants to allow an administrator to view detailed information for a particular device.

Desired Outcome

The service provider's administrative application displays all known details about a given device, including MAC address, lease information, provisioned status of the device, and the device type (if known).

Step 1 The administrator enters the MAC address for the device being queried into the service provider's administrator user interface.

Step 2 BAC queries the embedded database for the device details.

```
    // The host name or IP address of the RDU. It is
    // recommended that you normally use a fully-qualified domain name
    // since it lends itself to the greatest flexibility going forward.
    // For example, you could change the host running RDU without
    // having to reassign IPs. For that reason, having an alias for
    // the machine is better than a specific name.

    final String rduHost = "localhost";
```

```
// The port number of RDU on the server.

final int rduPort = 49187;

// The user name for connecting to RDU.

final String userName = "admin";

// The password to use with the username.

final String password = "changeme";

// ------------------------------------------------------------------
// DEVICE PARAMETERS, see IPDevice.getDetails()
// ------------------------------------------------------------------

// The MAC address of the modem to be queried. MAC addresses in BAC
// must follow the simple "1,6,XX:XX:XX:XX:XX:XX" format.

final DeviceID modemMACAddress = DeviceID.getInstance("1,6,00:11:22:33:44:55",
    KeyType.MAC_ADDRESS);

// The PACE connection to use throughout the example. When
// executing multiple batches in a single process, it is advisable
// to use a single PACE connection that is retrieved at the start
// of the application. When done with the connection, YOU MUST
// explicitly close the connection with the releaseConnection()
// method call.

PACEConnection connection = null;

// 1) Connect to the Regional Distribution Unit (RDU).
//
// The parameters defined at the beginning of this class are
// used here to establish the connection. Connections are
// maintained until releaseConnection() is called. If
// multiple calls to getInstance() are called with the same
// arguments, you must still call releaseConnection() on each
// connection you received.
//
// The call can fail for one of the following reasons:
// - The hostname / port is incorrect.
// - The authentication credentials are invalid.

try
{
connection = PACEConnectionFactory.getInstance(
// RDU host   rduHost,
// RDU port   rduPort,
// User name  userName,
// Password   password
);
}
catch (PACEConnectionException e)
{
// failed to get a connection

System.out.println("Failed to establish a PACEConnection to [" +
    userName + "@" + rduHost + ":" + rduPort + "]; " +
    e.getMessage());

System.exit(1);
}
```

```
// 2) Create a new batch instance.
//
// To perform any operations in the Provisioning API, you must
// first start a batch. As you make commands against the batch,
// nothing will actually start until you post the batch.
// Multiple batches can be started concurrently against a
// single connection to the RDU.

Batch myBatch = connection.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);

// 3) Register the getDetails(...) with the batch.

// Use the Provisioning API to get all of the information for
// the specified MAC address. Since methods aren't actually
// executed until the batch is posted, the results are not
// returned until after post() completes. The getCommandStatus()
// followed by getData() calls must be used to access the results
// once the batch is posted.

final DeviceID modemMACAddress = DeviceID.getInstance("1,6,00:11:22:33:44:55",
        KeyType.MAC_ADDRESS);

List options = new ArrayList();
     options.add(DeviceDetailsOption.INCLUDE_LEASE_INFO);

myBatch.getDetails(modemMACAddress, options);

// 4) Post the batch to the server.
//
// Executes the batch against the RDU. All of the
// methods are executed in the order entered and the data
// changes are applied against the embedded database in RDU.

BatchStatus bStatus = null;
try
{
    bStatus = myBatch.post();
}
catch (ProvisioningException pe)
{
System.out.println("Failed to query for modem with MAC address [" +
    modemMACAddress + "]; " + pe.getMessage());

System.exit(2);
}
// 5) Check to see if the batch was successfully posted.
//
// Verify if any errors occurred during the execution of the
// batch. Exceptions occur during post() for truly exception
// situations such as failure of connectivity to RDU.
// Batch errors occur for inconsistencies such as no lease
// information for a device requiring activation. Command
```

```
// errors occur when a particular method has problems, such as
// trying to add a device that already exists.

if (bStatus.isError())
{
// Batch error occurred.

System.out.println("Failed to query for modem with MAC address [" +
    modemMACAddress + "]; " + bStatus.getErrorMessage());

System.exit(3);
}
```

Step 3 The service provider's application presents a page of device data details, which can display everything that is known about the requested device. If the device was connected to the service provider's network, this data includes lease information (for example, IP address and relay agent identifier). The data indicates whether the device was provisioned, and if it was, the data also includes the device type.

```
// Successfully queried for device.
System.out.println("Queried for DOCSIS modem with MAC address ["+
    modemMACAddress + "]");

// Display the results of the command (TreeMap is sorted). The
// data returned from the batch call is stored on a per-command
// basis. In this example, there is only one command, but if
// you had multiple commands all possibly returning results, you
// could access each result by the index of when it was added.
// The first method added is always index 0. From the status of
// each command, you can then access the accompanying data by
// using the getData() call. Since methods can return data of
// different types, you will have to cast the response to the
// type indicated in the Provisioning API documentation.

Map<String, Object> deviceDetails = new HashMap<String,
    Object>((Map)bStatus.getCommandStatus(0).getData());

String deviceType = (String)deviceDetails.get(DeviceDetailsKeys.DEVICE_TYPE);
String macAddress = (String)deviceDetails.get(DeviceDetailsKeys.MAC_ADDRESS);
String fqdn = (String)deviceDetails.get(DeviceDetailsKeys.FQDN);
String duid = (String)deviceDetails.get(DeviceDetailsKeys.DUID);
String host = (String)deviceDetails.get(DeviceDetailsKeys.HOST);
String domain = (String)deviceDetails.get(DeviceDetailsKeys.DOMAIN);

// if the device is DocsisModem, get the COS

String cos   = (String)deviceDetails.get(DeviceDetailsKeys.CLASS_OF_SERVICE);
String dhcpCriteria   = (String)deviceDetails.get(DeviceDetailsKeys.DHCP_CRITERIA);
String provGroup = (String)deviceDetails.get(DeviceDetailsKeys.PROV_GROUP);
Boolean isProvisioned = (Boolean)deviceDetails.get(DeviceDetailsKeys.IS_PROVISIONED);
String ownerID = (String)deviceDetails.get(DeviceDetailsKeys.OWNER_ID);
Boolean isRegistered = (Boolean)deviceDetails.get(DeviceDetailsKeys.IS_REGISTERED);
String oidNumber = (String)deviceDetails.get(GenericObjectKeys.OID_REVISION_NUMBER);

// if the device is a modem, get the device behind

String relayAgentMacAddress =
    (String)deviceDetails.get(DeviceDetailsKeys.RELAY_AGENT_MAC);
String relayAgentDUID = (String)deviceDetails.get(DeviceDetailsKeys.RELAY_AGENT_DUID);

// get the map of Device property

Map deviceProperties = (Map)deviceDetails.get(DeviceDetailsKeys.PROPERTIES);
```

```
// get the map of discovery data v4

Map dhcpdiscovermapv4 =
    (Map)deviceDetails.get(DeviceDetailsKeys.DISCOVERED_DATA_DHCPV4);

// if discovery data is not null, get the inform, response, request and environment
// map from discovery data map

Map dhcpInformMap = (Map)dhcpdiscovermapv4.get("INFORM");
Map dhcpRespMap = (Map)dhcpdiscovermapv4.get("RESPONSE");
Map dhcpReqMap = (Map)dhcpdiscovermapv4.get("REQUEST");
Map dhcpEnvMap = (Map)dhcpdiscovermapv4.get("ENVIRONMENT");

// get the map of lease query v4

Map leasemapv4 = (Map)deviceDetails.get(DeviceDetailsKeys.LEASE_QUERY_DATA_DHCPV4);
String leaseTime = (String)leasemapv4.get(CNRNames.DHCP_LEASE_TIME.toString());
String rebindingTime =
    (String)leasemapv4.get(CNRNames.DHCP_REBINDING_TIME.toString());

String clientLastTransTime =
    (String)leasemapv4.get(CNRNames.CLIENT_LAST_TRANSACTION_TIME.toString());
String clientIPAddress= (String)leasemapv4.get(CNRNames.CLIENT_IPADDRESS.toString());
String relayAgentRemoteID=
    (String)leasemapv4.get(CNRNames.RELAY_AGENT_REMOTE_ID.toString());
String relayAgentCircuitID=
    (String)leasemapv4.get(CNRNames.RELAY_AGENT_CIRCUIT_ID.toString());

// get the map of discovery DHCP v6

Map dhcpdiscovermapv6 =
    (Map)deviceDetails.get(DeviceDetailsKeys.DISCOVERED_DATA_DHCPV6);

// if discovery data is not null , get the inform, response, request and environment
// map from discovery data map

Map dhcpv6InformMap = (Map)dhcpdiscovermapv6.get("INFORM");

Map dhcpv6RespMap = (Map)dhcpdiscovermapv6.get("RESPONSE");

Map dhcpv6ReqMap = (Map)dhcpdiscovermapv6.get("REQUEST");

Map dhcpv6RelReqMap = (Map)dhcpdiscovermapv6.get("RELAY_REQUEST");

Map dhcpv6EnvMap = (Map)dhcpdiscovermapv6.get("ENVIRONMENT");

// get the map of lease query V6

Map leasemapv6 = (Map)deviceDetails.get(DeviceDetailsKeys.LEASE_QUERY_DATA_DHCPV6);

String iaprefixkey = (String)leasemapv6.get(CNRNames.IAPREFIX.toString());
String iaaddrkey = (String)leasemapv6.get(CNRNames.IAADDR.toString());
String leasetimev6 = (String)leasemapv6.get(CNRNames.VALID_LIFETIME.toString());
String renewaltimev6 = (String)leasemapv6.get(CNRNames.PREFERRED_LIFETIME.toString());
String dhcplasttranstimev6 =
    (String)leasemapv6.get(CNRNames.CLIENT_LAST_TRANSACTION_TIME);
String clientIpAddressv6 = (String)leasemapv6.get(CNRNames.CLIENT_IPADDRESS);
String relayagentremoteidv6 = (String)leasemapv6.get(CNRNames.RELAY_AGENT_REMOTE_ID);
String relayagentcircuitidv6 =
    (String)leasemapv6.get(CNRNames.RELAY_AGENT_CIRCUIT_ID);
```

Searching Using the Device Type

A service provider wants to allow an administrator to view data for all DOCSIS modems.

