Configuring Prime Cable Provisioning Components

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

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

Configuring Regional Distribution Unit

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

The following table identifies the workflow to follow when configuring the RDU.

Table 1: RDU Configuration Workflow

	Task	**See...**
Step 1	Configure the system syslog service for use with Prime Cable Provisioning.	Configuring Syslog Utility to Receive Alerts
Step 2	Access the Prime Cable Provisioning Admin UI.	Administrator User Interface
Step 3	Change the login password.	Accessing Admin UI
Step 4	Add the license file.	License Keys for Prime Cable Provisioning
Step 5	Configure the RDU SNMP agent.	Using snmpAgentCfgUtil.sh Tool

	Task	See...
Step 6	Configure the default severity log level, which is the Notification level.	Using the RDU Log Level Tool
Step 7	Enable provisioning-group capabilities for IPv4 or IPv6.	Monitoring Provisioning Groups

Configuring RDU Extensions

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

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

The following table lists the RDU extension points that Prime Cable Provisioning requires to execute extensions.

Table 2: Required RDU Extension Points

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

Extension Point	Description	Use	Specific to Technology?
Publishing	Executed to publish provisioning data to an external datastore. The default extensions built into Prime Cable Provisioning do not include any publishing plug-ins.	Optional	No
Service-Level Selection	Executed to select the service level to grant to a device. Extensions attached to this extension point are executed before any common configuration generation extensions and the technology-specific configuration generation extensions.	Optional	Yes
Authentication	Executed to authenticate the user via remote authentication servers. Extensions will be attached to the extension points based on the authentication mode listed in RDU Defaults Page. Radius extensions are default built in authentication extensions in Prime Cable Provisioning.	Required	Yes

Managing extensions includes:

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

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

Writing a New Class

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

To write a new class:

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

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

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

For example:

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

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

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

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

For example:

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

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

Device Detection process

There are three phases in the device detection process:

1. The initial setup phase initializes various "house keeping" fields, fields that hold basic information about the device and items such as logging controls etc.
2. The second phase consists of gathering detection information. The device detection related information is extracted from the DHCPv4 and DHCPv6 configuration. DHCPv4 information such as class identifier, relay agent circuit id and remote id, vendor specific information and DHCPv6 information such as vendor class, vendor opts etc. are gathered .
3. The last phase looks at all the information gathered and attempts to determine the device type and whether there is another device in front of this device. If it can be determined, it will be set in the device detection context.

Installing RDU Custom Extension Points

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

Step 1 To add the new JAR file, see Adding Files.

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, on page 4, 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 Admin UI.

Step 2 Click **Submit**.

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

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

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

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

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

Viewing RDU Extensions

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

RDU Extension Dependencies on IPDeviceKeys Properties

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

Table 3: RDU Extension Dependencies on IPDeviceKeys Properties

Property	Scope	RDU Built-in Extension
EXTENDED_FILENAME_SCRIPT	DHCP	ConfigGeneration
DROP_IF_MAX_IA_ADDRESSES_EXCEEDED_ENABLE MAX_IA_ADDRESSES	DHCP	ConfigGeneration
MUST_BE_BEHIND_DEVICE MUST_BE_IN_PROV_GROUP MUST_BE_BEHIND_DEVICE_AUTO_ENABLE	DHCP	ServiceLevelSelection
USE_BOOT_FILE_OPTION_FOR_CONFIG_FILE USE_FILE_OPTION_FOR_CONFIG_FILE	DHCP	ConfigGeneration
USE_VALIDATE_CONTINUE_FOR_CABLELABS_SOFTWARE_VERSION_DHCPV4 USE_VALIDATE_CONTINUE_FOR_CABLELABS_SOFTWARE_VERSION_DHCPV6 USE_VALIDATE_CONTINUE_FOR_CLIENT_ID USE_VALIDATE_CONTINUE_FOR_CLIENT_ID_CREATED_FROM_MAC_ADDRESS USE_VALIDATE_CONTINUE_FOR_DHCP_CLASS_IDENTIFIER USE_VALIDATE_CONTINUE_FOR_DHCP_PARAMETER_REQUEST_LIST USE_VALIDATE_CONTINUE_FOR_VENDOR_CLASS USE_VALIDATE_PREFIX_FOR_DHCP_CLASS_IDENTIFIER USE_VALIDATE_CONTINUE_FOR_DHCP_CL_OPTION_IP_PREF_DHCPV6 USE_VALIDATE_CONTINUE_FOR_DHCP_CL_OPTION_IP_PREF_DHCPV4 USE_VALIDATE_CONTINUE_FOR_DHCP_CL_OPTION_ORO_DHCPV6 USE_VALIDATE_CONTINUE_FOR_DHCP_CL_OPTION_ORO_DHCPV4 USE_VALIDATE_CONTINUE_FOR_RELAY_AGENT_CIRCUIT_ID	DHCP	ConfigGeneration

