Provisioning API Use Cases

To help you understand how and where to use the provisioning API, this chapter presents a series of the most common provisioning API use cases, including code segments that you can use as models for your code. The code segments are embedded in typical service provider workflows.

Note This chapter uses pseudocode to illustrate the code segments. This pseudocode resembles Java code, but is *not* intended for compilation.

A Quick Reference to the Use Cases

This section identifies all uses cases found in this guide.

DSTBs (new or existing DSTBs)

- Preprovisioned First Time Activation for DSTB, page 2-17
- Modify an Existing DSTB, page 2-18
- Replace an Existing DSTB, page 2-19
- Remove an Existing DSTB, page 2-20

Voice

- Preprovisioned xGCP Capable DOCSIS Modem, page 2-20
- Activating an Additional Telephone Line on an xGCP Capable DOCSIS Modem, page 2-22
- Disable an Existing Telephone Line on an xGCP Capable DOCSIS Modem, page 2-23

Configuration Generation

- Generate a Configuration, page 2-23

DHCP Criteria

- Add DHCP Criteria, page 2-24

System Defaults

- Change System Defaults, page 2-25

Using Events

- Log Device Deletions, page 2-25
- Monitor a Connection to the RDU, page 2-26
- Log Batch Completions, page 2-27

Device Searches

- Get Detailed Information for a Device, page 2-27
- Retrieve Devices Matching a Vendor Prefix, page 2-28

The remainder of this chapter contains detailed examples of these use cases.

Self-Provisioned Modem and Computer in Fixed Standard Mode

Case: A subscriber has a computer (with a browser application) installed in a single dwelling unit and has purchased a DOCSIS cable modem.

Desired Outcome: Use this workflow model to bring a new unprovisioned DOCSIS cable modem and computer online so that they have the appropriate level of service.

Step 1 The subscriber purchases a DOCSIS cable modem, installs it in the home, and connects the computer to the cable modem.

Step 2 The subscriber turns on the modem and BPR gives it restricted access.

Step 3 The subscriber turns on the computer, BPR gives it restricted access.

Step 4 The subscriber starts a browser application on the computer and a spoofing DNS server points the browser to the service provider's registration server (for example, an OSS user interface or a mediator).

Step 5 The subscriber uses the service provider's user interface to complete the steps required for registration, including selecting class of service.

Step 6 The service provider's user interface passes the subscriber's information, such as the selected class of service and computer IP address, to BPR. The subscriber's modem and computer are then registered with BPR.

```
// First we query the computer's information to find the modem’s MAC address
// We use the computers IP address (the web browser received this when the
// subscriber opened the service providers web interface)

get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// NO_ACTIVATION is the activation mode because this is a query

// NO_CONFIRMATION is the confirmation mode because we are not attempting to
// reset the device

Map computerLease = getDetailsforIPDevicebyIPAddress(
      "10.0.14.38")             // ipAddress: restricted access computer lease, from
                                // Network Registrar DHCP
// Derive the modem MAC address from the computer's network information
// The 1,6, is a standard prefix for an Ethernet device
// The fully qualified MAC address is required by BPR
String modemMACAddress = "1,6," + computerLease.RELAY_AGENT_REMOTE_ID;

// Now let’s provision the modem and the computer in the same batch.
// This can be done because the activation mode of this batch is NO_ACTIVATION.
// More than one device can be operated on in a batch if the activation mode
// does not lead to more than one device being reset.

// NO_ACTIVATION will generate a configuration for the devices. However it will
// not attempt to reset the devices.
// The configuration can be generated because the devices have booted.

// NO_CONFIRMATION is the confirmation mode because we are not attempting to
// reset the modem

// We do not want to reset the modem here because we want to notify the user to
// reset their computer. If we reset the modem in this batch, we will not be
// able to notify the user of anything until the modem has come back online.

// To add a DOCSIS modem:
addDOCSISModem(
modemMACaddress,                // macAddress: derived from computer lease
      null,                     // FQDN: not used in this example
      "0123-45-6789",           // ownerID: here, account number from billing system
      "Silver",                 // ClassOfService
      "provisionedCM",          // DHCP Criteria: Network Registrar uses this to
                                // select a modem lease granting provisioned IP address
      null,                     // CPE DHCP Criteria: not used in this example
      null);                    // properties: not used
```

```
// Create a Map for the computer's properties

Map properties;

// Setting the property IPDeviceKeys.MUST_BE_BEHIND_DEVICE on the computer
// ensures that when the computer boots it will only receive
// its provisioned access when it is behind the given device. If it is not
// behind the given device, it will receive default access (unprovisioned)

properties.put(IPDeviceKeys.MUST_BE_BEHIND_DEVICE, modemMACaddress);


addComputerByIPAddress(
      "10.0.14.37",             // ipAddress: restricted access lease, from Network
                                // Registrar DHCP
      null,                     // FQDN: not used in this example
      "0123-45-6789",           // ownerID: here, account number from billing system
null,                           // ClassOfService: not used in this example
      "provisionedCPE",         // DHCP Criteria: Network Registrar uses this to select
                                // a CPE lease granting a provisioned IP address
      properties);              // properties: as previously set
```

Step 7 The user interface instructs the subscriber to reboot the computer.

Step 8 The provisioning client calls resetIPDevice to reboot the modem and gives the modem provisioned access.

```
get-new-batch(AUTOMATIC, NO_CONFIRMATION)

// AUTOMATIC is the activation mode because we are attempting to reset the modem
// so that it receives its new class of service.

// NO_CONFIRMATION is the confirmation mode because we don't want the batch to fail
// if we can't reset the modem. The user may have power cycled the modem when
// they rebooted their computer.

// Send a batch to reset the modem now that the user has been notified to reboot
// their computer.

resetIPDevice(modemMACaddress);
```

Step 9 When the computer is rebooted, it receives a new IP address, and the cable modem and computer are now both provisioned devices. The computer has access to the Internet through the service provider's network.

Add a New Computer in Fixed Standard Mode

Case: The landlord of an apartment building has four tenants sharing a modem and accessing the service provider's network. The landlord wants to provide Internet access to a new tenant, sharing the building's modem. The new tenant has a computer connected to the modem.

