Broadband Access Center for Cable のアプリケーション プログラミング インターフェイスの使用例

この付録では、プロビジョニング API の最も一般的な使用例を列挙します。この使用例には、サービス プロバイダーの一般的なワークフローをモデル化するときに使用できる擬似コード セグメントなどがあります。使用例で使われている擬似コードは java に似ていますが、直接コンパイルできるようにはなっていません。詳細や、個々の API コールと機能を説明するサンプルの Java コード セグメントについては、BACC 2.7 API Javadoc を参照してください。

次の使用例は、デバイス（およびサービス）のプロビジョニングに直接関連しています。サービス クラス、DHCP 基準、およびライセンスの管理など、多くの管理操作についてはここでは取り上げません。関連する API コールの詳細については、API javadoc を参照することを強くお勧めします。また、管理者のユーザ インターフェイスを使用して、これらのアクティビティのほとんどを実行することもできます。

使用例

この付録には、次の使用例が含まれています。

固定標準モードでのセルフプロビジョニングされたモデムとコンピュータ

加入者は、建物内にコンピュータを設置し、DOCSIS ケーブル モデムを購入しています。コンピュータには Web ブラウザがインストールされています。

求める結果

次のワークフローを使用して、プロビジョニングされていない新しい DOCSIS ケーブル モデムとコンピュータをオンラインにし、適切なレベルのサービスを受けられるようにします。

ステップ 1　加入者が DOCSIS ケーブル モデムを購入して建物内に設置し、コンピュータをケーブル モデムに接続します。

ステップ 2　加入者がモデムとコンピュータの電源をオンにし、BACC が制限付きアクセス権をモデムに与えます。コンピュータとモデムには、制限付きアクセス プールから IP アドレスが割り当てられます。

ステップ 3　加入者が Web ブラウザを起動します。スプーフィング DNS サーバにより、Web ブラウザがサービス プロバイダーの登録サーバ（OSS ユーザ インターフェイスやメディエータなど）にアクセスします。

ステップ 4　加入者がサービス プロバイダーのユーザ インターフェイスを使用して、サービス クラスの選択など、登録に必要な手順を完了します。

ステップ 5　サービス プロバイダーのユーザ インターフェイスが、加入者の情報（選択されたサービス クラスやコンピュータの IP アドレスなど）を BACC に渡します。次に、BACC が加入者のモデムとコンピュータを登録します。

```
// 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

LeaseResults computerLease =

       getAllForIPAddress("10.0.14.38");
       // ipAddress: restricted access.

// Derive the modem MAC address from the computer's network
// information. The 1,6, is a standard prefix for an Ethernet
// device. The fully qualified MAC address is required by BACC

String modemMACAddress = "1,6," +
       computerLease.getSingleLease().get(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:

get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

add(
        DeviceType: DOCSIS,
                // deviceType: DOCSIS
        modemMACaddress,
                // macAddress: derived from computer lease
        null,
                // hostName: not used in this example
        null,
                // domainName: not used in this example
        "0123-45-6789",
                // ownerID: here, account number from billing system
        "Silver",
                // ClassOfService
        "provisionedCM",
                // DHCP Criteria: Network Registrar uses this to
                // select a modem lease granting provisioned IP address
        null
                // properties: not used
        );
                // properties: not used
String computerMACAddress = computerLease.
        getSingleLease().get(DeviceDetailsKeys.MAC_ADDRESS);

// 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). This makes
// the computer "fixed" behind the specified modem.

properties.put(IPDeviceKeys.MUST_BE_BEHIND_DEVICE,
                modemMACaddress);

add(
        DeviceType.COMPUTER,
                // deviceType: COMPUTER
```

```
        computerMACAddress,
                // macAddress: derived from computer lease
        null,
                // hostName: not used in this example
        null,
                // domainName: not used in this example
        "0123-45-6789",
                // ownerID: here, account number from billing system
        null,
                // classOfService : get the default COS
        "provisionedCPE",
                // DHCP Criteria: Network Registrar uses this to
                // select a modem lease granting provisioned IP address
        properties
                // properties:
        );
```

ステップ 6　ユーザ インターフェイスにより、加入者はコンピュータをリブートするように求められます。

ステップ 7　プロビジョニング　クライアントが　*performOperation(DeviceOperation.RESET,　modemMACaddress, null)* を呼び出してモデムをリブートし、プロビジョニングされたアクセス権をモデムに与えます。

```
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.

performOperation(
        DeviceOperation.RESET,
                //deviceOperation: Reset operation
        modemMACaddress,
                // macAddress:Modem's MAC address
        null
                // properties: not used
        );
```