Property	Scope	RDU Built-in Extension
VALIDATE_CABLELABS_SOFTWARE_VERSION_DHCPV4 VALIDATE_CABLELABS_SOFTWARE_VERSION_DHCPV6 VALIDATE_CLIENT_ID VALIDATE_CLIENT_ID_CREATED_FROM_MAC_ADDRESS VALIDATE_DHCP_CLASS_IDENTIFIER VALIDATE_DHCP_PARAMTER_REQUEST_LIST VALIDATE_VENDOR_CLASS VALIDATE_DHCP_CL_OPTION_IP_PREF_DHCPV6 VALIDATE_DHCP_CL_OPTION_IP_PREF_DHCPV4 VALIDATE_DHCP_CL_OPTION_ORO_DHCPV6 VALIDATE_DHCP_CL_OPTION_ORO_DHCPV4 VALIDATE_RELAY_AGENT_CIRCUIT_ID	DHCP	ConfigGeneration

Configuring SNMP Reset

Default SNMP Version for Cable Modem Reset

The Cable Modem reset is done using SNMP v1, by default, whereas users can configure to use the SNMP v2c / v3 for Cable Modem reset.

Note To configure the default SNMP version for DOCSIS modem reset:

- The API constant is: **ServerProperties.RDU_DOCSIS_RESET_SNMP_VERSION**
- Property name: **/rdu/docsis/reset/snmp/version**

This property contains the SNMP version, which can be used while sending the SNMP device reset messages for DOCSIS devices. The default value for this property is 0 (SNMP v1) and the applicable values are (0 = SNMP v1; 1 = SNMP v2c; 2 = SNMP v2c; 3 = SNMP v3).

This property can be configured in Admin UI too (System Defaults and Property hierarchy).

Configuring SNMPv3 Reset

Prime Cable Provisioning prior to 6.1.3 was providing SNMPv3 reset support restricted only for PacketCable secure provisioning flow.

Prime Cable Provisioning 6.1.3 supports SNMPv3 reset with / without DH Kickstart for the DOCSIS cable modem and behind devices (PacketCable and eRouter) disruption from the RDU.

The following table describes the properties that you can use to configure SNMPv3 reset operation.

Table 4: Properties for SNMPv3

Property Name	Description
/rdu/docsis/reset/snmp/version	Identifies the SNMP reset version
	API Constant `RDU_DOCSIS_RESET_SNMP_VERSION`
/snmpv3/dhKickstartEnabled	Identifies whether DH kickstart is enabled or disabled. Default value is disabled.
	API Constant `RDU_DOCSIS_RESET_DH_KICKSTART_ENABLED`
/snmpv3/usmSecurityName	Identifies the USM security name and is set to default value docsisOperator when DH kickstart is enabled
	API Constant `RDU_RESET_SNMP_V3_SECURITY_NAME`
/snmpv3/authProto	Identifies the USM authentication protocol value. The values accepted are – 0 (NoAuth) / 1 (MD5) / 2 (SHA)
	API Constant `RDU_RESET_SNMP_V3_AUTH_PROTO`
/snmpv3/privProto	Identifies the USM privacy protocol value. The values accepted are – 0 (NoPriv) / 1 (DES) / 2 (AES)
	API Constant `RDU_RESET_SNMP_V3_PRIV_PROTO`
/snmpv3/resetSecurityMode	Identifies the USM security level. The values accepted are - authPriv / authNoPriv / noAuthNoPriv
	API Constant `RDU_RESET_SECURITY_MODE`
/snmpv3/dhKickstartBase	Identifies the DH kickstart base value which is used in the generation of shared secret
	API Constant `RDU_DOCSIS_RESET_DH_KICKSTART_BASE`

Property Name	Description
/snmpv3/dhKickstartPrime	Identifies the DH kickstart prime value which is used in the generation of shared secret
	API Constant `RDU_DOCSIS_RESET_DH_KICKSTART_PRIME`
/snmpv3/dhKickstartAuthKeySalt	Identifies the DH kickstart auth USM salt key which is used in the generation of USM auth key
	API Constant `RDU_DOCSIS_RESET_DH_KICKSTART_AUTHKEY_SALT`
/snmpv3/dhKickstartPrivKeySalt	Identifies the DH kickstart USM priv salt key which is used in the generation of USM priv key
	API Constant `RDU_DOCSIS_RESET_DH_KICKSTART_PRIVKEY_SALT`
/snmpv3/dhKickstartQueryMaxSecurityNames	Identifies the count of maximum security name that can be allowed in DH kickstart
	API Constant `RDU_DOCSIS_RESET_DH_KICKSTART_QUERY_MAX_SECURITYNAMES`
/snmpv3/usmDHKickstartMgrPublic	Identifies the DH kickstart manager public random number
	API Constant `RDU_DOCSIS_RESET_DH_KICKSTART_MANAGER_PUBLIC_NUM`
/snmpv3/dhKickstartPrivateRandomNumber	Identifies the DH kickstart manager private random number
	API Constant `RDU_DOCSIS_RESET_DH_KICKSTART_MANAGER_PRIVATE_RANDOM_NUM`
/snmpv3/snmpVersionFallbackEnabled	Identifies the boolean value which enables fallback to SNMPv2 device reset when SNMPv3 device reset operation fails and is set to default value true when RDU_DOCSIS_RESET_DH_KICKSTART_ENABLED is enabled
	API Constant `RDU_RESET_SNMPV3_VERSION_FALLBACK_ENABLED`