Desired Outcome: Use this workflow model to bring a new unprovisioned computer online with a previously provisioned cable modem so that the new computer has the appropriate level of service.

Step 1 The subscriber turns on the computer, BPR gives it restricted access.

Step 2 The subscriber starts a browser application on the computer and a spoofing DNS server points the browser to the service provider's registration server (for example, an OSS user interface or a mediator).

Step 3 The subscriber uses the service provider's user interface to complete the steps required for adding a new computer.

```
get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// NO_ACTIVATION will generate a configuration for the computer. However it will
// not attempt to reset the computer (this cannot be done)
// The configuration can be generated because the device has booted.

// NO_CONFIRMATION is the confirmation mode because we are not attempting to
// reset the modem.

Map properties;                  // Map containing the properties for the computer

// Setting the property IPDeviceKeys.MUST_BE_BEHIND_DEVICE on the computer
// ensures that when the computer boots, it will only receive
// its provisioned access when it is behind the given device. If it is not
// behind the given device, it will receive default access (unprovisioned).

properties.put(IPDeviceKeys.MUST_BE_BEHIND_DEVICE, modemMACaddress);

addComputerByIPAddress(
      "10.0.14.37",              // ipAddress: restricted access lease, from Network
                                 // Registrar DHCP
      null,                      // FQDN: not used in this example
      "0123-45-6789",            // ownerID: here, account number from billing system
      null,                      // ClassOfService: not used in this example
      "provisionedCPE",          // DHCP Criteria: Network Registrar uses this to select
                                 // a CPE lease granting a provisioned IP address
      properties);               // properties: as previously set
```

Step 4 The user interface instructs the subscriber to reboot the computer so that BPR can give the computer its registered service level.

Step 5 The computer is now a provisioned device with access to the appropriate level of service.

Disable a Subscriber in Fixed Standard Mode

Case: A service provider needs to disable a subscriber from the network due to recurring tardiness in payment.

Desired Outcome: Disable an operational modem and computer so that the devices temporarily cannot access the service provider's network.

Step 1 The service provider’s application disables the subscriber's device information.

Step 2 The service provider's application uses a provisioning client program to request a list of all of the subscriber's devices from BPR. Then, the service provider's application uses the provisioning client to disable each of the subscriber's devices individually.

```
// The service provider’s application uses a provisioning client to get a list of
// all of the subscriber's devices.
get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// NO_ACTIVATION is the activation mode because this is a query.

// NO_CONFIRMATION is the confirmation mode because we are not attempting to
// reset the devices.

getIPDevicesByOwnerID(
      "0123-45-6789");            // Query all the devices for this account number
// For each device in the retrieved list:
// We are only interested in the modems. If we disable network access
// of the modems, we disable network access for all the subscriber’s devices.
// If we are using roaming standard mode, we also change the computer’s
// network access (DHCPCriteria) because the subscriber may still be able
// to access the network from a different modem.

DeviceLoop:
      {
      if (Device.deviceType == DOCSIS_MODEM)
      {
      get-new-batch(AUTOMATIC, NO_CONFIRMATION)
      // AUTOMATIC is the activation mode because we are attempting to
      // reset the modem so that becomes disabled.

      // NO_CONFIRMATION is the confirmation mode because we do not
      // want the batch to fail if we cannot reset the modem.
      // If the modem is off, when it will be disabled when it is turned back on.

      changeDOCSISModemClassOfService(
      Device.MAC_ADDRESS,         // macAddress: unique identifier for this modem
      "Disabled");                // newClassOfService: disable network access
}
}
// end DeviceLoop
```

Step 3 The subscriber is now disabled.

Preprovisioned Modem/Self-Provisioned Computer in Roaming Standard Mode

Case: A new subscriber contacts the service provider and requests service. The subscriber has a computer installed in a single dwelling unit. The service provider preprovisions all of its cable modems in bulk.

Desired Outcome: Bring a preprovisioned cable modem and an unprovisioned computer online so that they have the appropriate level of service. Both devices will be registered.

Step 1 The service provider chooses a subscriber username and password for the billing system.

Step 2 The service provider selects the services that the subscriber can access.

Step 3 The service provider's field technician installs the physical cable to the new subscriber's house and installs the preprovisioned device, connecting it to the subscriber's computer.

Step 4 The technician turns on the modem and BPR gives it a provisioned IP address.

Step 5 The technician turns on the computer and BPR gives it a private IP address.

Step 6 The technician starts a browser application on the computer and points the browser to the service provider's user interface.

Step 7 The technician uses the service provider's user interface to complete the steps required for registering the computer behind the provisioned cable modem.

```
// To provision a computer:
// MSO admin UI calls the provisioning API to provision a computer.

get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// NO_ACTIVATION will generate a configuration for the computer however it will
// not attempt to reset the computer
// The configuration will be able to be generated because the computer has booted.
// NO_CONFIRMATION is the confirmation mode because we are not attempting to
// reset the computer because this cannot be done
addComputerByIPAddress(
     "10.0.14.37",            // ipAddress: restricted access lease, from Network
                              // Registrar DHCP
     null,                    // FQDN: not used in this example
     "0123-45-6789",          // ownerID: here, account number from billing system
     null,                    // ClassOfService: required, but value can be null
     "provisionedCPE",        // DHCP Criteria: Network Registrar uses this to select
                              // a CPE lease
                              // granting provisioned IP address
     null);                   // properties: what has been set earlier
```

Step 8 The technician restarts the computer and the computer receives a new IP address. The cable modem and the computer are now both provisioned devices. The computer has access to the Internet through the service provider's network.

Modify an Existing Modem in Roaming Standard Mode

Case: A service provider's subscriber currently has Silver service. The subscriber has decided to upgrade to Gold service. The subscriber has a computer installed in the home.

Note The intent of this use case is to show how to modify a device. You can apply this example to devices provisioned in modes other than roaming standard.

Desired Outcome: Modify an existing modem's class of service and pass that change of service to the service provider's external systems.