ステップ 8　リブート後、コンピュータが新しい IP アドレスを受信します。これで、ケーブル モデムとコンピュータの両方が、プロビジョニングされたデバイスとなります。コンピュータは、サービス プロバイダーのネットワークを介してインターネットにアクセスできます。

固定標準モードでの新しいコンピュータの追加

Multiple Service Operator（MSO; マルチプル サービス オペレータ）が、加入者に対し、ケーブル モデムの背後に 2 台のコンピュータを接続することを許可しています。加入者は 1 台のコンピュータをすでに登録しています。今回は、職場からラップトップ コンピュータを持ち帰ってアクセスします。加入者がハブを設置し、ラップトップをハブに接続します。

求める結果

次のワークフローを使用して、プロビジョニングされていない新しいコンピュータを、すでにプロビジョニングされているケーブル モデムでオンラインにし、新しいコンピュータが適切なレベルのサービスを受けられるようにします。

ステップ 1　加入者が新しいコンピュータの電源をオンにし、BACC が制限付きアクセス権をコンピュータに与えます。

ステップ 2　加入者が新しいコンピュータの Web ブラウザを起動します。スプーフィング DNS サーバにより、Web ブラウザがサービス プロバイダーの登録サーバ（OSS ユーザ インターフェイスやメディエータなど）にアクセスします。

ステップ 3　加入者がサービス プロバイダーのユーザ インターフェイスを使用して、新しいコンピュータの追加に必要な手順を完了します。

ステップ 4　サービス プロバイダーのユーザ インターフェイスが、加入者の情報（選択されたサービス クラスやコンピュータの IP アドレスなど）を BACC に渡します。次に、BACC が加入者のモデムとコンピュータを登録します。

```
// First we query the computer's lease information to its
// 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.

LeaseResults computerLease =
       getAllForIPAddress("10.0.14.39");
       // ipAddress: restricted access.

String computerMACAddress = computerLease.
       getSingleLease().get(DeviceDetailsKeys.MAC_ADDRESS);
```

```
// We have the MAC address now. Let's add the computer to BACC.

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
// can not 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) and hence the
// fixed mode.

properties.put(IPDeviceKeys.MUST_BE_BEHIND_DEVICE, modemMACaddress);

// The IPDeviceKeys.MUST_BE_IN_PROV_GROUP property ensures that
// the computer will receive its provisioned access only when
// it's brought up in the specified provisioning group. This prevents
// the computer (and/or the modem) from moving from one locality to
// to another locality.

properties.put(IPDeviceKeys.MUST_BE_IN_PROV_GROUP, "bostonProvGroup");

add(
        DeviceType.COMPUTER,
                // deviceType: COMPUTER
        computerMACAddress,
                // macAddress: derived from computer lease
        null,
                // hostName: not used in this example
        null,
                // domainName: not used in this example
        "0123-45-6789",
                // ownerID: here, account number from billing system
        null,
                // classOfService: get the default COS
        "provisionedCPE",
                // DHCP Criteria: Network Registrar uses this to
                // select a modem lease granting provisioned IP address
        properties
                // properties:
        );
```

ステップ 5　ユーザ　インターフェイスにより、加入者は新しいコンピュータをリブートするように求められます。リブートすると、BACC が登録済みのサービス レベルをコンピュータに与えます。

ステップ 6　これで、コンピュータはプロビジョニングされたデバイスとなり、適切なレベルのサービスにアクセスできます。

加入者のディセーブル化

加入者が滞納を繰り返したという理由により、サービス プロバイダーが加入者のインターネット アクセスをディセーブルにします。

求める結果

次のワークフローを使用して、動作中のケーブル モデムとコンピュータをディセーブルにします。この結果、デバイスがユーザのインターネット アクセスを一時的に制限します。また、この使用例では、ユーザのブラウザを特別なページにリダイレクトして、次のようなメッセージを通知することも可能です。

```
You haven't paid your bill so your Internet access has been disabled.
```