Configuring Remote SNMP Reset

The default device SNMP reset (Activation) is done by RDU by using disruption extensions. The device disruption implementation sends a SNMP set message from RDU to the device.

Prime Cable Provisioning supports remote SNMP reset, in which the device SNMP reset request can be sent from DPE rather than RDU. During reset operation, RDU sends a reset request to DPE and in turn, DPE sends SNMP set to device.

You can enable or disable device SNMP reset through DPE feature from either the Admin UI or using the API. The PG capability to enable/disable this feature is:

- Capability name: **/provgroup/capability/dpe/remote/snmp/reset**
- The API constant is: **ProvGroupCapabilitiesKeys.REMOTE_SNMP_RESET**

Excluding DPEs from Reset Operation

When the SNMP Remote Reset feature is enabled for a PG, RDU will send the reset request to one of the available DPEs (based on MAC/DUID based affinity) in the PG during a device reset operation. However, user can set an exclusion list of DPEs, so that the RDU will not send the reset requests to those DPE(s).

The property to exclude specific DPEs at the PG level while sending remote reset is:

- Capability name: **/provgroup/capability/dpe/remote/snmp/reset/exclude/dpes/csv**
- The API constant is: **ProvGroupCapabilitiesKeys.REMOTE_RESET_EXCLUDE_DPES_CSV**

The property value is a comma-separated value (CSV) consisting of the hostnames of DPEs that needs to be excluded for remote SNMP reset.

Note Cisco Prime Cable Provisioning does not support SNMPv3 Reset in Remote SNMP Reset.

Configuring Provisioning Web Services

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

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

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

The provisioning service is described in RESTful web service using a set of resources that identify the targets of the interaction with its clients; identified by the URIs is a contract between the web service client and the RDU, and it describes the PWS RESTful operations.

Note Shared secret is not supported for the PWS component.

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

The following table identifies the workflow to follow when configuring the RDU.

Table 5: PWS Configuration Workflow

	Task	See...
Step 1	Configure the PWS to connect to the RDU.	Configuring RDU and User Details in PWS, on page 11
Step 2	To communicate with the client, you can either create a session or use the API.	Configuring Device Provisioning Engines, on page 13

Configuring RDU and User Details in PWS

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

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

To configure RDU details in PWS:

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

For example:

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

Run the same command to add additional RDUs.

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

For example:

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

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

For example:

RHEL/CentOS 6.x — `/etc/init.d/bprAgent restart <pws|restpws>`

RHEL/CentOS 7.x — `BPR_HOME/agent/bin/bprAgent restart <pws|restpws>`

Procedure to post SOAP messages through Provisioning Web Services

A web services client can get the WSDL (Web Services contract) file from the PWS by accessing the URL in the following format:

http://<PWS-HOST>:<PWS-PORT>/cp-ws-prov/provService?wsdl

Note

- PWS-HOST – The hostname (or) IP address of the server where PWS is installed
- PWS-PORT – Port through which PWS can be accessed. In the case of secure mode selection, the default PWS HTTPS port is 9443. For non-secure mode selection, the default PWS HTTP port is 9100.However, the user can configure a different (non-default) HTTPS / HTTP port during PWS installation.

For posting SOAP messages the following URL should be used and WSDL file is provided as input to load the defined PWS SOAP messages:

http://<PWS-HOST>:<PWS-PORT>/cp-ws-prov/provService

Note

- “http” protocol identifier is used as a citation in the URL formats referenced above.The protocol identifier https or http should be used accordingly for secure or non-secure mode of communication respectively.

Procedure to post RESTful messages through Provisioning Web Services

A web services client need to access the unique URI by accessing the URL in the following format:

http://<PWS-HOST>:<PWS-PORT>/cp-ws-rest-prov/<methodName>

Note

- PWS-HOST – The hostname (or) IP address of the server where PWS is installed
- PWS-PORT – Port through which PWS can be accessed. In the case of secure mode selection, the default PWS HTTPS port is 9790. For non-secure mode selection, the default PWS HTTP port is 9101.However, the user can configure a different (non-default) HTTPS / HTTP port during PWS installation.