Desired Outcome

The service provider's administrative application returns a list of DOCSIS devices.

Step 1 The administrator selects the search option in the service provider's administrator user interface.

Step 2 BAC queries the embedded database for a list of all MAC addresses for the DOCSIS modems.

```
public static void getAllDevicesByDeviceType() throws Exception  {
      DeviceSearchType dst = DeviceSearchType.getByDeviceType(
          DeviceType.getDeviceType(DeviceTypeValues.DOCSIS_MODEM),
          ReturnParameters.ALL);

      RecordSearchResults rs = null;

      SearchBookmark sb = null;

      rs = searchDevice(dst, sb);
      sb = rs.getSearchBookmark();

      while (sb != null)
      {
          // print out the data in the record search result.
          sb = printRecordSearchResults(rs);

          // call the search routine again
          rs = searchDevice(dst, sb);
      }
 }
 private static RecordSearchResults searchDevice(DeviceSearchType dst,
                SearchBookmark sb) throws Exception  {
      RecordSearchResults rs = null;
      final Batch batch = s_conn.newBatch();
      final int numberOfRecordReturn = 10;

      //calling the search API
      batch.searchDevice(dst, sb, numberOfRecordReturn);

      // Call the RDU.
      BatchStatus batchStatus = batch.post();

      // Check for success.
      CommandStatus commandStatus = null;

      if (0 < batchStatus.getCommandCount())
      {
          commandStatus = batchStatus.getCommandStatus(0);
      }
      //check to see if there is an error
      if (batchStatus.isError()
      || batchStatus.isWarning()
      || commandStatus == null
      || commandStatus.isError())
      {
          System.out.println("report batch error.");
          return null;
      }
```

```
      //batch success without error, retrieve the result
      //this is a list of devices
      rs = (RecordSearchResults)commandStatus.getData();
      return rs;
 }


 private static SearchBookmark printRecordSearchResults(RecordSearchResults rs)  throws
Exception  {

      SearchBookmark sb = rs.getSearchBookmark();

      List<RecordData> rdlist = rs.getRecordData();
      Iterator<RecordData> iter = rdlist.iterator();

      while (iter.hasNext())
      {
          RecordData rdObj = iter.next();
          Key keyObj = rdObj.getPrimaryKey();

           System.out.println("DeviceOID: " + ((DeviceID)keyObj).getDeviceId());

          //this is for secondary keys.
          List<Key> deviceList = rdObj.getSecondaryKeys();

          if (deviceList != null && !deviceList.isEmpty())
          {
              for (int i=0; i<deviceList.size(); i++)
              {
                  Key key = deviceList.get(i);
                  System.out.println("DeviceID : " + key.toString());
              }
          }
      }
      return sb;

 }
```

Searching for Devices Using Vendor Prefix or Class of Service

A service provider wants to allow an administrator to search for all devices matching a particular vendor prefix or a particular Class of Service.

Desired Outcome

The service provider's administrative application returns a list of devices matching the requested vendor prefix or the Class of Service.

Step 1 The administrator enters the substring matching the desired vendor prefix into the service provider's administrator user interface.

Step 2 BAC queries the embedded database for a list of all MAC addresses for the devices that match the requested vendor prefix or Class of Service. This example illustrates how you can build the search query to retrieve devices using the MAC address. Also see Searching Using the Device Type, page D-38.

```
DeviceIDPattern pattern = new MACAddressPattern("1,6,22:49:*");

DeviceSearchType dst = DeviceSearchType.getDevices(pattern, ReturnParameters.ALL);

// To set up search for class of service:

DeviceSearchType searchType = DeviceSearchType.getByClassOfService(

new ClassOfServiceName(name), AssociationType

.valueOf(association), ReturnParameters.ALL);
```

Step 3 The service provider's application requests details on these devices from BAC, and presents a page of device data. For each device, the code displays the device type, MAC address, client class, and provisioned status of the device. One device is identified per line.

```
// calling the search procedure

rs = searchDevice(connection, dst, sb);
sb = processRecordSearchResults(rs);

if (rs != null)
{
    while (sb != null)
         {
            // The search returns a search bookmark, which can be used to make
            // the next search call that would return next set of results

            rs = searchDevice(connection, dst, sb);
            sb = processRecordSearchResults(rs);
        }
    }
}
}
```

Preprovisioning PacketCable eMTA

A new customer contacts a service provider to order PacketCable voice service. The customer expects to receive a provisioned embedded MTA.

Desired Outcome

Use this workflow to preprovision an embedded MTA so that the modem MTA component has the appropriate level of service when brought online.

Note This use case skips the call agent provisioning that is required for making telephone calls from eMTAs.

Step 1 The service provider chooses a subscriber username and password for the billing system.

Step 2 The service provider chooses the appropriate Class of Service and DHCP Criteria for the modem component and adds it to BAC.

```
// Create a new connection

PACEConnection conn = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "admin123");


Batch batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION);

// Let's provision the modem and the MTA component in the same
// batch. This can be done because the activation mode of this
// batch is NO_ACTIVATION. More than one device can be operated
// on in a batch if the activation mode does not lead to more
// than one device being reset.
// To add a DOCSIS modem:

List<DeviceID> modemDeviceIDList = new ArrayList<DeviceID>();
modemDeviceIDList.add(new MACAddress("1,6,01:02:03:04:05:06"));
batch.add(
    DeviceType.DOCSIS,      // deviceType: DOCSIS
    modemDeviceIDList,      // macAddress: scanned from the label
    null,                   // hostName: not used in this example
    null,                   // domainName: not used in this example
    "0123-45-6789",         // ownerID: here, account number from billing system
    "Silver",               // classOfService
    "provisionedCM",        // DHCP Criteria: Network Registrar uses this to
                            // select a modem lease granting provisioned IP address
    null                    // properties: not used
);
```

Step 3 The service provider chooses the appropriate Class of Service and DHCP Criteria for the MTA component and adds it to BAC.

```
List<DeviceID> packetcableMTADeviceIDList = new ArrayList<DeviceID>();
packetcableMTADeviceIDList.add(new MACAddress("1,6,01:02:03:04:05:07"));

// Continuation of the batch in Step2
// To add the MTA component:

batch.add(
    DeviceType.PACKET_CABLE_MTA,  // deviceType: PACKET_CABLE_MTA
    packetcableMTADeviceIDList,   // macAddress: scanned from the label
    null,                         // hostName: not used in this example, will be auto
                                  // generated
```

```
        null,                           // domainName: not used in this example, will be
                                        // auto generated. The FqdnKeys.AUTO_FQDN_DOMAIN
                                        // property must be set somewhere in the property
                                        // hierarchy.
        "0123-45-6789",                 // ownerID: here, account number from billing system
        "Silver",                       // ClassOfService
        "provisionedMTA",               // DHCP Criteria: Network Registrar uses this to
                                        // select an MTA lease granting provisioned IP
                                        // address
        null                            // properties: not used
    );

    BatchStatus batchStatus = null;

    // post the batch to RDU server

    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
}
```

Step 4 The embedded MTA gets shipped to the customer.

Step 5 The customer brings the embedded MTA online and makes telephone calls using it.

SNMP Cloning on PacketCable eMTA

An administrator wants to grant SNMP Element Manager access to a PacketCable eMTA.

Desired Outcome

An external Element Manager is granted secure SNMPv3 access to the PacketCable eMTA.

Note Changes made to RW MIB variables are not permanent and are not updated in the BAC configuration for the eMTA. The information written into the eMTA MIB is lost the next time the MTA powers down or resets.

Step 1 Call the provisioning API method, *performOperation(...)*, passing in the MAC address of the MTA and the username of the new user to create on the MTA. This will be the username used in subsequent SNMP calls by the Element Manager.

```
// Create a new connection
PACEConnection conn = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "admin123");

Batch batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,
```

```
        // No need to confirm activation

        ConfirmationMode.NO_CONFIRMATION);

    // NO_ACTIVATION is the activation mode because we don't want to
    // reset the device.
    // NO_CONFIRMATION is the confirmation mode because we are
    // not attempting to reset the device.
    // The goal here is to create a new user on the MTA indicated
    // by the MAC address. The other parameter needed here is the new
    // user name, which is passed in the Map.
    // Create a map that contains one element - the name of
    // the new user to be created on the MTA

    HashMap<String, Object> map = new HashMap<String, Object>();
    map.put( SNMPPropertyKeys.CLONING_USERNAME, "newUser" );

    // The first param is the actual device operation to perform.

    batch.performOperation(
        DeviceOperation.ENABLE_SNMPV3_ACCESS,    // deviceOperation : ENABLE_SNMPV3_ACCESS
        new MACAddress("1,6,00:00:00:00:00:99"), // macORFqdn : MAC Address of the modem
        map                                      // parameters: operation specific
                                                 // parameters
    );

    BatchStatus batchStatus = null;

    // post the batch to RDU server

    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
}
```

Step 2 The provisioning API attempts to perform an SNMPv3 cloning operation to create an entry on the MTA for the new user passed in Step 1. The keys used in the new user entry row are a function of two passwords defined within BAC. These passwords will be made available to the customer and the RDU command passes these passwords (the auth and priv password) through a key localization algorithm to create an auth and priv key. These are stored, along with the new user, in the eMTA's user table.