Step 1 The subscriber phones the service provider and requests to have service upgraded. The service provider uses its user interface to change the class of service from Silver service to the Gold.

Step 2 The service provider's application makes these API calls in BPR:

```
get-new-batch(AUTOMATIC, NO_CONFIRMATION)

// AUTOMATIC will generate a configuration for the device and will
// attempt to reset the device
// The configuration will be able to be generated because the device has booted.

// NO_CONFIRMATION is the confirmation mode because we don't want the batch to fail
// if the reset failed

// This use case is a perfect example of the different confirmation modes that
// could be used instead of NO_CONFIRMATION. These confirmation modes give you
// a method to test whether a configuration was taken by a device. Also,
// these modes will take more time because the batch has to wait for the modem to reset.

changeDOCSISModemClassOfService(
      "1,6,00:11:22:33:44:55",    // macAddress: unique identifier for this modem
      "Gold");                    // newClassOfService
```

Step 3 The subscriber can now access the service provider's network using the Gold service.

Delete a Subscriber’s Devices in Roaming Standard Mode

Case: A service provider needs to delete a subscriber who has discontinued service.

Note The intent of this use case is to show how to delete a device. You can apply this example to devices provisioned in modes other than roaming standard.

Desired Outcome: Permanently remove all of the subscriber's devices from the service provider's network.

Step 1 The service provider's user interface discontinues service to the subscriber.

Step 2 The service provider's application uses the provisioning client program to request a list of all of the subscriber's devices from BPR.

Step 3 The service provider's application uses the provisioning client program to delete and reset each of the subscriber's devices individually.

```
// MSO admin UI calls the provisioning API to get a list of all the subscriber's
// devices.

get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// NO_ACTIVATION is the activation mode because this is a query

// NO_CONFIRMATION is the confirmation mode because we are not attempting to
// reset the devices

     getIPDevicesByOwnerID(
     "0123-45-6789");         // query all the devices for this account number
                              // for each device in the retrieved list:
DeviceLoop:
{

     // get a new batch for each modem being deleted
     get-new-batch(AUTOMATIC, NO_CONFIRMATION)

     // AUTOMATIC is the activation mode because we want
     // to generate a configuration because we are deleting the device.
     // This will also attempt to reset the device

     // NO_CONFIRMATION is the confirmation mode because we do not want
     // the batch to fail if the reset fails

     deleteIPDevice(
     Device.MAC_ADDRESS);     // macAddress: unique identifier for this device
}
// end DeviceLoop.
```

Self-Provisioned First Time Activation in Promiscuous Mode

Case: The subscriber has a computer (with a browser application) installed in a single dwelling unit and has purchased a DOCSIS cable modem.

Desired Outcome: Bring a new unprovisioned DOCSIS cable modem and computer online with the appropriate level of service.

Step 1 The subscriber purchases a DOCSIS cable modem and installs it in the home.

Step 2 The subscriber turns on the modem, and BPR gives it restricted access.

Step 3 The subscriber turns on the computer, and BPR gives it restricted access.

Step 4 The subscriber starts a browser application on the computer and a spoofing DNS server points the browser to the service provider's registration server (for example, an OSS user interface or a mediator).

Step 5 The subscriber uses the service provider's user interface to complete the steps required for registration, including selecting class of service.

Step 6 The service provider's user interface passes the subscriber's information to BPR, including the selected classes of service and computer IP address. The subscriber is then registered with BPR.

```
get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// NO_ACTIVATION is the activation mode because this is a query

// NO_CONFIRMATION is the confirmation mode because we are not attempting to
// reset the device

// First we query the computer's information to find the modems MAC address
// We use the computers IP address (the web browser received this when the
// subscriber opened the service providers web interface

Map computerLease = getDetailsforIPDevicebyIPAddress(
     "10.0.14.38")            // ipAddress: restricted access computer lease, from
                              // Network Registrar DHCP

// derive the modem MAC address from the computer's network information

// The 1,6, is a standard prefix for an Ethernet device
// The fully qualified MAC address is required by BPR

String modemMACaddress = "1,6," + computerLease.RELAY_AGENT_REMOTE_ID;

get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// now let’s provision the modem
// NO_ACTIVATION will generate a configuration for the modem however it will
// not attempt to reset it
// The configuration will be able to be generated because the modem has booted.

// NO_CONFIRMATION is the confirmation mode because we are not attempting to
// reset the modem

// Create a Map for the properties of the modem

Map properties;

// Set the property ModemKeys.PROMISCUOUS_MODE_ENABLED to enable promiscuous
// mode on modem

properties.put(ModemKeys.PROMISCUOUS_MODE_ENABLED, "enabled");

addDOCSISModem(
     modemMACaddress,         // macAddress: the unique MAC for the modem
     null,                    // FQDN: not used in this example
     "0123-45-6789",          // ownerID: here, account number from billing system
     "Silver",                // ClassOfService
     "provisionedCM",         // DHCP Criteria: Network Registrar uses this to
                              // select a modem
                              // lease granting provisioned IP address
     "provisionedCPE",        // CPE DHCP Criteria: Network Registrar uses
                              // to grant a provisioned lease to the
                              // CPEs behind this modem
     properties);             // properties: the properties of the modem
```

Step 7 The user interface instructs the subscriber to reboot the computer.

Step 8 The provisioning client calls resetIPDevice to reboot the modem and gives the modem provisioned access.

```
get-new-batch(AUTOMATIC, NO_CONFIRMATION)

// AUTOMATIC is the activation mode because we are attempting to reset the modem
// so that is receives its new class of service

// NO_CONFIRMATION is the confirmation mode because we don't want the batch to fail
// if we can't reset the modem. The user might have power cycled the modem when
// they rebooted their computer

// send a batch to reset the modem now that the user has been notified to reboot
// their computer

resetIPDevice(modemMACaddress);
```

Step 9 When the computer is rebooted, it receives a new IP address. The cable modem is now a provisioned device. The computer is not registered with BPR, but it gains access to the Internet through the service provider's network. Computers that are online behind promiscuous modems are still available using the provisioning API.