For posting Restful messages the following URL should be used and request body attributes is provided as input to load the defined PWS Restful messages:

http://<PWS-HOST>:<PWS-PORT>/cp-ws-rest-prov/<methodName>

Note

- “http” protocol identifier is used as a citation in the URL formats referenced above. The protocol identifier https or http should be used accordingly for secure or non-secure mode of communication.

Configuring Device Provisioning Engines

You perform the tasks described in this workflow only after configuring the RDU which is described in Table 1: RDU Configuration Workflow, on page 1. You can configure the DPE to support, IPv4 and IPv6 as shown in the below tables.

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

The following table identifies the workflow to follow when configuring the DPE for IPv4.

Table 6: DPE Configuration Workflow for IPv4

	Task	See...
Step 1	Configure the system syslog service for use with Prime Cable Provisioning	Configuring Syslog Utility to Receive Alerts
Step 2	Change the password	The **password** command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Step 3	Configure the provisioning interface*	The **interface ip** *ip_address* **provisioning** command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Step 4	Configure the provisioning FQDN	The **interface ip** *ip_address* **provisioning fqdn** command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Step 5	Configure the interface that communicates with Cisco Prime Network Registrar extensions	The **interface ip** *ip_address* **pg-communication** command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Step 6	Configure the Prime Cable Provisioning shared secret*	The **dpe shared-secret** command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide

	Task	**See...**
Step 7	Configure the DPE to connect to the RDU *	The **dpe rdu-server** *{host | x.x.x.x} port secure* command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Step 8	Configure the Network Time Protocol (NTP)	Linux documentation for configuration information
Step 9	Enable TFTP	The **service tftp** *1..1* **ipv4 enabled true** command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Step 10	Enable ToD	The **service tod** *1..1* **ipv4 enabled true** command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Step 11	Configure the primary provisioning group*	The **dpe provisioning-group primary** command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Step 12	Configure the DPE SNMP agent	The SNMP agent commands in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Note You can configure the SNMP agent using either the DPE command-line interface or the **snmpAgentCfgUtil.sh** tool (see Using snmpAgentCfgUtil.sh Tool).		
Step 13	Verify that you are connected to RDU	Monitoring Servers Using Admin UI
Step 14	Reload the DPE	The **dpe reload** command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Step 15	Enable provisioning-group capabilities for IPv4	Monitoring Provisioning Groups

The following ***Table:15*** 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 the above ***Table 14: DPE Configuration Workflow for IPv4***, then additionally complete the steps described in this ***Table:15***.

Table 7: DPE Configuration Workflow for IPv6

	Task	See...
Step 1	Configure the provisioning interface.*	The **interface ip** *ipv6_address* **provisioning** command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Step 2	Configure the provisioning FQDN	The **interface ip** *ipv6_address* **provisioning FQDNs** command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Step 3	Configure the interface (IPv6) that communicates with Cisco Prime Network Registrar extensions	The **interface ip** *ipv6_address* **pg-communication** command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Note The Step 3 is optional. This step enables Network Registrar extensions to use IPv6 address to communicate with DPE. This step is not required for the Network Registrar extensions using IPv4 address to communicate with DPE.		
Step 4	Enable TFTP	The **service tftp** *1..1* **ipv6 enabled true** command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Step 5	Enable ToD	The **service tod** *1..1* **ipv6 enabled true** command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Step 6	Reload the DPE	The **dpe reload** command described in the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide
Step 7	Enable provisioning-group capabilities for IPv6	Monitoring Provisioning Groups

Configuring Cisco Prime Network Registrar

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

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

To configure Network Registrar for:

- DHCPv4, see Table 8: Network Registrar Workflow for DHCPv4, on page 16.
- DHCPv6, see Table 9: Network Registrar Workflow for DHCPv6, on page 17.

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

The following table identifies the workflow to follow when configuring Network Registrar for DHCPv4.

Table 8: Network Registrar Workflow for DHCPv4

	Task	**See...**
Step 1	Validate the Network Registrar extensions.	Cisco Prime Cable Provisioning 6.1.3 Quick Start Guide
Step 2	Configure the system syslog service for use withPrime Cable Provisioning.	Configuring Syslog Utility to Receive Alerts
Step 3	Configure client classes/selection tags that match those defined in the RDU. *	Cisco Prime Network Registrar End-User Guides
Step 4	Configure policies.*	Cisco Prime Network Registrar End-User Guides
Step 5	Configure scopes.*	Cisco Prime Network Registrar End-User Guides
Step 6	Back up the Network Registrar database.	Cisco Prime Network Registrar End-User Guides
Step 7	Verify that you are connected to the correct RDU.	Monitoring Servers Using Admin UI
Step 8	Reload the DHCP server.	Cisco Prime Network Registrar End-User Guides

The following table 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 9: Network Registrar Workflow for DHCPv6