Note The auth and priv passwords mentioned in this step may be changed by setting `SNMPPropertyKeys.CLONING_AUTH_PASSWORD` (*/snmp/cloning/auth/password*) and `SNMPPropertyKeys.CLONING_PRIV_PASSWORD` (*/snmp/cloning/priv/password*) properties, respectively, in the *rdu.properties* configuration file.

Step 3 The customer issues SNMPv3 requests using the specified username, passwords, and key localization algorithm to allow for secure communication with the MTA.

Incremental Provisioning of PacketCable eMTA

A customer has a PacketCable eMTA in service with its first line (end point) enabled. The customer wants to enable the second telephone line (end point) on the eMTA and connect a telephone to it.

Desired Outcome

The customer should be able to connect a telephone to the second line (end point) on the eMTA and successfully make phone calls from it without any service interruption.

Note In order to use the second line on the eMTA, the Call Agent needs to be configured accordingly. This use case does not address provisioning of call agents.

Step 1 The service provider's application invokes the BAC API to change the Class of Service of the eMTA. The new Class of Service supports two end points on the eMTA. This change in Class of Service does not take effect until the eMTA is reset. Disrupting the eMTA is not desirable; therefore, incremental provisioning is undertaken in the next step.

```
PACEConnection conn = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "admin123");

Batch batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION);

// NO_ACTIVATION is the activation mode because we don't want to
// reset the device.
// NO_CONFIRMATION is the Confirmation mode because we are not
// disrupting the device.

batch.changeClassOfService(
    new MACAddress("1,6,ff:00:ee:11:dd:22"),  / eMTA's MAC address or FQDN
    "twoLineEnabledCOS"                      // This COS supports two lines.
);

BatchStatus batchStatus = null;

// post the batch to RDU server

try
{
    batchStatus = batch.post();
}
catch(ProvisioningException e)
{
    e.printStackTrace();
}
```

Step 2 The service provider's application uses the BAC incremental update feature to set SNMP objects on the eMTA and thereby enabling the service without disrupting the eMTA.

```
// The goal here is to enable a second phone line, assuming one
// phone line is currently enabled.   We will be adding a new
```

```
// row to the pktcNcsEndPntConfigTable.

batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION);

// NO_ACTIVATION is the activation mode because we don't want to
// reset the device.
// NO_CONFIRMATION is the confirmation mode because we are
// not attempting to reset the device.
// Create a map containing one element - the list of SNMP
// variables to set on the MTA

HashMap<String, Object> map = new HashMap<String, Object>();

// Create an SnmpVarList to hold SNMP varbinds

SnmpVarList list = new SnmpVarList();

// An SnmpVariable represents an oid/value/type triple.
// pktcNcsEndPntConfigTable is indexed by the IfNumber, which in this case we will
// assume is interface number 12 (this is the last number in each of the oids below).
// The first variable represents the creation of a new row in
// pktcNcsEndPntConfigTable we are setting the RowStatus
// column (column number 26). The value of 4 indicates that
// a new row is to be created in the active state.

SnmpVariable variable = new SnmpVariable( ".1.3.6.1.4.1.4491.2.2.2.1.2.1.1.26.12",
    "4", SnmpType.INTEGER );
list.add( variable );

// The next variable represents the call agent id for this new
// interface, which we'll assume is 'test.com'

variable = new SnmpVariable( ".1.3.6.1.4.1.4491.2.2.2.1.2.1.1.1.12", "test.com",
    SnmpType.STRING );
list.add( variable );

// The final variable represents the call agent port

variable = new SnmpVariable( ".1.3.6.1.4.1.4491.2.2.2.1.2.1.1.2.12", "2728",
    SnmpType.INTEGER );
list.add( variable );

// Add the SNMP variable list to the Map to use in the API call

map.put( SNMPPropertyKeys.SNMPVAR_LIST, list );

// Invoke the BACC API to do incremental update on the eMTA.

batch.performOperation(
    DeviceOperation.INCREMENTAL_UPDATE,      // device operation
    new MACAddress("1,6,00:00:00:00:00:99"), // MAC Address
    map                                      // Parameters for the operation
);

// post the batch to RDU server
```

```
    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
}
```

Step 3 The eMTA is enabled to use the second telephone line. The eMTA continues to receive the same service, after being reset, because the Class of Service was changed in Step 1.

Preprovisioning DOCSIS Modems with Dynamic Configuration Files

A new customer contacts a service provider to order a DOCSIS modem with high-speed *Gold* data service for two sets of CPE behind it.

Desired Outcome

Use this workflow to preprovision a DOCSIS modem with a Class of Service that uses DOCSIS templates. The dynamic configuration file generated from the templates is used while the modem comes online.

Step 1 The service provider chooses a subscriber username and password for the billing system.

Step 2 The service provider chooses Gold Class of Service, and the appropriate DHCP Criteria, and then adds the cable modem to BAC.

```
PACEConnection conn = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "admin123");

Batch batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION);


Map<String, Object> properties = new HashMap<String, Object>();

// Set the property PolicyKeys.COMPUTER_PROMISCUOUS_MODE_ENABLED to enable
// promiscuous mode on modem

properties.put(PolicyKeys.COMPUTER_PROMISCUOUS_MODE_ENABLED, Boolean.TRUE);

// enable promiscuous mode by changing the technology default

batch.changeDefaults(DeviceType.DOCSIS,properties, null);

BatchStatus batchStatus = null;

// post the batch to RDU server
```

```
    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
    // No CPE DHCP Criteria is specified.
    // The CPE behind the modem will use the default provisioned
    // promiscuous CPE DHCP criteria specified in the system defaults.
    // This custom property corresponds to a macro variable in the
    // DOCSIS template for "gold" class of service indicating the
    // maximum number of CPE allowed behind this modem. We set it
    // to two sets of CPE from this customer.

    properties = new HashMap<String, Object>();
    properties.put("docsis-max-cpes", "2");

    batch = conn.newBatch(

        // No reset

        ActivationMode.NO_ACTIVATION,

        // No need to confirm activation

        ConfirmationMode.NO_CONFIRMATION,

        // No publishing to external database

        PublishingMode.NO_PUBLISHING);

    // To add a DOCSIS modem:

    List<DeviceID> deviceIDList = new ArrayList<DeviceID>();
    deviceIDList.add(new MACAddress("1,6,01:02:03:04:05:06"));
    batch.add(
        DeviceType.DOCSIS,     // deviceType: DOCSIS
        deviceIDList,          // macAddress: scanned from the label
        null,                  // hostName: not used in this example
        null,                  // domainName: not used in this example
        "0123-45-6789",        // ownerID: here, account number from billing system
        "gold",                // classOfService:
        "provisionedCM",       // DHCP Criteria: Network Registrar uses this to
                               // select a modem lease granting provisioned IP address
        properties             // properties:
    );

    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
}
```

Step 3 The cable modem is shipped to the customer.

Step 4 The customer brings the cable modem online and connects the computers behind it.

Optimistic Locking

An instance of the service provider application needs to ensure that it is not overwriting the changes made by another instance of the same application.

Desired Outcome

Use this workflow to demonstrate the optimistic locking capabilities provided by the BAC API.

Note Locking of objects is done in multiuser systems to preserve integrity of changes, so that one person's changes do not accidentally get overwritten by another. With optimistic locking, you write your program assuming that any commit has a chance to fail if at least one of the objects being committed was changed by someone else since you began the transaction.

Step 1 The service representative selects the search option in the service provider's user interface and enters the cable modem's MAC address.

