Getting Started

This chapter describes how to begin service provisioning with the Cisco Prime Provisioning application program interface (API) and includes API-related installation notes, verifying the status of the API, and the typical work flow steps.

This chapter contains the following sections:

Installation Notes

All components needed for the API are included with the installation of Prime Provisioning. The API servlet has no additional startup or shutdown requirements than normal Prime Provisioning, as the API servlet is embedded in the Tomcat engine. The API is a common component and is aware of blade-specific data in any given installation.

The end-user interface for the Prime Provisioning API interface is XML-encoded messages. A java client library and sample messages are shipped as part of the Prime Provisioning product.

See the *Cisco Prime Provisioning 7.0 Installation Guide* for more information.

Checking the API Status

You can verify the status of the API from the API itself, or from the GUI.

The GUI and the API are both on the same port, 8030 (port=8443 for HTTPS). Both clients run from the Tomcat server. Normally, if the GUI client is started, the API client is also started.

Checking from the API

To view the status of the API, you can:

- View the status of the watchdog client. Enter the following command:

  ```
  prime.sh status
  ```

The following is an example output for the **prime.sh status** command:

```
$ prime.sh status

Status on machine : valmont.cisco.com
Name                 State           Gen    Exec Time                 Success  Missed
nspoller             started         1      Oct 14 09:56:32 MDT       72       0
dbpoller             started         1      Oct 14 09:56:32 MDT       73       0
httpd                started         1      Oct 14 09:56:37 MDT       71       0
rgserver             started         1      Oct 14 09:56:57 MDT       72       0
cnsserver            started         1      Oct 14 09:56:37 MDT       73       0
discovery            started         1      Oct 14 09:56:32 MDT       72       0
```

The httpd is the Tomcat server for the GUI and API. If the API is running, the httpd server status is *started*.

- Attempt to log into the API.

  ```
  runNbi x $PRIMEF_HOME/resources/nbi/xml/examples/Session/Login.xml
  ```

  - If the API is not running, you receive a message:

    ```
    Failed to start client, Connection Refused.
    ```

  - If the API is running, you receive a session token.

Checking from the GUI

To check the status of the API from the GUI:

Step 1 From the Administration tab, choose **Control Center > Hosts**.

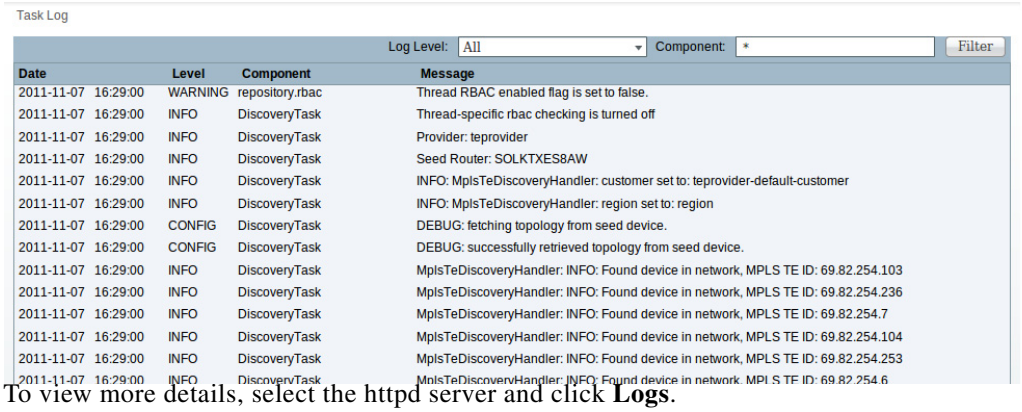
Step 2 Choose your server.

Step 3 Click **Servers**.

Step 4 Check the state of the httpd server. The httpd server is the http Tomcat server for the API client. If the API is running, it is in the *Started* state.

Figure 2-1 shows the status of the httpd server.

Figure 2-1 Server Status Window in the GUI



Step 5 To view more details, select the httpd server and click **Logs**.