	Task	Refer to...
Step 1	Validate the Network Registrar extensions.	Cisco Prime Cable Provisioning 6.1.3 Quick Start Guide
Step 2	Configure the system syslog service for use with Prime Cable Provisioning.	Configuring Syslog Utility to Receive Alerts
Step 3	Configure client classes/selection tags that match those defined in the RDU. *	Cisco Prime Network Registrar End-User Guides
Step 4	Configure policies.*	Cisco Prime Network Registrar End-User Guides
Step 5	Configure links.*	Cisco Prime Network Registrar End-User Guides
Step 6	Configure prefixes. For each prefix, ensure that you configure the appropriate policy, link, and selection tag.* **Note** Some DHCP clients, such as cable modems, reject Offers that contain multiple IPv6 addresses. While defining prefixes, configure Network Registrar such that it does not assign more than one IPv6 address to a client. Ensure that you do not add the same selection tag to two prefixes, because doing so makes Network Registrar pick one IP address from each prefix, thus assigning two IP addresses to the client.	Cisco Prime Network Registrar End-User Guides
Step 7	Back up the Network Registrar database.	Cisco Prime Network Registrar End-User Guides
Step 8	Verify that you are connected to the correct RDU.	Monitoring Servers Using Admin UI

	Task	Refer to...
Step 9	Reload the DHCP server.	Cisco Prime Network Registrar End-User Guides

Configuring Prime Network Registrar Extension

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

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

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

There are four types of dictionaries:

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

The dictionaries represent various DHCP options and settings as configured on Prime Cable Provisioning and Prime 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 Prime Network Registrar behavior. For example, “drop” to indicate that the packet is to be dropped.
- Those that provide information.

Prime Cable Provisioning, along with Prime Network Registrar 9.x, supports two API versions, each of which Prime Cable Provisioning extensions use to enable DHCPv4 or DHCPv6:

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

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

Properties that Control Data Discovered for DHCPv6

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

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

- Properties that control the behavior of Prime Network Registrar extensions in versions earlier than Prime Cable Provisioning, see Table 10: Properties for DHCPv4 Cisco Prime Network Registrar Extensions, on page 19.
- Properties that control the behavior of Prime Network Registrar extensions for DHCPv4 in Prime Cable Provisioning, see Table 10: Properties for DHCPv4 Cisco Prime Network Registrar Extensions, on page 19.
- Properties that control the behavior of Prime Network Registrar extensions for DHCPv6 in Prime Cable Provisioning for the client (cable modem) and the relay agent (CMTS). This distinction occurs because the DHCPv4 standard combines the client and relay message into one message, while the DHCPv6 standard splits them. See Table 11: Properties for DHCPv6 Cisco Prime Network Registrar Extensions , on page 20.

The following table describes the properties that influence the behavior of Prime Network Registrar extensions in Prime Cable Provisioning versions.

Table 10: Properties for DHCPv4 Cisco Prime Network Registrar Extensions

Property Name	Description
/cnrExtension/attributesToReadFrom EnvironmentDictionary	Identifies a list of attributes that must be pulled from the Prime Network Registrar environment dictionary
	API Constant `CNRExtensionSettingKeys.CNR_ATTRIBUTES_TO_READ_FROM_ENVIRONMENT_DICTIONARY`

Property Name	Description
/cnrExtension/attributesRequiredIn V4Request	Identifies a list of attributes that the Prime Network Registrar request dictionary must contain for extensions to submit a request for configuration generation to the RDU **API Constant** `CNRExtensionSettingKeys.CNR_ATTRIBUTES_REQUIRED_IN_V4_REQUEST_DICTIONARY`

/cnrExtension/attributesToPullFrom V4RequestAsBytes	Identifies a list of attributes to be pulled from the Prime Network Registrar request dictionary in binary format
	API Constant
	`CNRExtensionSettingKeys.CNR_ATTRIBUTES_TO_READ_FROM_V4_REQUEST_DICTIONARY_`
	`AS_BYTES`

/cnrExtension/attributesToPullFrom V4RequestAsStrings	Identifies a list of attributes to be pulled from the Prime Network Registrar request dictionary in string format
	API Constant
	`CNRExtensionSettingKeys.CNR_ATTRIBUTES_TO_READ_FROM_V4_REQUEST_DICTIONARY_`
	`AS_STRINGS`

The following table describes the properties that control the behavior of Prime Network Registrar extensions for DHCPv6 in Prime Cable Provisioning.