Step 2 BAC queries the embedded database, gets the details of the device, and the MSO user interface displays the information.

```
PACEConnection conn = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "admin123");

Batch batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);


final DeviceID modemMACAddress = DeviceID.getInstance("1,6,00:11:22:33:44:55",
        KeyType.MAC_ADDRESS);
List<DeviceDetailsOption> options = new ArrayList<DeviceDetailsOption>();

options.add(DeviceDetailsOption.INCLUDE_LEASE_INFO);

// MSO admin UI calls the provisioning API to query the details
// for the requested device. Query may be performed based on MAC
// address or IP address, depending on what is known about the
// device.

batch.getDetails(modemMACAddress, options);

// post the batch to RDU server

BatchStatus batchStatus = null;

try
{
    batchStatus = batch.post();
}
```

```
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
```

Step 3 The service representative attempts to change the Class of Service and the DHCP Criteria of the modem using the user interface. This in turn invokes the BAC API.

```
    Map<String, Object> deviceDetails = new TreeMap((Map<String,
        Object>)batchStatus.getCommandStatus(0).getData());

    //  extract device detail data from the map

    String deviceType = (String)deviceDetails.get(DeviceDetailsKeys.DEVICE_TYPE);
    String macAddress = (String)deviceDetails.get(DeviceDetailsKeys.MAC_ADDRESS);
    String relayAgentID = (String)deviceDetails.get(DeviceDetailsKeys.RELAY_AGENT_MAC);
    Boolean isProvisioned = (Boolean)deviceDetails.get(DeviceDetailsKeys.IS_PROVISIONED);

    //  Let's save the OID_REVISION_NUMBER property so that we can set it in
    //  step 3.

    String oidRevisionNumber =
        (String)deviceDetails.get(GenericObjectKeys.OID_REVISION_NUMBER);

    // We need a reference to Batch instance so that ensureConsistency()
    // method can be invoked on it.

    batch = conn.newBatch() ;
    List<String> oidList = new ArrayList<String>();

    // Add the oid-rev number saved from step 2 to the list

    oidList.add(oidRevisionNumber);

    // Sends a list of OID revision numbers to validate before processing the
    // batch. This ensures that the objects specified have not been modified
    // since they were last retrieved.

    batch.ensureConsistency(oidList);
    batch.changeClassOfService (
        new MACAddress("1,6,00:11:22:33:44:55"),  // macORFqdn: unique identifier for the
                                                  // device.
        "gold"                                    // newCOSName : Class of service name.
    );

    batch.changeDHCPCriteria (
        new MACAddress("1,6,00:11:22:33:44:55"),  // macORFqdn: unique identifier for the
                                                  // device.
        "specialDHCPCriteria"                     // newDHCPCriteria : New DHCP Criteria.
    );

    // This batch fails with BatchStatusCodes.BATCH_NOT_CONSISTENT,
    // in case if the device is updated by another client in the meantime.
    // If a conflict occurs, then the service provider client
    // is responsible for resolving the conflict by querying the database
    // again and then applying changes appropriately.
    }
}
```

Step 4 The user is ready to receive Gold Class of Service with appropriate DHCP Criteria.

Temporarily Throttling a Subscriber's Bandwidth

An MSO has a service that allows a subscriber to download only 10 MB of data a month. Once the subscriber reaches that limit, their downstream bandwidth is turned down from 10 MB to 56 K. When the month is over they are moved back up to 10 MB.

Note You may want to consider changing upstream bandwidth as well, because peer-to-peer users and users who run websites tend to have heavy upload bandwidth.

Desired Outcome

Use this workflow to move subscribers up and down in bandwidth according to their terms of agreement.

Step 1 The MSO has a rate tracking system, such as *NetFlow*, which keeps track of each customer's usage by MAC address. Initially a customer is provisioned at the *Gold* Class of Service level with 1 MB downstream.

Step 2 When the rate tracking software determines that a subscriber has reached the 10-MB limit it notifies the OSS. The OSS then makes a call into the BAC API to change the subscriber's Class of Service from *Gold* to *Gold-throttled*.

```
PACEConnection conn = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "admin123");

Batch batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);


// AUTOMATIC is the activation mode because we are
// attempting to reset the modem so that it
// receives low bandwidth service.
// NO_CONFIRMATION is the confirmation mode
// because we do not want the batch to fail if we cannot
// reset the modem. If the modem is off, then it will
// be disabled when it is turned back on.
// Let's change the COS of the device so that it restricts
// bandwidth usage of the modem.

batch.changeClassOfService(
    new MACAddress("1,6,00:11:22:33:44:55"),  // macAddress: unique identifier for
                                              // this modem
    "Gold-throttled"                          // newClassOfService: restricts
                                              // bandwidth usage to 56k
);

BatchStatus batchStatus = null;

// post the batch to RDU server
```

```
    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
}
```

Step 3 At the end of the billing period, the OSS calls the BAC API to change the subscriber's Class of Service back to *Gold*.

Preprovisioning CableHome WAN-MAN

A new customer contacts a service provider to order home networking service. The customer expects a provisioned CableHome device.

Desired Outcome

Use this workflow to preprovision a CableHome device so that the cable modem and WAN-MAN components on it will have the appropriate level of service when brought online.

Step 1 The service provider chooses a subscriber username and password for the billing system.

Step 2 The service provider chooses the appropriate Class of Service and the DHCP Criteria for the modem component, then adds it to BAC.

```
PACEConnection conn = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "admin123");

Batch batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation
    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);

// Let's provision the modem and the WAN-Man component in the same
// batch.
// To add a DOCSIS modem:

List<DeviceID> docisDeviceIDList = new ArrayList<DeviceID>();
docisDeviceIDList.add(new MACAddress("1,6,01:02:03:04:05:06"));
batch.add(
    DeviceType.DOCSIS,     // deviceType: DOCSIS
    docisDeviceIDList,     // macAddress: scanned from the label
    null,                  // hostName: not used in this example
    null,                  // domainName: not used in this example
    "0123-45-6789",        // ownerID: here, account number from billing system
    "Silver",              // classOfService
```

```
        "provisionedCM",        // DHCP Criteria: Network Registrar uses this to
                                // select a modem lease granting provisioned IP address
        null                    // properties: not used
    );
```

Step 3 The service provider chooses the appropriate Class of Service and DHCP Criteria for the WAN-MAN component, then adds it to BAC.

```
    List<DeviceID> wanManDeviceIDList = new ArrayList<DeviceID>();
    wanManDeviceIDList.add(new MACAddress("1,6,01:02:03:04:05:07"));
    batch.add(
        DeviceType.CABLEHOME_WAN_MAN,   // deviceType: CABLEHOME_WAN_MAN
        wanManDeviceIDList,             // macAddress: scanned from the label
        null,                           // hostName: not used in this example
        null,                           // domainName: not used in this example
        "0123-45-6789",                 // ownerID: here, account number from billing
                                        // system
        "silverWanMan",                 // classOfService
        "provisionedWanMan",            // DHCP Criteria: Network Registrar uses this to
                                        // select a modem lease granting provisioned IP
                                        // address
        null                            // properties: not used
    );
}
```

Step 4 The CableHome device is shipped to the customer.

Step 5 The customer brings the CableHome device online.

CableHome with Firewall Configuration

A customer contacts a service provider to order a home networking service with the firewall feature enabled. The customer expects to receive a provisioned CableHome device.

Desired Outcome

Use this workflow to preprovision a CableHome device so that the cable modem and the WAN-MAN components on it have the appropriate level of service when brought online.

Step 1 The service provider chooses a subscriber username and password for the billing system.

Step 2 The service provider chooses the appropriate Class of Service and DHCP Criteria for the cable modem component, then adds it to BAC.

```
PACEConnection conn = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "admin123");

Batch batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database
```

```
    PublishingMode.NO_PUBLISHING);

// Let's provision the modem and the WAN-Man component in the same
// batch.
// To add a DOCSIS modem:

List<DeviceID> docisDeviceIDList = new ArrayList<DeviceID>();
docisDeviceIDList.add(new MACAddress("1,6,01:02:03:04:05:06"));
batch.add(
    DeviceType.DOCSIS,      // deviceType: DOCSIS
    docisDeviceIDList,      // macAddress: scanned from the label
    null,                   // hostName: not used in this example
    null,                   // domainName: not used in this example
    "0123-45-6789",         // ownerID: here, account number from billing system
    "Silver",               // classOfService
    "provisionedCM",        // DHCP Criteria: Network Registrar uses this to
                            // select a modem lease granting provisioned IP address
    null                    // properties: not used
);
```

Step 3 The service provider chooses the appropriate Class of Service and DHCP Criteria for the WAN-MAN component and adds it to BAC.

```
//      Continuation of the batch in Step 2
// To add the WAN-Man component:
// Create a Map to contain WanMan's properties

Map<String, Object> properties = new HashMap<String, Object>();

// The fire wall configuration for the Wan Man component is specified
// using the CableHomeKeys.CABLEHOME_WAN_MAN_FIREWALL_FILE property.
// This use case assumes that the firewall configuration file named
// "firewall_file.cfg" is already present in the RDU database and the
// firewall configuration is enabled in the Wan Man configuration file
// specified with the corresponding class of service.

properties.put(CableHomeKeys.CABLEHOME_WAN_MAN_FIREWALL_FILE, "firewall_file.cfg");

List<DeviceID> wanManDeviceIDList = new ArrayList<DeviceID>();
wanManDeviceIDList.add(new MACAddress("1,6,01:02:03:04:05:07"));
batch.add(
    DeviceType.CABLEHOME_WAN_MAN,    // deviceType: CABLEHOME_WAN_MAN
    wanManDeviceIDList,              // macAddress: scanned from the label
    null,                         // hostName: not used in this example
    null,                         // domainName: not used in this example
    "0123-45-6789",               // ownerID: here, account number from billing system
    "silverWanMan",               // classOfService
    "provisionedWanMan",          // DHCP Criteria: Network Registrar uses this to
                                  // select a modem lease granting provisioned IP
                                  // address
    null                          // properties: not used
);

BatchStatus batchStatus = null;

// post the batch to RDU server

try
{
    batchStatus = batch.post();
}
catch(ProvisioningException e)
{
```

```
            e.printStackTrace();
        }
    }
```

Step 4 The CableHome device is shipped to the customer.

Step 5 The customer brings the CableHome device online and the cable modem and the WAN-MAN component get provisioned IP addresses and proper configuration files.

Retrieving Device Capabilities for CableHome WAN-MAN

A service provider wants to allow an administrator to view capabilities information for a CableHome WAN-MAN device.