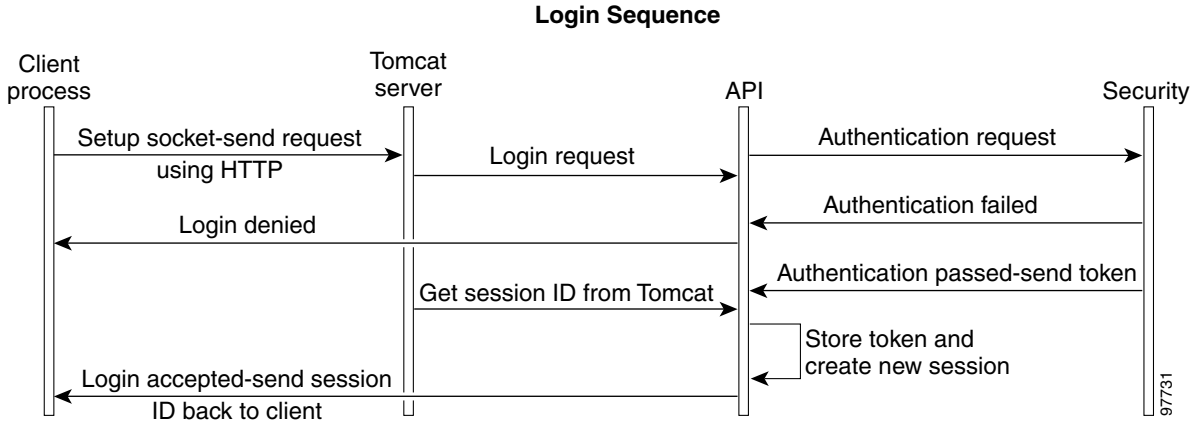
Logging In

Prime Provisioning uses Cisco role-based access control (RBAC) for user login and logoff. These user roles and permissions are set up using the GUI.

When a user sends a login request, the login is validated against the RBAC processor. If the validation is successful, an RBAC security token (session token) is maintained internally and also returned in the XML response as the **SessionId**. Each subsequent request requires the security token.
Prime Provisioning uses this security token to restrict access to Prime Provisioning data. The session token is generated by Prime Provisioning from the Tomcat session ID.

Figure 2-2 shows the user login sequence.