ステップ 1　サービス プロバイダーのアプリケーションが、プロビジョニング クライアント プログラムを使用して、加入者のデバイスすべてのリストを BACC に要求します。次に、サービス プロバイダーのアプリケーションが、プロビジョニング クライアントを使用して、加入者のデバイスを個別にディセーブルにするか、または制限します。

```
// 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.

getAllForOwnerID(

"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 may 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, it will
                // be disabled when it is turned back on.

                // Let's change the COS of the device so that it will restrict
                // the bandwidth usage of the modem.

                changeClassOfService(

                        Device.MAC_ADDRESS,

                                // macAddress: unique identifier for this modem
```

```
        "DisabledCOS");
                // newClassOfService: restricts bandwidth usage

// Let's change the DHCP Crtieria so that the modem will get an IP
// address from a disabled CNR scope. This scope also points to
// a spoofing DNS server so that the subscriber gets a restricted
// access page.

changeDHCPCriteria(
        Device.MAC_ADDRESS,
                // macAddress: unique identifier for this modem
        "DisabledDHCPCriteria");
                // newDHCPCriteria: disables Internet access
        }
}
// end DeviceLoop
```

（注） DisabledCOS の特性を定義し、モデムをリセットするときは、モデムの背後の CPE に及ぼす影響の考慮が必要となる場合があります。このことは、モデムの背後に音声エンドポイントが接続されている場合に特に重要です。これは、ケーブル モデムの動作を中断すると、その時点で進行中の電話会話に影響を及ぼす可能性があるためです。

これで、加入者がディセーブルになります。

モデムおよびセルフプロビジョニングされたコンピュータの事前プロビジョニング

新しい加入者がサービス プロバイダーに連絡し、サービスを要求します。加入者は、建物内にコンピュータを設置しています。サービス プロバイダーが、すべてのケーブル モデムをまとめて事前プロビジョニングします。

求める結果

次のワークフローを使用して、事前プロビジョニングされたケーブル モデムとプロビジョニングされいないコンピュータを、ローミング標準モードでオンラインにします。この作業は、両方のデバイスが適切なレベルのサービスを受けられ、登録されるようにするために必要です。

ステップ 1　サービス プロバイダーが、課金システムで使用する加入者のユーザ名とパスワードを選択します。

ステップ 2　サービス プロバイダーが、加入者にアクセスを許可するサービスを選択します。

ステップ 3　サービス プロバイダーの現場技術者が、新しい加入者の建物内に物理ケーブルを敷設し、事前プロビジョニングされたデバイスを設置して、加入者のコンピュータに接続します。

ステップ 4　技術者がモデムの電源をオンにし、BACC がプロビジョニングされた IP アドレスをモデムに与えます。

ステップ 5　技術者がコンピュータの電源をオンにし、BACC がプライベート IP アドレスをコンピュータに与えます。

ステップ 6　技術者がコンピュータのブラウザ アプリケーションを起動し、ブラウザがサービス プロバイダーのユーザ インターフェイスにアクセスします。

ステップ 7　技術者がサービス プロバイダーのユーザ インターフェイスにアクセスし、プロビジョニングされたケーブル モデムの背後にあるコンピュータの登録に必要な手順を完了します。

```
// To provision a computer:
// First we query the computer's lease information to its
// MAC address. We use the computers IP address (the web browser
// received this when the subscriber opened the service provider's
// 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

LeaseResults computerLease =
        getAllForIPAddress("10.0.14.38");
        // ipAddress:
String computerMACAddress = computerLease.
        getSingleLease().get(DeviceDetailsKeys.MAC_ADDRESS);
```

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

get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// NO_ACTIVATION will generate a new configuration for the computer
// however it will not attempt to reset it.
// NO_CONFIRMATION is the confirmation mode because we are not
// attempting to reset the computer because this cannot be done.