Table 11: Properties for DHCPv6 Cisco Prime Network Registrar Extensions

Property Name	Description
Client Message	
/cnrExtension/attributesRequiredIn V6Request	Identifies a list of attributes that the Prime Network Registrar DHCPv6 request dictionary must contain for extensions to submit a request for configuration generation to the RDU **API Constant** `CNRExtensionSettingKeys.CNR_ATTRIBUTES_REQUIRED_IN_V6_REQUEST_DICTIONARY`

/cnrExtension/attributesToPullFrom V6RequestAsBytes	Identifies a list of attributes to be pulled from the Prime Network Registrar DHCPv6 request dictionary in binary format
	API Constant
	`CNRExtensionSettingKeys.CNR_ATTRIBUTES_TO_READ_FROM_V6_REQUEST_DICTIONARY_`
	`AS_BYTES`

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

Property Name	Description
/cnrExtension/optionsToPullFrom V6RelayAsBytes	Identifies a list of DHCP options to be pulled from the Prime Network Registrar DHCPv6 Relay-Forward request Relay dictionary in binary format
	API Constant `CNRExtensionSettingKeys.CNR_OPTIONS_TO_READ_FROM_V6_RELAY_DICTIONARY_AS_BYTES`

Configuring Key Distribution Center

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

Default KDC Properties

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

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

The default properties are:

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

  For example:

  ```
  interface address = 10.10.10.1
  ```

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

  For example:

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

  Note You must enter the interface address and FQDN values through the KDC Realm Name screen during installation. For specific information, see the Cisco Prime Cable Provisioning 6.1.3 Quick Start Guide.

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

  For example:

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

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

  For example:

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

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

  ```
  log debug level = 5
  ```

  The following table describes the available logging levels for the KDC log file.

  Table 12: KDC Logging Levels

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

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

  For example:

  ```
  minimum ps backoff = 150
  ```

Using the sample values shown above, a sample INI file might contain data similar to that shown in the following example.

Sample kdc.ini Configuration File

```
interface address =  10.10.10.1
FQDN = kdc.example.com
maximum log file size = 1000
n saved log files = 10
```

```
log debug level = 5
minimum ps backoff = 150
maximum ps backoff = 300
```

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

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

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

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

  For example:

  ```
  maximum ticket duration = 168
  ```

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

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

  For example:

  ```
  minimum ticket duration = 144
  ```

KDC Certificates

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

Note Certificates are required for the KDC to function.

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

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

Installing KDC Licenses

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

To install a KDC license file:

Step 1 Obtain your license file from your Cisco representative.

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

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

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

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

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

Configuring Additional Realms

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

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

The following table lists the different certificates, with their corresponding filenames, that must be available in the *BPR_HOME/kdc/linux/packetcable/certificates* directory.

Table 13: 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/linux/packetcable/certificates* directory. Its private key (KDC_private_key.pkcs8) resides in the *BPR_HOME/kdc/linux/* 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.

The following table 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 14: Directory Structure for Multiple Realms

Directory	File Content in Directory
BPR_HOME/kdc/linux/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/linux/packetcable/certificates/CISCO1.COM	For secondary realm CISCO1.COM: • *KDC.cer* • *KDC private key*
BPR_HOME/kdc/linux/packetcable/certificates/CISCO2.COM	For secondary realm CISCO2.COM: • *KDC.cer* • *KDC private key*

Configuring the KDC for Multiple Realms

This section describes the workflow to configure the KDC for multiple realms. Before proceeding, complete the installation of the RDU, the DPE, and the Network Registrar extensions. For installation instructions, see the Cisco Prime Cable Provisioning 6.1.3 Quick Start Guide.

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

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

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

To configure the KDC for multiple realms:

Step 1 Verify the following configuration settings on the DPE:

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

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

For example:

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

For details on the commands, see the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide.

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

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

For example:

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

For details on the commands, see the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide.

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

For example:

```
dpe# show run
  aaa authentication radius
  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 about the commands, and the specific security privileges to run these commands, see the Cisco Prime Cable Provisioning 6.1.3 DPE CLI Reference Guide.

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

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

For example:

```
/opt/CSCObac/cnr_ep/conf# more cnr_ep.properties
#DO NOT MODIFY THIS FILE.
#Tue Aug 13 23:24:00 PDT 2013
/ccc/tgt=01
/cccv6/dssid/primary=ff\:ff\:ff\:ff
/secure/keystore/file=/opt/CSCObac/lib/security/.keystore
/ccc/dhcp/primary=10.81.90.90
/secure/keystore/password=f2c2060fdbca0e60ae1864adb73155b9
/lib/cpcp/ssllib=/opt/nwreg2/local/lib/libssl.so.1.0.1
/rdu/fqdn=bactst-lnx-4
/server/rdu/secure/enabled=true
/rdu/port=49188
/cnr/sharedSecret=fgL7egT9zcYHs
/ccc/kerb/realm=CISCO.COM
/provgroup/capability/both/packetcable/ipv6=enabled
/provgroup/capability/both/packetcable/ipv4=enabled
/lib/cpcp/cryptolib=/opt/nwreg2/local/lib/libcrypto.so.1.0.1
/ccc/dns/primary=10.81.90.90
/cccv6/dssid/secondary=ff\:ff\:ff\:ff
/cnr/sharedSecret/digest=a3\:1f\:32\:6e\:57\:ed\:83\:b7\:68\:42\:f3\:31\:2b\:47\:d3\:36\:eb\:85\:93\:98
/cache/provGroupList=default
[root@bactst-lnx-7 ~]#
```

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

Step 4 Create DNS realm zones for the DNS server that is listed in the *cnr_ep.properties* file. You can add zones using the Network Registrar Admin UI 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 Admin UI, see the Cisco Prime Network Registrar End-User Guides.

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/linux/packetcable/certificates*.

For example:

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

For more information on creating directories, see Linux documentation.

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

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

For example:

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

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

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

d) Copy the following certificates to the *BPR_HOME/kdc/linux/packetcable/certificates* directory:

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

For information on copying files, see Linux 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/linux/
packetcable/certificates)
Copy KDC_private_key.pkcs8 to the KDC platform directory (i.e. /opt/CSCObac/
kdc/linux)
Copy KDC_private_key_proprietary. to the KDC platform directory (i.e. /opt/CSCObac/
kdc/linux)
```