Figure 2-2 User Login Sequence



The following is an example of an XML response to a **createSession** (Login) operation. The security token is indicated in **bold**.

```
<soapenv:Body>
    <ns1:createSessionResponse>
      <returns xsi:type="ns1:CIMPropertyList" soapenc:arrayType="ns1:CIMProperty[]">
        <item xsi:type="ns1:CIMProperty">
          <name xsi:type="xsd:string">SessionId</name>
          <value xsi:type="xsd:string">F322D1A95424C095B58083C5C3A45633</value>
        </item>
      </returns>
    </ns1:createSessionResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

The security token in the XML response must be used in the header message for all subsequent requests to the server.

Work Flow

This section describes the typical order of operations when using the API for service provisioning. The fundamental steps include:

- Populating the Repository, page 2-4

- Collecting Device Configurations, page 2-10
- Creating the Service Policy, page 2-11
- Creating the Service Order/Service Request, page 2-11
- Performing a Configuration Audit, page 2-12

Most end-to-end provisioning services require most or all of these operations.

Populating the Repository

Every network element that Prime Provisioning manages must be defined as a device in the system. An element is any device that Prime Provisioning can collect configuration information from. Additionally, you must define device roles, groups, resource pools, and the topology of the network. Each of these items represents inventory in the Prime Provisioning database.

Prime Provisioning offers several options for populating the inventory in the repository.

- Inventory Manager—The Inventory Manager (IM) is a stand-alone Java Application on a client machine that allows you to import and administer network-specific data into the Prime Provisioning database. You must access the IM from the Prime Provisioning GUI. Use the IM for:
  - Autodiscovery—A IM feature that automatically discovers the state of the service provider's network, including physical connections, encapsulation types, and routing information.
  - Importing configuration files
- GUI—Use the Inventory and Connection Manager in the GUI to add each object.

  For more information on IM and the Prime Provisioning GUI, see the *Cisco Prime Provisioning 7.0 User Guide*.
- API—Use the API to create each object and then collect the configurations from each device or device group.

Note Inventory Manager and Autodiscovery do not apply to Traffic Engineering (TE) devices. A special TE Discovery tool is available for discovering TE devices in the core of the network. An example of how to use it from an API perspective is found in the TEM Provisioning Example, page 10-24. More detailed information about the TE Discovery tool is found in the chapter on TE Network Discovery in the *Cisco Prime Provisioning 7.0 User Guide.*

This section describes the basic steps to populate the Prime Provisioning repository using the API.

Note Use the XML example CreateInventory.xml for populating the database in bulk.

Populating the repository typically includes the following operations:

- Creating the inventory
- Defining provider and customer relationships
- Assigning route aggregation
- Defining access domains and VPNs
- Creating the physical topology.

Creating Inventory

To provision most Prime Provisioning services, the following steps for creating inventory are required:

- Create devices
- Create device groups
- Create providers and regions
- Assign PE devices to access domains