Bulk Provision 100 Modems in Promiscuous Mode

Case: A service provider wants to preprovision 100 cable modems for distribution by a customer service representative at a service kiosk.

Desired Outcome: Modem data for all modems to be distributed to new subscribers is already online, so the customer service representative has a list of modems available for assignment.

Step 1 Modem MAC address data for new (or recycled) cable modems is collected into a list at the service provider's loading dock.

Step 2 Modems being assigned to a particular kiosk are bulk loaded into BPR, flagged with the identifier for that kiosk.

Step 3 When the modems are distributed to new subscribers at the kiosk, the customer service representative will enter new service parameters, and will change the Owner ID field on the modem to reflect the new subscriber's account number.

```
// get a single batch for bulk load or 100 modems

get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// The activation mode for this batch should be NO_ACTIVATION
// NO_ACTIVATION should be used in this situation because no network information
// exists for the devices because they have not booted. A configuration cannot
// be generated if no network information is present. And because
// the devices have not booted, they are not online and therefore cannot be reset

// NO_CONFIRMATION is the confirmation mode because we are not attempting to
// reset the devices

// Create a Map for the properties of the modem

Map properties;

// Set the property ModemKeys.PROMISCUOUS_MODE_ENABLED to enable promiscuous
// mode on modem
// This could be done at a system level if promiscuous mode is your
// default provisioning mode

properties.put;(ModemKeys.PROMISCUOUS_MODE_ENABLED, "enabled")

// for each modem MAC-address in list:
ModemLoop:
{
     addDOCSISModem(
          List.MAC_ADDRESS,       // macAddress: unique identifier for this modem
               null,              // FQDN: not used in this example
               kioskIDNumber,     // ownerID: here, each modem is flagged with the
                                  // identifier for the kiosk at which it will
                                  // be assigned to a new subscriber
               "Bronze",          // ClassOfService
               "provisionedCM",   // DHCP Criteria: Network Registrar uses this to
                                  // select a modem
                                  // lease granting provisioned IP address
               "provisionedCPE",  // CPE DHCP Criteria: Network Registrar uses
                                  // to grant a provisioned lease to the
                                  // CPEs behind this modem
               properties);       // properties: the properties of the modem
}
     // end ModemLoop.
```

Preprovisioned First Time Activation in Promiscuous Mode

Case: A new subscriber contacts the service provider and requests service. The subscriber has a computer installed in a single dwelling unit.

Desired Outcome: Bring a new unprovisioned cable modem and computer online so that they have the appropriate level of service.

Step 1 The service provider chooses a subscriber username and password for the billing system.

Step 2 The service provider selects the services that the subscriber can access.

Step 3 The service provider registers the device using their own user interface.

Step 4 The service provider's user interface passes information to BPR, such as the modem's MAC address and the class of service. Additionally, the modem gets a CPE DHCP Criteria setting that enables Network Registrar to select a provisioned address for any computers that will be connected behind the modem. The new modem is then registered with BPR.

Step 5 The service provider's field technician installs the physical cable to the new subscriber's house and installs the preprovisioned device, connecting it to the subscriber's computer.

```
// MSO admin UI calls the provisioning API to pre-provision an HSD modem.

get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// The activation mode for this batch should be NO_ACTIVATION
// NO_ACTIVATION should be used in this situation because no network information
// exists for the modem because it has not booted. A configuration cannot
// be generated if no network information is present. And because
// the modem has not booted, it is not online and therefore can not be reset

// NO_CONFIRMATION is the confirmation mode because we are not attempting to
// reset the modem

// Create a Map for the properties of the modem

Map properties;

// Set the property ModemKeys.PROMISCUOUS_MODE_ENABLED to enable promiscuous
// mode on modem

properties.put;(ModemKeys.PROMISCUOUS_MODE_ENABLED, "enabled")

addDOCSISModem(
     "1,6,00:11:22:33:44:55",   // macAddress: technician reads this off the modem
     null,                      // FQDN: not used in this example
     "0123-45-6789",            // ownerID: here, account number from billing system
     "Silver",                  // ClassOfService
     "provisionedCM",           // DHCP Criteria: Network Registrar uses this to
                                // select a modem
                                // lease granting provisioned IP address
     "provisionedCPE",          // CPE DHCP Criteria: Network Registrar uses
                                // to grant a provisioned lease to the
                                // CPEs behind this modem
     properties);               // properties: the properties of the modem
```

Step 6 The technician turns on the modem and BPR gives it provisioned access.

Step 7 The technician turns on the computer and BPR gives it provisioned access. The cable modem and the computer are now both provisioned devices. The computer has access to the Internet through the service provider's network.

Replace an Existing Modem in Promiscuous Mode

Case: A service provider wants to replace a broken modem.

Note The intent of this use case is to show how to replace a device. You can apply this example to devices provisioned in modes other than promiscuous.

If the computer has the option restricting roaming from one computer to another, and the modem is replaced, the computer's MAC Address for the modem must also be changed.

Desired Outcome: Physically replace an existing modem with a new modem without changing the level of service provided to the subscriber.

Step 1 The service provider changes the MAC address of the existing modem to that of the new modem.

```
get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// NO_ ACTIVATION is the activation mode because we will not be able
// to generate a new configuration because the new modem has not booted on the
// network.

// NO_CONFIRMATION is the confirmation mode because we are not trying to
// reset the modem


// To change the MAC address of a DOCSIS modem:
changeDOCSISModemMacAddress (
     "1,6,00:11:22:33:44:55"    // old macAddress: unique identifier for the old modem
     "1,6,11:22:33:44:55:66");  // new macAddress: unique identifier for the new modem
```

Step 2 The service provider replaces the modem and turns it on. The computer is also turned on.

Step 3 The cable modem is now a fully provisioned device. The modem and the computer behind it have the appropriate level of service.

Add a Second Computer in Promiscuous Mode

Case: A subscriber wishes to connect a second computer behind an installed cable modem.

Desired Outcome: Providing the subscriber's selected service permits connecting multiple CPEs, the subscriber has network access from both connected computers.

