CHAPTER 7

Use Cases

This chapter describes the most common Cisco Broadband Access Center (Cisco BAC) API use cases. These use cases are directly related to device provisioning and device management provisioning.

Many system configuration and management operations, such as managing Class of Service, DHCP Criteria, and licenses, are not addressed here because these operations do not require integration with BSS and OSS. You can also use the Cisco BAC administrator user interface to perform most of these activities. See the *Cisco Broadband Access Center Administrator's Guide 4.2,* for details.

For more details on related API calls and sample API client code segments explaining individual API calls and features, refer to these resources that are available in the Cisco BAC installation directory:

- API Javadocs, located at *BAC_42_SolarisK9/docs/BAC_Javadoc_API_Provisioning*.
- Sample API client code, located at *BPR_HOME/rdu/samples/provapi*.

BPR_HOME is the home directory in which you install Cisco BAC. The default home directory is */opt/CSCObac*.

This chapter lists various API constants and their functions. To execute any API, you must follow the steps described in the Getting Started with the BAC API chapter.

Provisioning Operations

This section describes the following provisioning operation use cases:

Note The classfiles referenced in these use cases; for example, the *AddDeviceExample.java* classfile that illustrates how you can add a device record to the RDU, are only samples that are bundled with the Cisco BAC software.

- Adding a device record to the RDU—See Table 7-1.
- Modifying a device record in the RDU—See Table 7-2.
- Retrieving discovered device data from the RDU—See Table 7-3.
- Deleting device from the RDU—See Table 7-4.
- Retrieve Devices Matching Vendor Prefix—See Table 7-5.

Table 7-1 Adding a Device Record to the RDU

Classfile	API
`AddDeviceExample.java`	IPDevice.add()
Adds a new device record to the RDU database. Uses the IPDevice.add() API and submits the batch synchronously with the NO_ACTIVATION flag. This operation causes the RDU to generate instructions for the device, which are then cached in the DPE. The Figure 7-1 explains adding/modifying a device record in the RDU with Activation mode = No_ACTIVATION.	

Figure 7-1 Change Device Class of Service (Activation mode= NO_ACTIVATION)

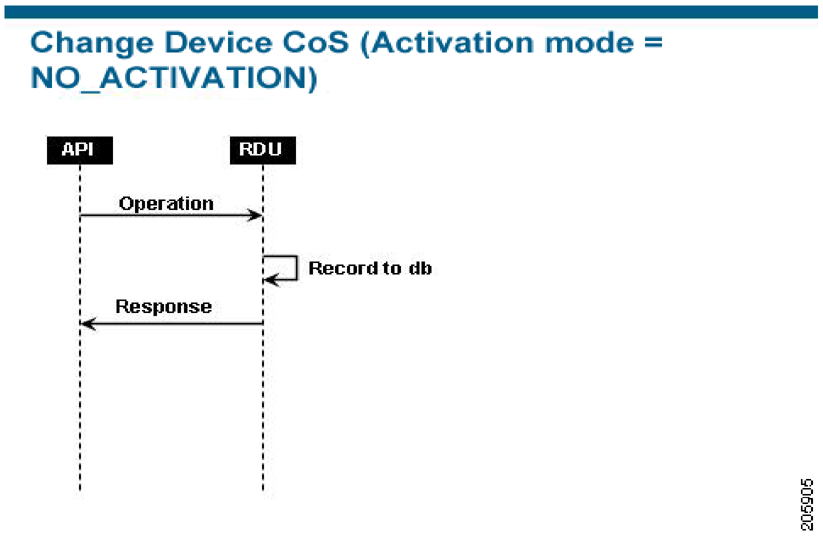

Table 7-2 Modifying a Device Record in the RDU

Classfile	API
`ModifyDeviceExample.java`	IPDevice.changeProperties()
Changes the properties of a device record stored in the RDU. Uses the IPDevice.changeProperties() API and submits the batch synchronously with the NO_ACTIVATION flag. This operation causes the RDU to generate instructions for the device, which are then cached in the DPE.	

Table 7-3 Retrieving Discovered Device Data in the RDU

Classfile	API
`QueryDeviceExample.java`	IPDevice.getDetails()
Retrieves the discovered data of a device that is stored in the RDU. Uses the IPDevice.getDetails() API and submits the batches synchronously using the on-connect mode with the NO_ACTIVATION flag.	

Table 7-4 Delete Device from the RDU

Classfile	API
`DeleteDeviceExample.java`	IPDevice.delete()
Deletes a device from the RDU. Uses the IPDevice.delete() API and submits the batch synchronously with the NO_ACTIVATION flag.	

Table 7-5 Retrieve Devices Matching Vendor Prefix

Classfile	API
`RetriveDevicesMatchingVendorPrefix.java`	IPDevice.searchDevice()
Searches for devices that exist in the database. Uses the IPDevice.searchDevice() API to query a list of devices that exist in the database.	