Desired Outcome

The service provider's administrative application displays all known details about a given CableHome WAN-MAN component, including MAC address, lease information, provisioned status, and the device capabilities information.

Step 1 The administrator enters the MAC address of the WAN-MAN being queried into the service provider's user interface.

Step 2 BAC queries the embedded database for details of the device identified using the MAC address entered.

```
PACEConnection conn = PACEConnectionFactory.getInstance(
    "localhost", 49187, "admin", "admin123");


Batch batch = conn.newBatch(

    // No reset

    ActivationMode.NO_ACTIVATION,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION,

    // No publishing to external database

    PublishingMode.NO_PUBLISHING);


final DeviceID modemMACAddress = DeviceID.getInstance("1,6,00:11:22:33:44:55",
        KeyType.MAC_ADDRESS);

List<DeviceDetailsOption> options = new ArrayList<DeviceDetailsOption>();
options.add(DeviceDetailsOption.INCLUDE_LEASE_INFO);

// MSO admin UI calls the provisioning API to query the details
// for the requested device. Query may be performed based on MAC
// address or IP address, depending on what is known about the
// device.

batch.getDetails(modemMACAddress, options);

// post the batch to RDU server
```

```
    BatchStatus batchStatus = null;
    try
    {
        batchStatus = batch.post();
    }
    catch(ProvisioningException e)
    {
        e.printStackTrace();
    }
}
```

Step 3 The service provider's application then presents a page of device data details, which can display everything that is known about the requested device. If the device was connected to the service provider's network, this data includes lease information, such as the IP address or the relay agent identifier. This data indicates whether the device is provisioned. If it is provisioned, the data also includes the device type and device capabilities information.

```
    Map<String, Object> deviceDetails = new TreeMap((Map<String,
        Object>)batchStatus.getCommandStatus(0).getData());

    //  extract device detail data from the map

    String deviceType = (String)deviceDetails.get(DeviceDetailsKeys.DEVICE_TYPE);
    String macAddress = (String)deviceDetails.get(DeviceDetailsKeys.MAC_ADDRESS);
    String relayAgentID = (String)deviceDetails.get(DeviceDetailsKeys.RELAY_AGENT_MAC);
    Boolean isProvisioned = (Boolean)deviceDetails.get(DeviceDetailsKeys.IS_PROVISIONED);

    String deviceID = (String) deviceDetails.get(CNRNames.DEVICE_ID.toString());
    String serNum = (String) deviceDetails.get(CNRNames.DEVICE_SERIAL_NUMBER.toString());
    String hwVer = (String)
        deviceDetails.get(CNRNames.HARDWARE_VERSION_NUMBER.toString());
    String swVer = (String)
        deviceDetails.get(CNRNames.SOFTWARE_VERSION_NUMBER.toString());
    String brVer = (String) deviceDetails.get(CNRNames.BOOT_ROM_VERSION.toString());
    String vendorOui = (String) deviceDetails.get(CNRNames.VENDOR_OUI.toString());
    String modelNum = (String) deviceDetails.get(CNRNames.MODEL_NUMBER.toString());
    String vendorNum = (String) deviceDetails.get(CNRNames.VENDOR_NAME.toString());

    //  The admin UI now formats and prints the detail data to a view page
    }
}
```

Self-Provisioning CableHome WAN-MAN

A subscriber has a computer with a browser application installed in a single-dwelling unit and has purchased an embedded CableHome device.

Desired Outcome

Use this workflow to bring a new unprovisioned embedded CableHome device online with the appropriate level of service, and give the subscriber Internet access from computers connected to the embedded CableHome device.

Step 1 The subscriber purchases an embedded CableHome device and installs it at home.

Step 2 The subscriber powers on the embedded CableHome device. BAC gives the embedded cable modem restricted access, allowing two sets of CPE: one for the CableHome WAN-MAN and the other for the computer.

Note This use case assumes an unprovisioned DOCSIS modem allows two sets of CPE behind it. Until configured to do otherwise, BAC supports only a single device behind an unprovisioned DOCSIS modem. You can change this behavior by defining an appropriate Class of Service that supports two sets of CPE and then using it as the default Class of Service for DOCSIS devices.

Step 3 BAC configures the CableHome WAN-MAN, including IP connectivity and downloading the default CableHome boot file. The default CableHome boot file configures the CableHome device in passthrough mode. The CableHome device is still unprovisioned.

Step 4 The subscriber connects the computer to the CableHome device. The computer gets an unprovisioned (restricted) IP address. The subscriber starts a browser application on the computer. A spoofing DNS server points the browser to the service provider's registration server (for example, an OSS user interface or a mediator).

Step 5 The subscriber uses the service provider's user interface to complete the steps required for cable modem registration, including selecting a Class of Service. The subscriber also selects a CableHome Class of Service.

Step 6 The service provider's user interface passes the subscriber's information to BAC, including the selected Class of Service for cable modem and CableHome, and computer IP address. The subscriber is then registered with BAC.

Step 7 The user interface prompts the subscriber to reboot the computer.

Step 8 The provisioning client calls *performOperation(…)* to reboot the modem and gives the modem provisioned access.

```
// create a new batch

batch = conn.newBatch(

    // No reset

    ActivationMode.AUTOMATIC,

    // No need to confirm activation

    ConfirmationMode.NO_CONFIRMATION);


// register performOperation command to the batch

batch.performOperation(DeviceOperation.RESET, modemMACAddressObject, null);

// post the batch to RDU server

try
{
    batchStatus = batch.post();
}
catch(ProvisioningException e)
{
    e.printStackTrace();
```

```
    }
}
```

Step 9 When the computer is rebooted, it receives a new IP address from the CableHome device's DHCP server. The cable modem and the CableHome device are now both provisioned. Now the subscriber can connect a number of computers to the Ethernet ports of the CableHome device and they have access to the Internet.

Note If the configuration file supplied to the WAN-MAN component enables the WAN-Data component on the box, it will be provisioned in the promiscuous mode. This assumes that the promiscuous mode is enabled at the technology defaults level for the DeviceType.CABLEHOME_WAN_DATA device type.

APPENDIX E

FAQs on Provisioning Broadband Access Center

This appendix lists answers to FAQs about BAC provisioning.

- BAC Configuration, page E-1
- IPv6 Configuration, page E-2
- CMTS Configuration, page E-5

BAC Configuration

This section features FAQs related to general BAC configurations.

- How do I enable or disable Network Registrar extensions?
- How do I enable tracing for Network Registrar extensions?
- Why is my DPE server registration failing?

How do I enable or disable Network Registrar extensions?

The procedures described in this section assume that:

- The BAC component is installed in */opt/CSCObac*.
- Cisco Network Registrar is installed in */opt/nwreg2*.

To manually install Network Registrar extension points:

Step 1 Log in to the Network Registrar server, with *root* access.

Step 2 Copy the *libbprextensions.so* directory to the *NR_HOME/local/extensions/dhcp/dex/* directory.

Step 3 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 in to 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 in to 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 is my DPE server registration failing?

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 BAC.

IPv6 Configuration

This section features FAQs related to IPv6 while configuring BAC.

- How do I enable provisioning in IPv6 for the DPE?
- How do I configure an IPv4 interface for provisioning?

- DPE is configured for IPv6 provisioning, but BAC 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 the 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 BAC 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 BAC administrator user interface:

Step 1 Click **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
# 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 E-1 lists the various modem states in both IPv4 and IPv6.

Table E-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
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

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

BAC — An integrated solution for data-over-cable service providers to configure and manage broadband modems, and enable and administer subscriber self-registration and activation. BAC 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.

Broadband Access Center — *See* BAC.

Broadband Access Center for Cable — *See* BAC.

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 BAC 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 BAC 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 BAC 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).

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.

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 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 BAC 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.

R

RDU Regional distribution unit. The primary server in the BAC provisioning system, it manages generation of device configurations, processes all API requests, and manages the BAC 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. BAC 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 BAC 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

A

administrator user interface
- about 2-13, 9-3
- class of service
  - about 4-3, 13-1
  - adding 13-2
  - deleting 13-4
  - modifying 13-3
- configuring, interface 11-1
- custom property, configuring 13-5
  - *See also* property hierarchy
- defaults, configuring
  - about 13-6
  - CableHome WAN-Data 13-7
  - CableHome WAN-MAN 13-7
  - computer 13-7
  - DOCSIS 13-8
  - Network Registrar 13-9
  - PacketCable 13-11
  - RDU 13-11
  - STB 13-14
  - system 13-12
- devices, managing
  - about 12-4
  - adding record 12-13
  - deleting record 12-14
  - modifying record 12-14
  - regenerating configuration 12-14
  - relating and unrelating 12-15
  - resetting 12-17
  - searching 12-4
  - viewing details 12-9
- DHCP criteria, managing
  - about 13-15
  - adding 13-15
  - deleting 13-16
  - modifying 13-16
- discovered data, viewing 12-9
  - *See also* discovered data
- files, managing 13-17
  - adding 13-18
  - deleting 13-22
  - exporting 13-22
  - replacing 13-21
  - viewing 13-19
- icons, understanding 11-6
- licenses, managing 13-22
  - adding 13-24
  - deleting 13-25
  - modifying 13-24
- logging in, interface 11-2
- logging out, interface 11-5
- nodes, managing
  - adding 12-18
  - deleting 12-20
  - modifying 12-20
  - searching devices in node 12-19
  - viewing details 12-21
- node types, managing
  - adding 12-17
  - deleting 12-18
  - modifying 12-18
  - relating and unrelating to nodes 12-20
- provisioning data, publishing 13-28
  - disabling plug-in 13-29