Note This case does not require calls to the provisioning API.

Step 1 The subscriber connects a second computer behind the cable modem.

Step 2 The subscriber turns on the computer.

Step 3 If the subscriber's selected service permits connecting multiple CPEs, BPR gives the second computer access to the Internet.

Self-Provisioning First Time Activation with NAT

Case: A university has purchased a DOCSIS cable modem with network address translation (NAT) and DHCP capability. The five occupants of the unit each have a computer installed with a browser application.

Desired Outcome: Use this workflow model to bring a new unprovisioned cable modem (with NAT) and the computers behind it online so that they have the appropriate level of service.

Step 1 The subscriber purchases a cable modem with NAT and DHCP capability, and installs it in a multiple dwelling unit.

Step 2 The subscriber turns on the modem and BPR gives it restricted access.

Step 3 The subscriber connects a laptop computer to the cable modem, and the DHCP server in the modem provides an IP address to the laptop.

Step 4 The subscriber starts a browser application on the computer and a spoofing DNS server points the browser to the service provider's registration server (for example, an OSS user interface or a mediator).

Step 5 The subscriber uses the service provider's user interface to complete the steps required for registration of the modem. The registration user interface detects that the modem is using NAT and registers the modem, making sure that the modem gets a class of service that is compatible with NAT.

```
get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// NO_ACTIVATION is the activation mode because this is a query

// NO_CONFIRMATION is the confirmation mode because we are not attempting to
// reset the device
// First we query the computer's information to find the modems MAC address
// With NAT, the computer is never seen by the Network Registrar DHCP server. Instead,
// the modem has translated the IP address it assigned to the computer, so
// the web browser sees the modem's IP address. When the lease data is
// examined, the MAC address for the device and the relay-agent-remote-id are the
// same, indicating that this is an unprovisioned modem device, which is
// therefore presumed to be performing NAT.

Map deviceDetails = getDetailsForIPDevice(
     "10.0.15.22");              // ipAddress

// MSO client registration program then calls the provisioning API to
// provision the NAT modem (with an appropriate class of service for NAT).

get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// NO_ACTIVATION will generate a configuration for the modem however it will
// not attempt to reset it
// The configuration will be able to be generated because the modem has booted.

// NO_CONFIRMATION is the confirmation mode because we are not attempting to
// reset the modem

addDOCSISModem(
     deviceDetails.MAC_ADDRESS,  // macAddress: technician reads this off the modem
     null,                       // FQDN: not used in this example
     "0123-45-6789",             // ownerID: here, account number from billing system
     "Silver",                   // classOfService
     "provisionedCM",            // modem DHCP Criteria: Network Registrar uses
                                 // to select a modem lease granting
                                 // provisioned IP address
     null,                       // CPE DHCP Criteria: not used in this example
     null);                      // properties: not used (but map could contain
                                 // optional data, such as a description field)
```

Step 6 The user interface instructs the subscriber to reboot the computer.

Step 7 The provisioning client calls resetIPDevice to reboot the modem and gives the modem provisioned access.

```
get-new-batch(AUTOMATIC, NO_CONFIRMATION)

// AUTOMATIC is the activation mode because we are attempting to reset the modem
// so that it receives its new class of service.

// NO_CONFIRMATION is the confirmation mode because we don't want the batch to fail
// if we can't reset the modem. The user might have power cycled the modem when
// they rebooted their computer

// send a batch to reset the modem now that the user has been notified to reboot their
// computer

resetIPaddress {
deviceDetails.MAC_ADDRESS
};
```

Step 8 The cable modem is now fully provisioned and the computers behind it have full access to the service provider's network.

Note Certain cable modems with NAT may require you to reboot the computer to get the new class of service settings.

If the cable modem and NAT device are separate devices, the NAT device would also have to be registered similar to registering a computer.

Add a New Computer Behind a Modem with NAT

Case: The landlord of an apartment building has four tenants sharing a modem and accessing the service provider's network. The landlord wants to provide Internet access to a new tenant, sharing the building's modem. The modem has NAT and DHCP capability. The new tenant has a computer connected to the modem.

Desired Outcome: Use this workflow model to bring a new unprovisioned computer online with a previously provisioned cable modem so that the new computer has the appropriate level of service.

Note This case does not require calls to the provisioning API.

Step 1 The subscriber turns on the computer.

Step 2 The computer is now a provisioned device with access to the appropriate level of service.

Note The provisioned NAT modem hides the computers behind it from the network.

Preprovisioned First Time Activation for DSTB

Case: A new subscriber contacts the service provider and requests service. The service provider determines the type of service the customer wants (for example, interactive digital video and data, or just digital video).

Desired Outcome: Use this workflow model to bring a new preprovisioned DSTB online so that it has the appropriate level of service.

Step 1 The service provider chooses a subscriber username and password for the billing system.

Step 2 The service provider selects the services that the subscriber can access.

Step 3 The service provider selects an appropriate DSTB based on the services the subscriber can access. For example, digital video service requires a DVB DSTB, while digital video and interactive data require a DOCSIS/DVB DSTB.

Step 4 The service provider uses their own user interface to complete the steps required to register the DSTB.

Step 5 The service provider's user interface passes information to BPR, such as the MAC addresses and classes of service for each component of the DSTB (for example, the DOCSIS cable modem, DVB cable modem, and computer middleware components). The DSTB is then registered with BPR.

```
// MSO admin UI calls the provisioning API to pre-provision DOCSIS modem,
// DVB Modem, and DSTB middleware components of a DSTB.

get-new-batch(NO_ACTIVATION)       // get a single batch for all 2 DSTB components.
                                   // ActivationMode.NO_ACTIVATION
                                   // means that BPR will NOT reboot the modem(s).

addDOCSISModem(
        "1,6,00:11:22:33:44:55",   // macAddress: technician reads this off the modem
        null,                      // FQDN: not used in this example
        "0123-45-6789",            // ownerID: here, account number from billing system
        "Bronze",                  // ClassOfService
        "provisionedCM",           // modemClientClass: Network Registrar uses this to:
                                   //      select a modem
                                   //      lease granting provisioned access
        null,                      // cpeDHCPCriteria: not used in this example
        null);                     // properties: not used (but map could contain
                                   //      optional data, such as a description field)

addDSTB(
        "1,6,99:88:77:66:55:44",   // macAddress: technician reads this off the DSTB
        null,                      // FQDN: not used in this example
        "0123-45-6789",            // ownerID: here, account number from billing system
        null,                      // ClassOfService: not used in this example
        "provisionedDSTB",         // DHCP Criteria; Network Registrar uses this to
                                   //select a CPE lease granting provisioned access
        "1,6,00:11:22:33:44:55",   // relatedDOCSISModemID: MAC address for DOCSIS modem
                                   //      component
        null,                      // DVB component MAC address
        null);                     // properties: not used (but map could contain
                                   //      optional data, such as a description field)
```