- Create customers and sites, and assign CPEs to them.

See the "Inventory" section on page 3-1 for a complete list of inventory items that can be created using Prime Provisioning.

Defining the Provider and Customer Relationship

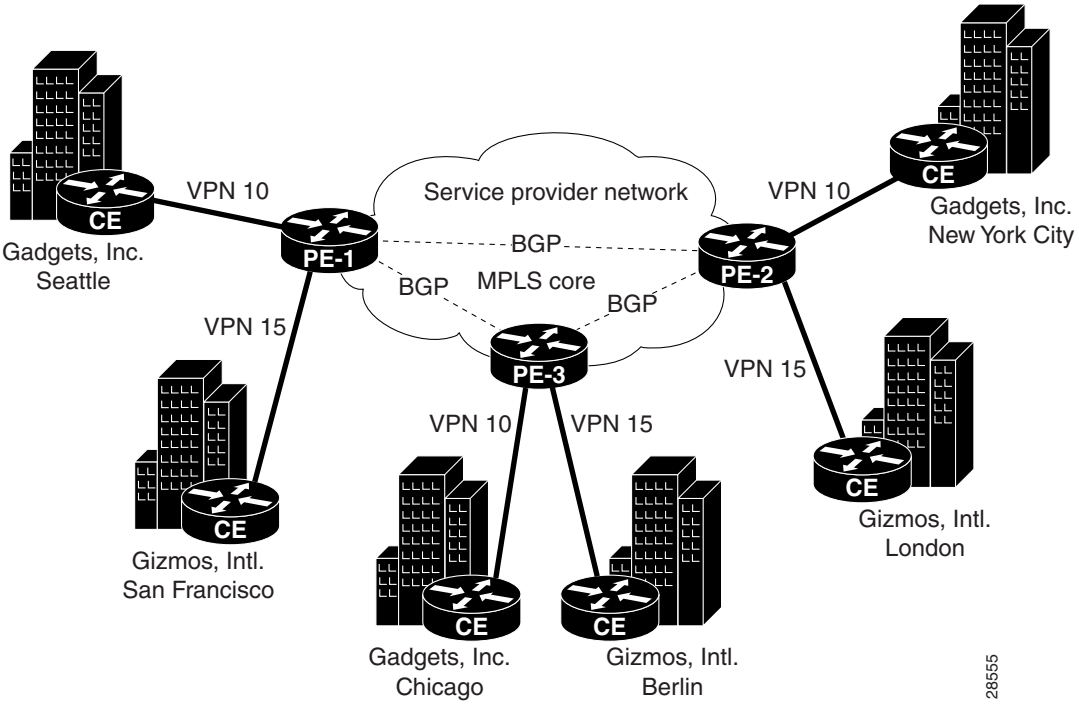
A service provider network architecture contains access routers, distributed routers, and core routers or ATM switches. Access routers terminate customer connections at the edge of the network.
Prime Provisioning provisioning is configured on the access circuit that involves the access router (provider edge devices or PEs) in the service provider network, and the customer premise equipment (CPE) in the customer network.

Provider

The provider administrative domain is the administrative domain of an ISP with one BGP autonomous system (AS) number. The network owned by the provider administrative domain (PAD) is called the backbone network. If an ISP has two AS numbers, you must define it as two PADs.

A service provider supplies VPN services to multiple customers. Figure 2-3 shows two different customers, each with a single VPN.

Figure 2-3 Provider View of the Network



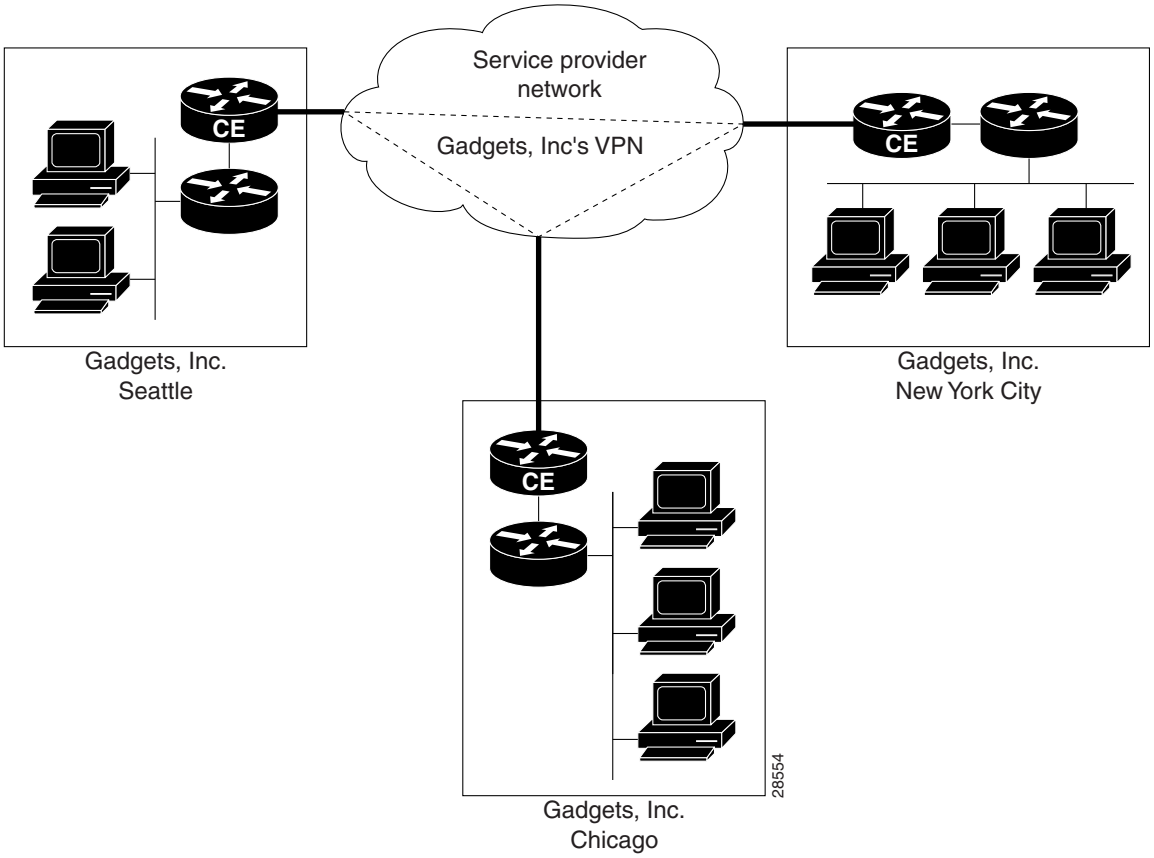
To create a **Provider** object, you need:

- Provider name (**Name**)
- Autonomous system number (**AsNumber**).

Customer

Customers have internal routers that communicate with their own customer edge devices (CEs) from one site to another through a VPN, which is managed by the service provider. See Figure 2-4.

Figure 2-4 Customer View of the Network



By using VPNs, the customers experience direct communication to each site as though they had their own private network, even though their traffic is traversing a public network.

To create a customer object, you need a customer (**Organization**).

Region

A region is group of PE devices within a single BGP AS or PAD. Each provider can contain multiple region objects.

To create a Region object, you need:

- Region name (**Name**)
- Provider domain for this region (**Provider**).

Site

A customer site is a set of IP systems with mutual IP connectivity between them, without the use of a VPN. A customer site can belong to only one customer, and can contain one or more CPEs, for load balancing.

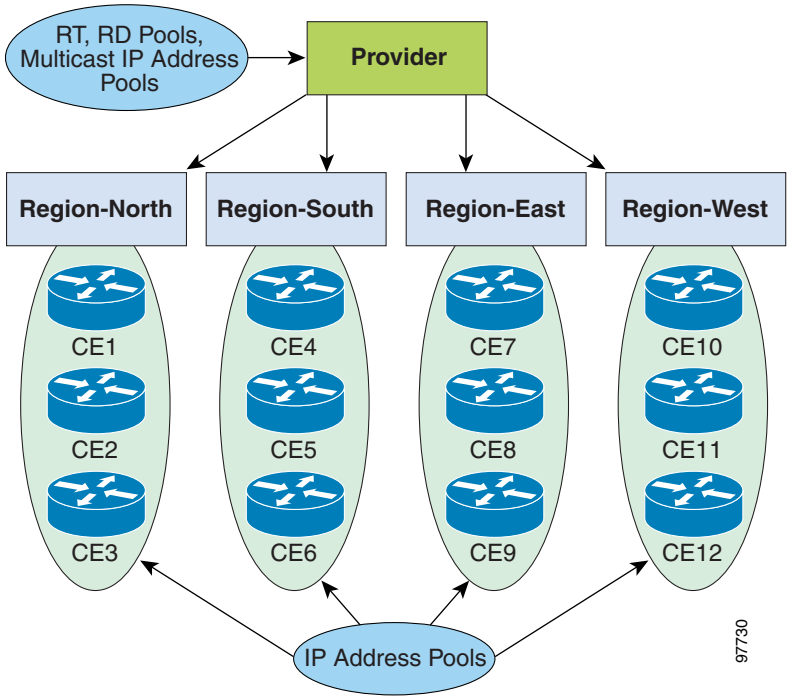
To create a **Site** object, you need:

- Site name (**Name**)
- Customer for this site (**Organization**).

Assigning Route Aggregation

Resource pools are assigned per provider and are used for route aggregation within regions. Resource pools include the route distinguisher (RD), route target (RT), and address pools. See Figure 2-5.

Figure 2-5 Resource Pools



Resource pools allow you to manage various pool types, and to associate a pool with any service model object in the Prime Provisioning repository. Pools help to automate service deployment.

See the "Resource Pools" section on page 3-3 for a complete list of the supported resource pools.

Defining Access Domains and VPNs

Access domains characterize the physical relationship of the devices between the provider network and the CE devices. VPNs characterize the routing relationship between the CEs through the provider network.

Access Domains

An access domain is the Layer 2 Ethernet switching domain that is attached to the PE. All switches attached to the PE-POP belong to the same access domain. You can associate multiple VLAN pools to a single access domain. You can assign two PEs to an access domain for redundancy.