enabling plug-in 13-29
modifying plug-in settings 13-29
RDU extensions, managing 13-25
installing custom points 13-28
required points (table) 13-26
viewing 13-28
writing new class 13-26
search results, configuring 12-8
servers, monitoring
DPE 12-21
Network Registrar extensions 12-25
provisioning group 12-27
RDU 12-29
statistics 10-15
starting, stopping interface 11-1
users, managing
about 12-1
adding 12-2
deleting 12-3
modifying 12-3
adminui.properties file 11-1
agent, SNMP
about 10-9
MIB support 10-9
monitoring servers using 10-9
snmpAgentCfgUtil.sh tool, using 10-10
AIC echo, enabling 6-22
alert messages A-1
message format A-1
relating to
DPE A-3
Network Registrar extensions A-5
process watchdog A-4
RDU A-2
API use cases
See use cases
architecture 2-1
administrator user interface 2-13, 9-3
DPE 2-5
DOCSIS shared secret and 2-10
licensing 2-6
server states 2-8
synchronization with RDU 2-7
TACACS+ authentication 2-7
TFTP server 2-9, 6-11
ToD server 2-9
KDC 2-12
certificates 2-13, 7-9
default KDC properties 7-7
licenses 2-13, 7-9
multiple realm support 7-10
logging 2-16, 10-1
log files 10-4, 10-7, 10-8, 16-2
log files, rotating 10-3
log levels and structure 10-1
severity levels (table) 10-2
severity levels, configuring 10-3
MIBs 10-9
Network Registrar 2-11
DHCP and 2-11
DNS and 2-12
process watchdog 9-1
provisioning groups
about 2-14
capabilities 2-15
static, dynamic provisioning 2-15
RDU 2-3
configuration generation 2-3
service-level selection 2-4
registration modes 4-8
mixed 4-9
promiscuous 4-9
roaming 4-9
standard 4-8
SNMP agent 9-4, 10-9
audit.log 10-4
automatic FQDN generation 13-30

B

BAC 14-1
backup and recovery of database 15-4
See also database management
backupDb.sh tool 15-4
bundleState.sh 16-10
bundling server state 16-10

C

CableHome
configuring 8-1
DPE 8-4
Network Registrar 8-3
RDU 8-3
WAN defaults 13-7
option support B-22
provisioning
about 1-3
flow (figure and table) 8-1
provisioning, non-secure
checklist 3-10
CableLabs certificate trust hierarchy 16-18
certificate, validating 16-19
MTA device 16-20
device certificate 16-21
manufacturer certificate 16-21
root certificate 16-20
operational ancillary certificates
Delivery Function (DF) certificate 16-26
KDC certificate 16-25
PacketCable Server certificates 16-26
service provider 16-22
CA certificate 16-23
CA certificate, local system 16-24
root certificate 16-23
CableLabs code verification certificate hierarchy
CA certificate 16-29
certificate revocation lists 16-31
manufacturer certificate 16-30
requirements 16-28
root CA certificate 16-29
service provider certificate 16-30
captureConfiguration.sh 16-10
cautions, regarding
class of service, adding
CableHome 13-3
DOCSIS modem 13-3
PacketCable device 13-3
cnr_ep.properties file, setting property instances C-1
custom properties, deleting 13-5
DHCP options, settings in Network Registrar 3-3
DSS, configuring multiple in provisioning group 2-10
evaluation license key, deploying in network 13-23
KDC certificates, missing or uninstalled 2-13, 7-9
KDC license, copying 7-10
template files, deleting 13-22
troubleshooting devices by device ID 16-3
certificate trust hierarchy, PacketCable 16-18
ancillary certificates
delivery function 16-26
KDC 16-25
PacketCable server 16-26
certificate validation 16-19
MTA 16-20
device certificate 16-21
manufacturer certificate 16-21
root certificate 16-20
service provider 16-22
CA certificate 16-23
CA certificate, local system 16-24
root certificate 16-23
changeNRProperties.sh tool 14-11
CISCO-BACC-DPE-MIB 10-10
CISCO-BACC-RDU-MIB 10-9
CISCO-BACC-SERVER-MIB 10-9
CISCO-NMS-APPL-HEALTH-MIB 10-10

class of service, managing
configuring
adding a class 13-2
deleting a class 13-4
modifying a class 13-3
overview 4-3
code verification certificate hierarchy, PacketCable 16-28
CA certificate 16-29
code verification certificate requirements 16-28
manufacturer certificate 16-30
root CA certificate 16-29
service provider certificate 16-30
code verification certificate requirements 16-28
computer defaults, configuring 13-7
configuration file utility, using 5-21, 5-22
adding template 5-23
binary file
converting to template files 5-23
external, viewing 5-33
local, viewing 5-32
output, specifying 5-31
dynamic DOCSIS version selection, configuring 6-12
macro variables
specifying a device for 5-30
specifying from CLI 5-29
PacketCable Basic flow, activating 5-34
template processing, testing
external template files 5-26
local template file and adding shared secret 5-27
local template files 5-25
configuration workflows and checklists (tables) 3-1
component workflows 3-1
DPE checklist, IPv4 3-2
DPE checklist, IPv6 3-3
Network Registrar checklist, DHCPv4 3-4
Network Registrar checklist, DHCPv6 3-4
RDU checklist 3-1
technology workflows 3-5
CableHome 3-10
DOCSIS 3-5
PacketCable, Basic 3-8
PacketCable, Secure 3-6
configuring BAC
class of service
adding a class 13-2
deleting 13-4
modifying 13-3
custom properties 13-5
See also property hierarchy
defaults 13-6
CableHome WAN 13-7
computer 13-7
DOCSIS 13-8
Network Registrar extensions 13-9
PacketCable 13-11
RDU 13-11
STB 13-14
system 13-12
DHCP criteria
adding criteria 13-15
deleting criteria 13-16
modifying criteria 13-16
files, managing 13-17
adding files 13-18
deleting files 13-22
exporting files 13-22
replacing files 13-21
static versus template files 4-6
viewing files 13-19
FQDN, automatic generation
format 13-30
properties 13-31
sample 13-31
validation 13-31
IPv6 support 6-13
enabling 6-14
lease query in BAC 6-20
lease query using BAC as relay agent 6-21

workflows 3-3, 3-4, 6-19
license keys, managing
adding a license 13-24
deleting a license 13-25
modifying a license 13-24
provisioning data, publishing 13-28
datastore changes 13-29
plug-in settings, modifying 13-29
RDU unit extensions, managing 13-25
custom extension points, installing 13-28
new class, writing 13-26
viewing 13-28
SNMPv3 cloning on RDU, DPE 7-29
key generation 7-29
key material 7-29
SRV record in DNS server 7-28
configuring CableHome
DPE 8-4
Network Registrar 8-3
provisioning flow 8-1
RDU 8-3
configuring DOCSIS
DPE TFTP IP validation 6-11
dynamic configuration TLVs 6-10
dynamic DOCSIS version selection 6-11
IPv6 support
addressing 6-14
attributes versus options 6-15
DHCPv6 options 6-15
enabling on system 6-14
features and benefits 6-13
lease query 6-19 to 6-23
properties for discovered data 6-16
single versus dual stack 6-15
workflows 3-3, 3-4, 6-19
provisioning flow 6-1
troubleshooting 16-10
version support 6-11
workflow 3-5
configuring Network Registrar
and CableHome 8-3
defaults 13-9
DHCPv4 workflow 3-4
DHCPv6 workflow 3-4
SRV record in DNS server 7-28 to 7-29
configuring PacketCable 7-1
automatic FQDN generation 13-30
certificate trust hierarchies 16-31
certificate trust hierarchies, certificate revocation 16-28
defaults 13-11
Euro PacketCable
about 7-31
MIBs, configuring 7-31
FQDN, automatic generation 13-30
PacketCable Basic
about 1-2
provisioning flow 7-30
PacketCable Secure
about 7-1
KDC, configuring for multiple realms 7-10
KDC properties 7-6
provisioning flow 7-1
service keys, generating via KeyGen tool 14-8
troubleshooting eMTA provisioning
components involved 16-11
key variables 16-13
scenarios 16-15
tools 16-14
configuring RDU
CableHome and
WAN-Data 8-4
WAN-MAN 8-3
defaults 13-11
workflow checklist (table) 3-1
cos/docsis/file/1.0, 1.1, 2.0, 3.0 6-12
CPE provisioning
about 4-8

configuration generation 4-6
data discovered from device 4-4
IPv4 (table) 4-4
IPv6 (table) 4-5
properties 6-16
viewing from administrator user interface 4-5
device configuration workflow
initial, preprovisioned 4-9
initial, self-provisioned 4-9
update 4-12
device object model 4-2
device object relationship (table) 4-3
device support 4-1
DUID versus MAC address 4-5
promiscuous access 4-13
property hierarchy and 4-7
registration modes
about 4-8
mixed 4-9
promiscuous 4-9, 4-13 to 4-20
roaming 4-9
standard 4-8
static versus template files 4-6
workflows
configuration update 4-12
initial configuration, preprovisioned device 4-9
initial configuration, self-provisioned device 4-9
custom property
about 4-8
configuring 13-5
promiscuous devices 4-20