Step 6 The service provider's field technician installs the physical cable to the new subscriber's home and installs the preprovisioned DSTB.

Step 7 The DSTB boots and BPR gives it provisioned access.

Step 8 The technician verifies that the DSTB is operating properly for the services that the voice subscriber selected.

Modify an Existing DSTB

Case: A subscriber currently has the Bronze service and wants to upgrade to the Silver service. The subscriber has a DSTB installed in the home.

Desired Outcome: Modify an existing DSTB's class of service and pass that change of service to the service provider's external systems.

Step 1 The subscriber starts the DSTB and points the browser to the service provider's user interface. The subscriber uses the user interface to remove the current Bronze service and to select the Silver service.

Step 2 The service provider's user interface passes information to BPR, such as the IP address of the DSTB computer middleware and the new class of service.

```
// MSO admin UI calls the provisioning API to change service on a DSTB.
// BPR queries the DSTB to find the related modem(s).
Map DSTB = getDetailsForDSTBByIPAddress(
        "10.0.17.49");                          // query the DSTB
```

Step 3 BPR writes the changed class of service information for the DSTB's related modem components to the embedded database.

```
// MSO admin UI next calls the provisioning API to change service on
// the related DOCSIS modem components of the DSTB.
get-new-batch(NO_ACTIVATION)          // get a separate batch for each component modem

changeDOCSISModemClassOfService(
          DSTB.relatedDOCISModemID,   // macAddress: the identifier (MAC address) for
                                      //     the related DOCSIS modem component
          "Silver");                  // newClassOfService
```

Step 4 The service provider instructs the subscriber to reboot the DSTB.

Step 5 The user can now access the service provider's network using the Silver level of service.

Replace an Existing DSTB

Case: A service provider wants to replace an existing DSTB.

Desired Outcome: Physically replace an in-service DSTB with a new one. Provision the new box with the same classes of service as the old DSTB.

Step 1 The service provider, using its own user interface, accesses the data for the old DSTB. The new DSTB is then preprovisioned with the same class of service information as the old box.

Note Because BPR supports only preprovisioning of a DSTB, the service provider must perform step one.

```
// MSO admin UI calls the provisioning API to access data for the old
// DSTB and its related modem components.
get-new-batch()                 // 1st query

Map DSTB = getDetailsForDSTB(
      "1,6,99:88:77:66:55:44"); // query the DSTB

get-new-batch(NO_ACTIVATION)    // get a single batch for all 2 DSTB components.
```

```
                                  //    ActivationMode.NO_ACTIVATION
                                  //    means that BPR will NOT reboot the
                                  //    modem(s).

changeDOCSISModemMACAddress(DSTB.relatedDOCSISModemID,     // MAC address of the
                                                           // DOCSIS component of the
                                                           // old DSTB

                                  "1,6,99:88:77:66:55:35"); // MAC address of the new
                                                           // DSTB

changeDSTBMACAddress("1,6,99:88:77:66:55:44",              // MAC address of the old
                                                           // DSTB.

                                  "1,6,99:88:77:66:55:33"); // MAC address of the new DSTB
```

Step 2 The service provider sends the new DSTB to the subscriber.

Remove an Existing DSTB

Case: A subscriber wants to remove an existing DSTB currently installed in a subscriber’s home.

Desired Outcome: Physically remove an existing DSTB from a subscriber’s home.

Step 1 The subscriber removes an existing DSTB from the home.

Step 2 The subscriber sends the DSTB to the service provider.

Step 3 The service provider receives the DSTB into inventory. An inventory control person uses their own registration server (for example, an OSS user interface or a mediator) to remove the DSTB from active duty and places it in inventory.

Step 4 The service provider uses the registration server to remove each component of the DSTB from BPR.

```
// MSO admin UI calls the provisioning API to remove the inactive DSTB and its
// component modems from BPR.

deleteDSTB(
        "1,6,99:88:77:66:55:44",    // macAddress: unique identifier for this DSTB
        true);                      // deleteAllRelatedModems: true, so also delete
                                    //      related DOCSIS and/or DVB modems
```

Preprovisioned xGCP Capable DOCSIS Modem

Case: A new subscriber contacts the service provider and requests voice service.

Desired Outcome: Use this workflow model to bring a new xGCP DOCSIS modem (MTA) online so that it has the appropriate level of service to place and receive VoIP calls over the service provider's network.

Note A call agent is required for end-to-end provisioning.

Step 1 The service provider chooses a subscriber username and password for the billing system.

Step 2 The service provider selects the number of telephone lines to enable on the MTA, and selects the VoIP services and features that the subscriber can access on those lines.

Step 3 The service provider uses their own user interface to complete the steps required to register the device.