For more information on the tool, see Using PKCert.sh.

f) Copy the KDC.cer file to the KDC certificate directory (*BPR_HOME/kdc/linux/ packetcable/certificates)*. For information on copying files, see Linux documentation on the **cp** command.

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

h) Copy the private key KDC_private_key_proprietary. to the KDC platform directory (*BPR_HOME/ kdc/linux*). For information on copying files, see Linux 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/linux/
packetcable/certificates)
Copy KDC_private_key.pkcs8 to the KDC platform directory (i.e. /opt/CSCObac/
kdc/linux)
Copy KDC_private_key_proprietary. to the KDC platform directory (i.e. /opt/CSCObac/
kdc/linux)
```

For more information on the tool, see Using PKCert.sh.

j) Copy *KDC.cer* to the secondary realm directory; for example, the */CISCO1.COM* directory under *BPR_HOME/kdc/linux/packetcable/certificates*. For information on copying files, see Linux 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/linux/packetcable/certificates*. For information on copying files, see Linux 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/linux/packetcable/certificates*. For information on copying files, see Linux 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/linux/
packetcable/certificates)
Copy KDC_private_key.pkcs8 to the KDC platform directory (i.e. /opt/CSCObac/
kdc/linux)
Copy KDC_private_key_proprietary. to the KDC platform directory (i.e. /opt/CSCObac/
kdc/linux)
```

For information on the tool, see Using PKCert.sh.

n) Copy *KDC.cer* to the secondary realm directory; for example, the */CISCO2.COM* directory under *BPR_HOME/kdc/linux/packetcable/certificates*. For information on copying files, see Linux 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/linux/packetcable/certificates*. For information on copying files, see Linux 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/linux/packetcable/certificates*. For information on copying files, see Linux documentation on the **cp** command.

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

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

For example:

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

For details, see Using PKCert.sh.

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

For example:

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

For more information, see Linux 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  linux  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/linux/logs
# ls
kdc.log    kdc.log.1

# cd /opt/CSCObac/linux
# 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 ./linux/packetcable/certificates
# ls
KDC.cer
Local_System.cer
CableLabs_Service_Provider_Root.cer  MTA_Root.cer
CISCO1.COM                          Service_Provider.cer
CISCO2.COM

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

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

For more information, see Linux documentation.

Step 9 Restart the KDC.

For example:

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

For more information, see Using Prime Cable Provisioning Process Watchdog from CLI.

Step 10 Configure the Prime Cable Provisioning Admin UI 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.

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.

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

   For example:

   1. Choose **Configuration** > **Files**. The Manage Files page appears.
   2. Click **Add**, and the Add Files page appears.
   3. Select the CableLabs Configuration Template option from the File Type drop-down list.
   4. Add the *mot-mta.tmp*l file. This file is the template used to provision a Motorola MTA. For template syntax, see the example, **Template Used to Provision a Motorola MTA.**
   5. Click **Submit**.

   For more information, see Managing Files.

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

   For example:

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

   For more information, see Managing Files.

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

   For example:

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

   For more information, see Managing Files.

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.

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.

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.

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

Example 1

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

a) Provision the cable modem part of the device by setting it to use the **sample-bronze-docsis** Class of Service.
b) To provision the MTA part, go to the **Devices** > **Manage Devices** page. Search and select the PacketCable device you want to provision. The Modify Device page appears.
c) Set the domain name. This example uses bacclab.cisco.com.

d) From the drop-down list corresponding to Registered Class of Service, select **mot-mta**. This is the Class of Service that you added in Step 10-f.
e) From the drop-down list corresponding to Registered DHCP Criteria, select the **default** option.
f) Click **Submit**.