D

database
See database management
database management
backup and recovery 15-4
backing up 15-4
recovering 15-5
restoring 15-6
disk space
out of space, handling 15-3
requirements 15-3
failure resiliency 15-1
files 15-2
automatic log management 15-2
DB_VERSION 15-3
history log 15-3
storage 15-2
transaction log 15-2
location, changing 15-7
RDU, migrating 15-8
Data Over Cable Service Interface Specification
See DOCSIS
defaults, configuring 13-6
CableHome WAN 13-7
WAN-Data 13-7
WAN-MAN 13-7
computer 13-7
DOCSIS 13-8
Network Registrar 13-9
PacketCable 13-11
RDU 13-11
STB 13-14
system 13-12
deleting a license 13-25
device data model
See device object model
device deployment
promiscuous access 4-13
registration modes 4-8
mixed 4-9
promiscuous 4-9
roaming 4-9
standard 4-8
device ID, troubleshooting 16-2
device management 12-4

about 12-12
adding devices 12-13
controls 12-8
deleting devices 12-14
device configurations, regenerating
about 12-14
regenerating configurations 12-15
device details, viewing 12-9
modifying devices 12-14
relating and unrelating devices 12-15
resetting devices 12-17
searching for devices 12-4
troubleshooting devices 16-2
device object model
overview 4-2
relationships (figure) 4-2
relationships (table) 4-3
device provisioning
See CPE provisioning
Device Provisioning Engine
See DPE
device support 1-4
DEX API version 1 6-16
DEX API version 2 6-16
DHCP
criteria defaults, configuring
adding criteria 13-15
deleting criteria 13-16
modifying criteria 13-16
DUID 4-5
lease query ports 6-20
Network Registrar and 2-11, 16-12
v4 versus v6 2-12
DHCP Unique Identifier
See DUID
diagnostics tool, using 16-5
bundleState.sh 16-10
startDiagnostics.sh 16-5
interactive mode 16-6
noninteractive mode 16-7
statusDiagnostics.sh 16-8
stopDiagnostics.sh 16-9
interactive mode 16-9
noninteractive mode 16-9
discovered data 4-4
from IPv4 devices (table) 4-4
from IPv6 devices (table) 4-5
properties 6-16
before BAC 4.0 (table) 6-17
in BAC 4.0, DHCPv4 (table) 6-17
in BAC 4.0, DHCPv6 (table) 6-18
disk_monitor.sh tool 14-12
disk space, monitoring 14-12 to 14-13
DNS
Network Registrar and 2-12
DOCSIS
/cos/docsis/file/1.0, 1.1, 2.0, 3.0 6-12
about 1-2
defaults, configuring 13-8
dynamic configuration TLVs 6-10
dynamic version selection 6-11
configuration file 6-12
IPv6
about 1-2, 6-13
addressing 6-14
attributes versus options 6-15
configuration workflow 6-19
DHCP options 6-15
enabling 6-14
lease query 6-19
provisioning workflow 6-5
single versus dual stack 6-15
MIBs, using with dynamic DOCSIS templates 6-9
option support B-1
provisioning workflow
DHCPv4 6-2
DHCPv6 6-5
shared secret 2-10

version support 6-11
workflow checklist 3-5
DOCSIS shared secret
See DSS
Domain Name System
See DNS
DPE
about 2-5
alerts A-3
configuring
CableHome and 8-4
SNMPv3 cloning 7-29
configuring DOCSIS shared secret 2-10
DSS
about 2-10
resetting 2-10
license keys 2-6
log file
about 10-7
viewing 10-7, 11-2, 12-23
server, viewing details 12-21
server state 2-7
SNMP agent 10-10
SNMPv3 cloning, configuring 7-29
key generation 7-29
key material 7-29
synchronization with RDU 2-7
TACACS+, and DPE authentication 2-7
client settings 2-7
privilege levels 2-7
TFTP server and 6-11
ToD server and 2-9
viewing details 12-21
workflow checklist
IPv4 3-2
IPv6 3-3
dpe.log 10-7
DSS
about 2-10
resetting 2-10
dual stack 6-15
DUID
about 4-5
in automatic FQDN generation 13-30
troubleshooting devices 16-2
versus MAC address 4-5
dynamic DOCSIS version selection
about 6-11
configuration file 6-12
dynamic port for DHCP 6-20

E

embedded Service/Application Functional Entities
See eSAFE
eMTA provisioning for PacketCable, troubleshooting
components
call management server 16-12
DHCP server 16-12
DNS server 16-12
embedded MTA 16-11
KDC 16-12
PacketCable provisioning server 16-12
key variables
certificates 16-13
MTA configuration file 16-14
scope-selection tag 16-14
eSAFE 4-1
ethereal, for troubleshooting 16-15
extension points
See Network Registrar
extensions, RDU 13-25
external files, managing 13-17
adding 13-18
deleting 13-22
exporting 13-22
replacing 13-21
viewing 13-19

F

features, overview 1-4
FQDN, automatic generation
about 13-30
format 13-30
properties 13-31
sample 13-31
validation 13-31

G

GUI
See administrator user interface

I

icons on administrator user interface 11-6
include files 5-3

K

KDC
BAC architecture and 2-12
certificate
creating 14-3
validating 14-4
certificates 7-9
certificates, managing via PKCert.sh tool
creating 14-3
running the PKCert tool 14-2
setting log level for debug output 14-5
validating 14-4
default properties 7-7
licenses 7-9
multiple realm support
about 7-10
configuring 7-11
directory structure (table) 7-11
template, authoring 7-26
verifying service keys 14-10
KeyGen tool
using 14-8
verifying service keys 14-10

L

L2VPN
option support B-1
specification 6-1
template sample 5-5
Layer 2 Virtual Private Networks
See L2VPN
lease query 6-19
about 6-19
autoconfiguration 6-19
about 6-19
enabling and disabling 6-20
configuring 6-20
configuring BAC as relay agent 6-21
AIC echo, enabling 6-22
IPv4 6-21
IPv6 6-21
debugging 6-22
IPv6 use cases 6-23
source IP address 6-20
licenses, managing
about 13-23
adding a license 13-24
deleting a license 13-25
KDC 7-9
modifying a license 13-24
logging
BAC architecture and 2-16, 10-1
log files
DPE 10-7
Network Registrar 10-8

RDU 10-4
rotating 10-3
troubleshooting 16-2
log levels and structures 10-1
log level tool, using 10-4
RDU log level tool, using 10-4
severity levels (table) 10-2
severity log levels, configuring 10-3
logging in 11-2
logging out 11-5
log level tool, using 10-4
setting 10-6
viewing log level 10-6

M

MAC address, troubleshooting devices 16-2
MIBs
CableHome, and SNMP VarBind 5-8
DOCSIS, and SNMP VarBind 5-7
Euro PacketCable, and PacketCable configuration 7-32
PacketCable, and SNMP VarBind 5-7
SNMP agent, and MIB support 10-9
TLV 38, and MIB support 7-31
vendor-specific, adding 5-11
migrating, RDU database 15-8
modes of registration 4-8

N

Network Registrar
about 2-11
DHCP and 2-11
DNS, and 2-12
API versions 6-16
architecture 2-11
attributes 6-16
configuring CableHome 8-3
defaults, configuring 13-9
DHCP and 2-11
dictionaries 6-16
environment 6-16
inform 6-16
request 6-16
response 6-16
DNS
about 2-12
SRV record, configuring 7-28
extension point alerts A-5
extension points, viewing details 12-25
log file 10-8
viewing details 12-25
workflow checklist
for DHCPv4 3-4
for DHCPv6 3-4
Network Registrar defaults, configuring 13-9
Network Registrar log 10-8
nodes, managing 12-17
about 12-18
adding 12-19
deleting 12-20
details, viewing 12-21
modifying 12-20
node types 12-17
adding 12-17
deleting 12-18
modifying 12-18
relating and unrelating node types to nodes 12-20
NRProperties.sh tool, using 14-11

O

option support
CableHome B-22
DOCSIS B-1
PacketCable B-22
organizationally unique identifier

See OUI
OUI 5-5
template (example) 5-5
overview
features and benefits 1-4
product 1-1
technologies supported 1-2

P

PacketCable
about 1-2
BAC properties, mapping to DHCP options
Option 122 and C-2
Option 177 and C-2
Basic 1-2
checklist 3-8
provisioning workflow 7-30
SNMP v2C notifications 7-31
TLV 38 and MIB support 7-31
certificate trust hierarchy 16-18
CableLabs Service Provider 16-22
code verification 16-28
MTA device certificate 16-21
MTA device certificate hierarchy 16-20
revocation 16-28
validation 16-19
defaults, configuring 13-11
device details, viewing 12-8
eMTA provisioning, troubleshooting
components 16-11
key variables 16-13
Euro PacketCable
about 7-31
checklist 3-7
MIBs, configuring 7-32
KeyGen tool, using 14-8
MTAs, SNMPv3 cloning, and
key generation 7-29
key material 7-29
option support B-22
PKCert.sh tool, using 14-2
Secure 1-2
checklist 3-6
provisioning workflow 7-1
service keys, generating 14-8
troubleshooting
logs 16-14
scenarios 16-15
tools 16-15
workflow checklists 3-6
Basic PacketCable 3-8
Euro PacketCable 3-7
Secure PacketCable 3-6
PKCert.sh tool, using 14-2
KDC certificate
creating 14-3
validating 14-4
running 14-2
setting log level for debugging 14-5
process watchdog
about 9-1
alerts A-4
command line, using 9-2
commands (table) 9-2
product overview 1-1
promiscuous access
about 4-13
configuring (table) 4-14
configuring, properties 4-15
generating device configurations for 4-15
property hierarchy and 4-14
promiscuous mode 4-9
property hierarchy
custom properties 4-8
overview 4-7
promiscuous access and 4-14
versus templates 4-7

provisioning data, publishing 13-28
datastore changes 13-29
plug-in settings, modifying 13-29
provisioning group
capabilities
about 2-15
viewing 12-22, 12-26, 12-28
concepts 2-14
details, viewing 12-27
provisioning use cases, API D-1