To create an **AccessDomain** object, you need:

- Provider name (**Provider**)
- The VLAN pool reserved for this domain (**ReservedVlanPool**)—The range of the VLAN pool is specified with **Start** and **Size**.

Virtual Private Networks

A virtual private network (VPN) is a collection of customer sites that share the same routing table. A VPN can consist of sites that are all from the same enterprise (intranet), or from different enterprises (extranet). VPNs can consist of sites that all attach to the same service provider backbone or to different service provider backbones.

The path between two sites in a VPN, and the characteristics of that path, can also be determined (in whole or in part) by the service policy (service definition). See Figure 2-4 for an example of a VPN between customer sites.

To create a VPN, you need:

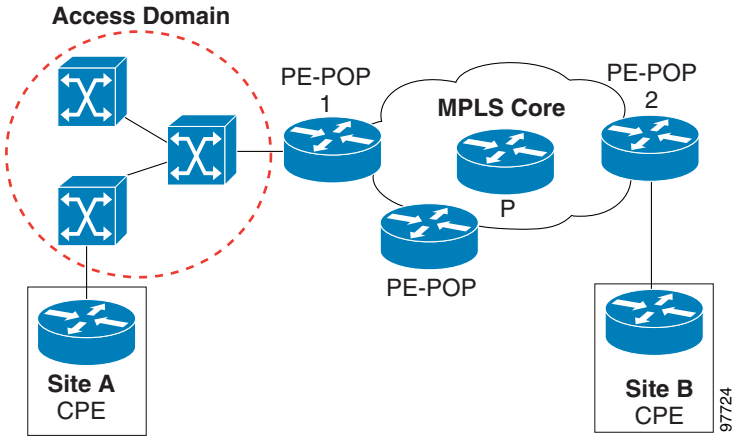
- VPN name (**Name**)
- Customer (**Organization**)
- CE routing community (**CERC**)—For MPLS VPNs only.

Creating Physical Topology

If you use a manual method for populating the repository (any method except Autodiscovery), you must also create the physical network topology. Named physical circuits (NPCs) are the physical connections between network elements and their PEs. You must create a connection from each network device in an access domain. NPCs are required for L2VPN and MPLS provisioning.

Figure 2-6 shows a network example with a distributed PE scenario. The CPE is connected to the PE through an access domain, which contains a layer 2 switching environment.

Figure 2-6 Sample L2VPN Network



To create the NPCs for this network, first specify the link from the CPE at Site A to the PE-POP 1, and then one physical link for each intermediate switch between the CPE and PE-POP.

For each **NPC** object, you need:

- Source device (**SrcDevice**)
- Destination device (**DestDevice**)
- Source device interface name (**SrcIfName**)
- Destination device interface name (**DestIfName**).

See the "Topology" section on page 3-4 for more information on the APIs available for defining network topology.

Collecting Device Configurations

A device configuration collection is a task. This task uploads the current configuration from the device to the Prime Provisioning database. The collection task is executed through a service request or a service order.

The following information is required for each device that you collect configurations from:

- General device information
- Login and Password information
- Device and configuration access information:
  - Device access protocol (Telnet, SSH, CNS)
  - SNMP Version
  - SNMP V1/V2/V3 information
  - Terminal Server Information.

Collection tasks can be executed immediately or they can be scheduled. See the "Collection" section on page 3-10 for more information.

Creating the Service Policy

A service policy is defined in a service definition. The service definition defines the service policy and policy characteristics. You can set an additional attribute (**editable=true**) for policy properties in the service definition to allow the service request creator to override the policy attributes. Service orders and service requests use service definitions to define common data to be used during the provisioning process.

Use service definitions to create configuration parameters that can be used by multiple services. Prime Provisioning supports service policies for each of the service types (MPLS, L2VPN).

To create a service definition for most services, you need:

- Service definition name (**Name**)