Example 2

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

a) Provision the cable modem part of the device by setting it to use the **sample-bronze-docsis** Class of Service.
b) To provision the MTA part, go to the **Devices > Manage Devices** page. Search and select the PacketCable device you want to provision. The Modify Device page appears.
c) Set the domain name. This example uses bacclab.cisco.com.
d) From the drop-down list corresponding to Registered Class of Service, select **linksys-mta**. This is the Class of Service that you added in Step 10-g.
e) From the drop-down list corresponding to Registered DHCP Criteria, select the **cisco1** option. This is the DHCP Criteria that you added for the secondary CISCO1.COM realm in Step 10-a.
f) Click **Submit**.

Example 3

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

a) Provision the cable modem part of the device by setting it to use the **sample-bronze-docsis** Class of Service.
b) To provision the MTA part, go to the **Devices > Manage Devices** page. Search and select the PacketCable device you want to provision. The Modify Device page appears.
c) Set the domain name. This example uses bacclab.cisco.com.
d) From the drop-down list corresponding to Registered Class of Service, select **sa-mta**. This is the Class of Service that you added in Step 10-h.
e) From the drop-down list corresponding to Registered DHCP Criteria, select the **cisco2** option. This is the DHCP Criteria that you added for the secondary CISCO2.COM realm in Step 10-b.
f) Click **Submit**.

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

Example 1

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

KDC Log Sample Output–Motorola MTA

```
INFO [Thread-4] 2007-02-07 07:56:21,133 (DHHelper.java:114) - Time to create DH key pair(ms): 48
 INFO [Thread-4] 2007-02-07 07:56:21,229 (DHHelper.java:114) - Time to create DH key pair(ms): 49
 INFO [Thread-4] 2007-02-07 07:56:21,287 (DHHelper.java:150) - Time to create shared secret: 57 ms.

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

DPE Log Sample Output–Motorola MTA

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

Example 2

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

KDC Log Sample Output–Linksys MTA

```
 INFO [Thread-8] 2007-02-07 08:00:10,664 (DHHelper.java:114) - Time to create DH key pair(ms): 49
 INFO [Thread-8] 2007-02-07 08:00:10,759 (DHHelper.java:114) - Time to create DH key pair(ms): 49
 INFO [Thread-8] 2007-02-07 08:00:10,817 (DHHelper.java:150) - Time to create shared secret: 57 ms.

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

DPE Log Sample Output–Linksys MTA

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

Example 3

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

KDC Log Sample Output–SA MTA

```
 INFO [Thread-6] 2007-02-07 08:01:31,556 (DHHelper.java:114) - Time to create DH key pair(ms): 49
 INFO [Thread-6] 2007-02-07 08:01:31,652 (DHHelper.java:114) - Time to create DH key pair(ms): 50
 INFO [Thread-6] 2007-02-07 08:01:31,711 (DHHelper.java:150) - Time to create shared secret: 57 ms.

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

DPE Log Sample Output–SA MTA

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

Authoring Template for Provisioning Devices in Multiple Realms

You can use the template syntax described here to provision devices in a particular realm. The examples shown here are specific to the Motorola SBV5120 MTA , the Linksys CM2P2 MTA , and the SA WebStar DPX2203 MTA . The respective templates used to Provision are shown below.

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

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,STRING,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,"CableLabs,
 Inc."
#
# Set the realm name and IPSec control for the CMS.
#
option 11
.pktcMtaDevCmsTable.pktcMtaDevCmsEntry.pktcMtaDevCmsIpsecCtrl.'CMS.CISCO.COM',INTEGER,true
option 11
pktcMtaDevCmsTable.pktcMtaDevCmsEntry.pktcMtaDevCmsKerbRealmName.'CMS.CISCO.COM',STRING,CISCO.COM
#
# Finally, the end marker.
#
option 254 255
```

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,STRING,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,"CableLabs,
 Inc."
#
# Set the realm name and IPSec control for the CMS.
#
option 11
.pktcMtaDevCmsTable.pktcMtaDevCmsEntry.pktcMtaDevCmsIpsecCtrl.'CMS.CISCO.COM',INTEGER,true
option 11
pktcMtaDevCmsTable.pktcMtaDevCmsEntry.pktcMtaDevCmsKerbRealmName.'CMS.CISCO.COM',STRING,CISCO1.COM
#
# Finally, the end marker.
#
option 254 255
```

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,STRING,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,"CableLabs,
 Inc."
#
# Set the realm name and IPSec control for the CMS.
#
option 11
.pktcMtaDevCmsTable.pktcMtaDevCmsEntry.pktcMtaDevCmsIpsecCtrl.'CMS.CISCO.COM',INTEGER,true
option 11
pktcMtaDevCmsTable.pktcMtaDevCmsEntry.pktcMtaDevCmsKerbRealmName.'CMS.CISCO.COM',STRING,CISCO2.COM
#
```

```
# Finally, the end marker.
#
option 254 255
```