R

RDU
about 2-3
alert messages A-2
configuration, generating 2-3
configuration file utility, running 5-22
configuring, and CableHome
WAN-Data 8-4
WAN-MAN 8-4
database, migrating 15-8
defaults, configuring 13-11
details, viewing 12-29
device configuration, generating 2-3
extensions, managing
custom extension points, installing 13-28
new class, writing 13-26
viewing 13-28
log files 10-4
audit.log 10-4
default log level 10-4
rdu.log 10-4
setLogLevel.sh tool, using 10-4
troubleshooting.log 16-2
viewing 10-4, 11-2, 12-30
log level tool, using 10-4
current log level, viewing 10-6
setting 10-6
MIBs 10-9
migrating database 15-8
runCfgUtil.sh, running 5-22
server details, viewing 12-29
service level, selecting 2-4
SNMP agent 10-9
SNMPv3 cloning, configuring 7-29
key generation 7-29
key material 7-29
templates 5-1
workflow checklist 3-1
rdu.log 10-4
recoverDb.sh tool 15-5
registration modes
mixed 4-9
promiscuous 4-9
roaming 4-9
standard 4-8
restoreDb.sh tool 15-6
RFCs supported 1-3
RNG-200 eSTB 1-4, 4-1
runCfgUtil.sh script, running 5-22

S

servers, monitoring and troubleshooting 16-5
bundleState.sh 16-10
startDiagnostics.sh 16-5
interactive mode 16-6
noninteractive mode 16-7
statusDiagnostics.sh 16-8
stopDiagnostics.sh
interactive mode 16-9
noninteractive mode 16-9
servers, viewing 12-21
DPE 12-21
Network Registrar extensions 12-25
provisioning groups 12-27
RDU 12-29

See also servers, monitoring and troubleshooting
service keys, PacketCable 14-8
setLogLevel.sh tool 10-6
shared secret
configuration file utility and 5-27
DSS (DOCSIS Shared Secret)
about 2-10
DPEs and 2-10
resetting 2-10
single stack 6-15
SnifferPro, for troubleshooting 16-15
SNMP
agent
BAC architecture and 9-4, 10-9
configuring 10-10
MIB support 10-9
starting 10-12
stopping 10-13
cloning on RDU, DPE
configuring 7-29
PacketCable eMTA (use case) D-42
snmpAgentCfgUtil.sh tool 10-10
community, adding 10-11
community, deleting 10-12
hosts, adding 10-10
hosts, deleting 10-11
location, changing 10-13
settings, listing 10-14
SNMP contacts, setting up new 10-14
SNMP listening port, identifying 10-13
SNMP notification types, specifying 10-14
starting 10-12
stopping 10-13
TLVs
without MIB, adding 5-10
with vendor-specific MIB, adding 5-11
v3 cloning, configuring on RDU, DPE
key generation 7-29
key material 7-29
SNMP agent
See SNMP
snmpAgentCfgUtil.sh
adding agent community 10-11
adding host 10-10
changing agent location 10-13
configuring agent port 10-13
deleting agent community 10-12
setting up contacts 10-14
specifying notification type 10-14
viewing agent settings 10-14
SRV record in DNS server, configuring 7-28
startDiagnostics.sh 16-5
statusDiagnostics.sh 16-5
STB
configuring defaults 13-14
RNG-200 1-4, 4-1
stopDiagnostics.sh 16-5
supported standards 1-3
syslog alerts
See alert messages
system defaults, configuring 13-12

T

template files, developing 5-1
definition options, encoding types for 5-14
BITS value syntax 5-21
OCTETSTRING syntax 5-21
grammar 5-2
comments 5-2
include files 5-3
instance modifier 5-4
options 5-3
OUI modifier 5-5
macro variables 5-8
option support
CableHome B-22
DOCSIS B-1

PacketCable B-22
SNMP TLVs 5-10
vendor-specific MIBs, adding 5-11
without MIBs, adding 5-10
SNMP VarBind 5-6
CableHome MIBs 5-8
DOCSIS MIBs 5-7
PacketCable MIBs 5-7
tools 9-4
bprAgent, using 9-2
changeNRProperties.sh, using 14-11
configuration file utility (runCfgUtil.sh), using 5-21
diagnostics, using 16-5
bundleState.sh 16-10
startDiagnostics.sh 16-5
statusDiagnostics.sh 16-8
stopDiagnostics.sh 16-9
disk_monitor.sh, using 14-12
KeyGen, using 14-8
PKCert.sh, using 14-2
RDU log level, using 10-4
setLogLevel.sh, using 10-4
snmpAgentCfgUtil.sh, using 10-10
tools and advanced concepts
configuration file utility 5-21
binary file, external, viewing 5-33
binary file, local, viewing 5-32
binary file output, specifying 5-31
binary files, converting to template files 5-23
generating TLV 43s for multivendor support 5-36
macro variables, specifying a device for 5-30
macro variables, specifying through CLI 5-29
PacketCable Basic flow, activating 5-34
running 5-22
testing template processing, external files 5-26
testing template processing, local files 5-25
testing template processing, local files and adding shared secret 5-27
using tool 5-21
disk_monitor.sh tool 14-12
KeyGen tool 14-8
NRProperties.sh tool 14-8
PKCert.sh tool 14-2
KDC certificate, creating 14-3
KDC certificate, validating 14-4
running 14-2
setting log level for debug 14-5
RDU log level tool
current log level, viewing 10-6
setting 10-6
snmpAgentCfgUtil.sh tool 10-10
hosts, adding 10-10
hosts, deleting 10-11
SNMP agent, starting 10-12
SNMP agent, stopping 10-13
SNMP agent community, adding 10-11
SNMP agent community, deleting 10-12
SNMP agent location, changing 10-13
SNMP agent settings, listing 10-14
SNMP contacts, setting up new 10-14
SNMP listening port, identifying 10-13
SNMP notification types, specifying 10-14
template files, developing
comments 5-2
definition options, encoding types for 5-14
grammar (table) 5-2
includes 5-3
instance modifier 5-4
macro variables 5-8
options 5-3
option support B-1, B-22
OUI modifier 5-5
SNMP VarBind 5-6
troubleshooting devices by device ID 16-2
configuring 16-3
relating device to node 16-3
viewing devices 16-4

troubleshooting
alert messages A-1
DPE alerts A-3
message format A-1
RDU alerts A-2
watchdog alerts A-4
bundleState.sh, using 16-10
device diagnostics 16-2
relating device to node 16-3
devices, using device ID 16-2
configuring for troubleshooting 16-3
relating device to node 16-3
sample log output 16-4
viewing devices in diagnostics mode 16-4
diagnostics tool 16-5
bundleState.sh 16-5, 16-10
startDiagnostics.sh 16-5
statusDiagnostics.sh 16-8
stopDiagnostics.sh 16-9
eMTA provisioning for PacketCable
components 16-11
key variables 16-13
logs 16-14
scenarios 16-15
tools 16-14
server state, bundling for support 16-10
tools for PacketCable configuration 16-14
troubleshooting.log 16-2
troubleshooting PacketCable provisioning 16-11
components
call management server 16-12
DHCP server 16-12
DNS server 16-12
eMTA 16-11
KDC 16-12
server 16-12
key variables
certificates 16-13
MTA configuration file 16-14
scope-selection tag 16-14

U

uBr, definition 1-6
upgrading license keys 13-24
use cases
about D-1
adding
new computer behind a modem with NAT D-30
new computer in fixed standard mode D-8
second computer in promiscuous mode D-29
bulk provisioning modems in promiscuous mode D-23
CableHome with firewall configuration D-52
creating API client D-1
disabling subscriber D-10
getting detailed device information D-33
incremental provisioning of PacketCable eMTA D-44
logging
batch completions using events D-33
device deletions using events D-31
modifying an existing modem D-15
monitoring RDU connection using events D-32
moving device to another DHCP scope D-30
optimistic locking D-48
preprovisioning
CableHome WAN-MAN D-51
DOCSIS modems with configuration files D-46
first-time activation in promiscuous mode D-25
modems and self-provisioned computers D-12
PacketCable eMTA D-40
replacing existing modem D-27
retrieving capabilities for CableHome WAN-MAN D-54
searching for devices using class of service D-39
searching for devices using vendor prefix D-39
searching using default class of service D-38
self-provisioning

CableHome WAN-MAN D-55
first-time activation in promiscuous mode D-20
first-time activation with NAT D-29
modem, computer in fixed standard mode D-5
SNMP cloning on PacketCable eMTA D-42
subscriber bandwidth, temporarily throttling D-50
unregistering, deleting subscriber device D-16
user interface
See administrator user interface
users, managing 12-1
adding 12-2
deleting 12-3
modifying 12-3
types
administrator 12-1
read/write 12-2
read-only 12-2

V

vendor-specific MIBs, adding 5-11
verifyDb.sh tool 15-5
voice technology
See PacketCable 1-2

W

WAN-Data default, configuring 13-7
WAN-MAN default, configuring 13-7
watchdog alerts A-4