- Service definition type (**Mpls**, **L2Vpn**)
- Service definition details (**ServiceDefinitionDetails**)—The service definitions details specify the policy information.

See the appropriate chapter on provisioning for more information on creating service policies for:

- Template Service Definitions
- MPLS Service Definitions
- L2VPN Service Definitions.

Creating the Service Order/Service Request

Service orders allow you to schedule a provisioning process and capture its history. Service orders are used to implement service requests. It is the service request that is provisioned and activated in the network. The service request defines attributes for the physical links and specifies the service policy (defined in a service definition) to use.

Use service orders to:

- Specify and implement one or more service requests, for batch operations
- Modify an existing service request
- Specify the order of implementation for service requests
- Initiate a task.

Service request deployment is scheduled based on the service order or service request due date. If the service order or any service requests within the service order has a due date in the future, it is placed in a schedule queue.

To schedule a service order, you need:

- Service order name (**ServiceName**)
- The number of requests (**NumberOfRequests**)
- The service request to implement (**ServiceRequest**)—The service request details specify the attribute and service policy information. You can specify one or more service requests.

Service Order Response

The XML response to a service order contains its own **LocatorId** (required) and **ServiceName** (optional). If the service order is created to implement a service request, the service order response also contains the response to the service request, which includes the **LocatorId** (required) and the **ServiceRequestName** (optional).

If the service order is created to initiate a task (collect configurations, perform functional audits), the service order XML response also contains data. Use the Locator ID for subsequent requests relating to the service order or service request.

You can view any service order record using the **LocatorId** attribute. The response to a view request contains a **TaskLocatorId** attribute, which is used to track the status of any task created in conjunction with the service order. The **TaskLocatorId** attribute also allows you to view the task logs. See the "Viewing Task Logs" section on page 5-24 for more information.

See the appropriate chapter on provisioning for more information on creating service orders.

Performing a Configuration Audit

You can perform a configuration audit as part of a service request deployment or as a separate operation. During a configuration audit, Prime Provisioning verifies that all Cisco IOS commands are present and that they have the correct syntax. A configuration audit also verifies that there were no errors during service deployment.

A configuration audit is considered a task in the API and is executed with a service order. To schedule a configuration audit you need to specify:

- The type of audit you want to perform in the service request details (**SubType=CONFIG_AUDIT**)
- Locator ID of the service request that was used to deploy the configuration to audit (**LocatorId**).

Note If the configuration audit is part of a service request deployment, you do not need to perform an audit as a separate operation.

Configuration Audits are described in the "Tasks" section on page 3-9.