Step 4 The service provider's user interface passes device information to BPR, such as MAC address, FQDN, classes of service, and VoIP services. The new device is then registered with BPR.

```
// First we add the DOCSIS Modem, then we add xGCP services to it

get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// NO_ACTIVATION is the Activation mode because we will not be able
// to generate a new configuration because the new modem has not booted on the
// network.

// NO_CONFIRMATION is the Confirmation mode because we are not attempting to
// reset the modem

addDOCSISModem(
     "1,6,00:11:22:33:44:55",   // macAddress: the MAC address of the modem
     "mta.mso.com",             // FQDN: the FQDN of the modem
     "0123-45-6789",            // ownerID: here, account number from billing system
     "Silver",                  // classOfService
     "provisionedCM",           // DHCP Criteria: Network Registrar uses this to
                                // select a modem
                                // lease granting provisioned IP address
     null,                      // CPE DHCP Criteria: not used in this example
     null);                     // properties: not used in this example

// Now lets add a xGCP service. This is done in the same batch as the modem

addXGCPService(
     "1,6,ff:00:ee:11:dd:22",   // macAddress: the MAC address of the modem
     0,                         // portNumber: Number of line to be enabled.
                                // Line numbers are 0 based
     "ca",                      // cmsQualifier: CMS qualifier for CMS which
                                // will control this line
     "ca1.mso.net",             // cmsFQDN: CMS FQDN for CMS which
                                // will control this line
     5678,                      // cmsPortNumber: CMS Port Number for CMS which
                                // will control this line
     "(978)123-4567",           // telephoneNumber: telephone number for this
                                // line - for informational purposes
     null);                     // properties: not used in this example
```

Step 5 The service provider's field technician installs the physical cable to the new subscriber's home and installs the pre-registered DOCSIS modem MTA. Analog phones are then connected to the MTA lines enabled through the provisioning workflow.

Step 6 The MTA boots and BPR gives it provisioned VoIP service through the service provider’s network.

Activating an Additional Telephone Line on an xGCP Capable DOCSIS Modem

Case: A service provider needs to provision an additional telephone line on a provisioned xGCP DOCSIS modem (MTA).

Desired Outcome: The subscriber can place and receive VoIP calls over an additional line using the service provider's network.

Step 1 The service provider uses their own user interface to enable an additional telephone line, as well as the VoIP services and features the subscribers can have access to for that new line.

Step 2 The service provider's user interface passes the information to BPR. BPR updates the device information and resets the device.

```
get-new-batch(AUTOMATIC, NO_CONFIRMATION)

// AUTOMATIC is the Activation mode because we are attempting to reset the modem
// so that is receives its new phone line

// NO_CONFIRMATION is the Confirmation mode because we don't want the batch to fail
// if we can't reset the modem.

addXGCPService(
     "1,6,ff:00:ee:11:dd:22",      // macAddress: the MAC address of the modem
     1,                            // portNumber: Number of line to be enabled.
     "ca",                         // cmsQualifier: CMS qualifier for CMS which
                                   // will control this line
     "ca1.mso.net",                // cmsFQDN: CMS FQDN for CMS which
                                   // will control this line
     5678,                         // cmsPortNumber: CMS Port Number for CMS which
                                   // will control this line
     "(978)123-4567",              // telephoneNumber: telephone number for this
                                   // line - for informational purposes
     null);                        // properties: not used in this example
```

Step 3 The subscriber can now place and receive telephone calls over the new telephone line.

Disable an Existing Telephone Line on an xGCP Capable DOCSIS Modem

Case: A service provider needs to disable a telephone line for a subscriber who has voluntarily decided to discontinue service on the line, or due to recurring tardiness in payment.

Desired Outcome: Disable an operational telephone line.

Step 1 The service provider uses their own user interface to disable a previously provisioned telephone line.

Step 2 The service provider's user interface, acting as a BPR client, passes the information to BPR. BPR updates the device information and resets the device.

```
get-new-batch(AUTOMATIC, NO_CONFIRMATION)

// AUTOMATIC is the Activation mode because we are attempting to reset the modem
// so that a phone line is disabled

// NO_CONFIRMATION is the Confirmation mode because we don't want the batch to fail
// if we can't reset the modem.

// To delete service for a xGCP DOCSIS modem
deleteXGCPService(
"1,6,ff:00:ee:11:dd:22",      // macAddress: the MAC address of the modem
0);                           // port number
```

Step 3 After being rebooted, the subscriber cannot place and receive telephone calls on the disabled phone line.

Generate a Configuration

Case: The service provider has changed the properties of the DOCSIS Gold class of service. Now a configuration has to be generated for all the DOCSIS devices that have the Gold class of service.

Desired Outcome: Generate a configuration for a device using provisioning the API.

Step 1 Get all devices with the DOCSIS class of service Gold.

```
get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

getIPDevicesByClassOfService("Gold", // COS name
                    ClassOfServiceType.DOCSIS, // Specifies DOCSIS COS Type
                    0) // Return all matches
```

Step 2 Generate configuration for all the devices in the list created in Step 1.

```
get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

generateConfiguration(deviceID );  // Device ID from the list returned in Step 1
```

Add DHCP Criteria

Case: A registered cable modem needs an IP address from a different Network Registrar scope.

Desired Outcome: A provisioning client adds the DHCP criteria, and the cable modem receives an appropriate IP address.

Step 1 Create a DHCP criteria with the desired scope selection tags.

```
get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

addDHCPCriteria( "modemCriteria", // DHCP Criteria Name
                              null, // Client Class
                              "provisionedModemTag", // Include Selection Tag
                              "provisionedCPETag", // Exclude Selection Tag
                              null // Properties
                              );
BatchStatus stat = batch.post();

if (!stat.isError())
}
                              //Added DHCP criteria successfully.
}
```

Step 2 Change the DOCSIS modem's DHCP criteria to that specified in modemCriteria. (See Step 1.)

```
get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// This use case assumes that the DOCSIS modem has been previously added to the database

changeDOCSISModemDHCPCriteria(deviceID, // Modem's MAC address or FQDN
                              "modemCriteria");

BatchStatus stat = batch.post();

if (!stat.isError())
{

          // Configuration generation and device reset is successful.
}
```

Step 3 The modem gets an IP address from the scope targeted by modemCriteria.

Change System Defaults

Case: A service provider wants to enable Promiscuous mode for all CPEs.

Desired Outcome: A service provider can add DOCSIS modems and support promiscuous CPEs by enabling Promiscuous mode on a system level basis.

Step 1 Use the changeSystemDefaults() command and enable the PROMISCUOUS_MODE_ENABLED property on the system level.

```
get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

Map newProp = new HashMap();

// This enables Promiscuous mode on a system level.
// You can override this value by setting PROMISCUOUS_MODE_ENABLED to false
// on a per device basis.

NewProp.add(ModemKeys.PROMISCUOUS_MODE_ENABLED,"enabled");

changeSystemDefaults(newProp, // Adds/Replaces these properties
                              null ); // No properties to remove
```

Note You can override these settings by disabling the Promiscuous mode on individual modems.

Log Device Deletions

Case: A service provider has multiple provisioning clients and wants to log device deletions.

Desired Outcome: When any provisioning client deletes a device, the provisioning client logs an event in one place.

Step 1 Create a listener for the device deletion event. This class must extend the DeviceAdapter abstract class or, alternatively, implement the DeviceListener interface. This class must also override the deletedDevice(DeviceEvent ev) method in order to log the event.

```
public DeviceDeletionLogger                     //Extend the DeviceAdapter class.
    extends DeviceAdapter
{
    public void deletedDevice(DeviceEvent ev)   //Override deletedDevice.
    {
          logDeviceDeletion(ev.getDeviceID())   //Log the deletion.
    }
}
```

Step 2 Register the listener and the qualifier for the events using the PACEConnection interface.

```
DeviceDeletionLogger deviceDeletionLogger = new DeviceDeletionLogger();
qualifier = new DeviceEventQualifier();
qualifier.setDeletedDevice ();
conn.addDeviceListener(deviceDeletionLogger, qualifier);
```

Step 3 When a device is deleted from the system, the event is fired, and the listener is notified.

Monitor a Connection to the RDU

Case: A service provider is running a single provisioning client and wants notification if the connection between the provisioning client and the RDU breaks.

Desired Outcome: The Event interface notifies the service provider if the connection breaks.

Step 1 Create a listener for the messaging event. This class must extend the MessagingAdapter abstract class or, alternatively, implement the MessagingListener interface. This class must override the connectionStopped(MessagingEvent ev) method.

```
// Extend the service provider’s Java program using the provisioning client to receive
// Messaging events.

public MessagingNotifier                                //Extend the MessagingAdapter
       extends MessagingAdapter                         // class.
{
   public void connectionStopped(MessagingEvent ev)     //Override connectionStopped.
   {
       doNotification(ev.getAddress(), ev.getPort());  //Do the notification.
   }
}
```

Step 2 Register the listener and the qualifier for the events using the PACEConnection interface.

```
MessagingQualifier qualifier = new MessagingQualifier();
qualifier.setAllPersistentConnectionsDown();

MessagingNotifier messagingNotifier = new MessagingNotifier();
conn.addMessagingListener(messagingNotifier, qualifier);
```

Step 3 If a connection breaks, the event is fired, and the listener is notified.

Log Batch Completions

Case: A service provider has multiple provisioning clients and wants to log batch completions.

Desired Outcome: When any provisioning client complete a batch, an event is logged in one place.

Step 1 Create a listener for the event. This class must extend the BatchAdapter abstract class or implement the BatchListener interface. This class must override the completion(BatchEvent ev) method in order to log the event.

```
public BatchCompletionLogger                                  //Extend the BatchAdapter
        extends BatchAdapter                                  // class.
{
    public void completion(BatchEvent ev)                     //Override completion.
    {
        logBatchCompletion(ev.BatchStatus().getBatchID());    //Log the completion.
    }
}
```

Step 2 Register the listener and the qualifier for the events using the PACEConnection interface.

```
BatchCompletionLogger batchCompletionLogger = new BatchCompletionLogger();
QualifyAll qualifier = new QualifyAll();
conn.addBatchListener(batchCompletionLogger, qualifier);
```

Step 3 When a batch completes, the event is fired, and the listener is notified.

Get Detailed Information for a Device

Case: A service provider wants to allow an administrator to view detailed information for a particular device.

Desired Outcome: The service provider's administrative application displays all known details about a given device, including MAC address, lease information, provisioned status of the device, and the device type (if known).

Step 1 The administrator enters either the MAC address or the IP address for the device being queried into the service provider's administrative user interface.

Step 2 BPR queries the embedded database for the details for that device.

```
// MSO admin UI calls the provisioning API to query the details for the
// requested device. Query may be performed based on MAC address or IP address,
// depending on what is known about the device.
Map deviceDetails = getDetailsForIPDevice(
          "1,6,00:11:22:33:44:55");   // macAddress: unique identifier for the device
// - OR -
Map deviceDetails = getDetailsForIPDeviceByIPAddress(
          "10.0.17.129");             // ipAddress
```

Step 3 The service provider's application then presents a detail page of device data, which can display everything that is known about the requested device. If the device has been connected to the service provider's network, this data will include lease information (for example, IP address, relay agent identifier). The data indicates whether or not the device has been provisioned, and, if it has, the data also includes the device type.

```
// extract device detail data from the map
String deviceType = (String)deviceDetails.get(DEVICE_TYPE);
String macAddress = (String)deviceDetails.get(MAC_ADDRESS);
String ipAddress = (String)deviceDetails.get(IP_ADDRESS);
String relayAgentID = (String)deviceDetails.get(RELAY_AGENT_ID);
Boolean isProvisioned = (Boolean)deviceDetails.get(IS_PROVISIONED);

// The admin UI now formats and prints the detail data to a view page
```

Retrieve Devices Matching a Vendor Prefix

Case: A service provider wants to allow an administrator to view data for all devices matching a particular vendor prefix.

Desired Outcome: The service provider's administrative application returns a list of devices matching the requested vendor prefix.

Step 1 The administrator enters the substring matching the desired vendor prefix into the service provider's administrative user interface.

Step 2 BPR queries the embedded database for a list of all MAC addresses for the devices that match the requested vendor prefix.

```
// MSO admin UI calls the provisioning API to query all devices for a specified
// vendor prefix.
List deviceList = getIPDevicesByMACAddress(
     "1,6,ff:00:ee:*",             // macAddressPattern: the requested vendor prefix
     10000);                       // maximumReturned:
```