add(
        DeviceType.COMPUTER,
                // deviceType: Computer
        computerMACAddress,
                // macAddress: derived from computer lease
        null,
                // hostName: not used in this example
        null,
                // domainName: not used in this example
        "0123-45-6789",
                // ownerID: here, account number from billing system
        null,
                // ClassOfService : get the default COS
        "provisionedCPE",
                // DHCP Criteria: Network Registrar uses this to
                // select a modem lease granting provisioned IP address
        null
                // properties: not used
);
```

（注）　IPDeviceKeys.MUST_BE_BEHIND_DEVICE プロパティはコンピュータに設定されませんが、このプロパティを使用すると、ケーブル モデムの背後から別のケーブル モデムへのローミングが可能になります。

ステップ 8　技術者がコンピュータを再起動し、コンピュータがプロビジョニングされた新しい IP アドレスを受信します。これで、ケーブル モデムとコンピュータはいずれもプロビジョニングされたデバイスとなります。コンピュータは、サービス プロバイダーのネットワークを介してインターネットにアクセスできます。

既存のモデムの修正

サービス プロバイダーの加入者が、現在 **Silver** というレベルのサービスを受けていますが、**Gold** サービスにアップグレードすることにしました。加入者は、建物内にコンピュータを設置しています。

（注） この使用例は、デバイスの修正方法を示すことを目的としています。この例は、ローミング標準以外のモードでプロビジョニングされたデバイスに適用可能です。

求める結果

次のワークフローを使用して、既存のモデムのサービス クラスを修正し、そのサービスの変更をサービス プロバイダーの外部システムに渡します。

ステップ 1　加入者がサービス プロバイダーに電話をかけて、サービスのアップグレードを依頼します。サービス プロバイダーはユーザ インターフェイスを使用して、サービス クラスを **Silver** から **Gold** に変更します。

ステップ 2　サービス プロバイダーのアプリケーションが、BACC で次の API コールを行います。

```
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.

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

ステップ 3　これで、加入者がサービス プロバイダーのネットワークにアクセスし、*Gold* サービスを受けられるようになります。

加入者のデバイスの登録解除と削除

サービス プロバイダーが、サービスの利用を中止した加入者を削除します。

求める結果

次のワークフローを使用して、加入者のデバイスすべてをサービス プロバイダーのネットワークから完全に削除します。

ステップ 1　サービス プロバイダーのユーザ インターフェイスが、加入者へのサービスを停止します。

ステップ 2　サービス プロバイダーのアプリケーションが、プロビジョニング クライアント プログラムを使用して、加入者のデバイスすべてのリストを BACC に要求します。次に、各デバイスを登録解除してリセットします。その結果、各デバイスがデフォルトの（プロビジョニングされていない）サービス レベルに低下します。

（注）　「unregister」API へのパラメータに指定されたデバイスがすでに登録解除された状態にある場合は、API コールからのステータス コードが CommandStatusCodes .CMD_ERROR_DEVICE_UNREGISTER_UNREGISTERED_ERROR に設定されます。これは、通常および所定の動作です。

```
// 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.

getAllForOwnerID(

        "0123-45-6789");
        // query all the devices for this account number

// We need to unregister all the devices behind each modem(s) or else the
// unregister call for that modem will fail.

// for each computer in the retrieved list:
DeviceLoop:
{

        if (Device.deviceType == COMPUTER)
        {
                get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

                // NO_ACTIVATION is the activation mode because we can't
                // to reset Computers.
                // NO_CONFIRMATION is the confirmation mode because
                // the unregister call will not reset the devices.

                unregister(

                        Device.MAC_ADDRESS,
                        // macAddress: unique identifier for this device

                );

        }
```

```
}

// for each modem in the retrieved list:
DeviceLoop:
{

        if (Device.deviceType == DOCSIS_MODEM)
        {
                get-new-batch(AUTOMATIC, NO_CONFIRMATION)

                // AUTOMATIC is the activation mode because we want
                // to reset as we unregister the device.

                unregister(
                        Device.MAC_ADDRESS,
                        // macAddress: unique identifier for this device
                );
        }
}

// end DeviceLoop.
```

（注） サービス プロバイダーによっては、故障の場合を除いてケーブル モデムをデータベースに残す場合があるため、次のステップはオプションです。次のステップが必要になるのは、デバイスをデータベースから削除する場合のみです。

ステップ 3　サービス プロバイダーのアプリケーションが、プロビジョニング クライアント プログラムを使用して、加入者の残りのデバイスをデータベースから個別に削除します。

```
// 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.

getAllForOwnerID(
        "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

        if (Device.deviceType == DOCSIS_MODEM)
        {
                get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

                // NO_ACTIVATION is the activation mode because we don't want
                // to reset as we are deleting the device.
                // NO_CONFIRMATION is the confirmation mode because
                // the delete call will not reset the devices.

                delete(
                        Device.MAC_ADDRESS,
                        // macAddress: unique identifier for this device
                        true
                        // deleteDevicesBehind: deletes CPEs behind this modem.
                );

        }
}
//end DeviceLoop.
```

無差別モードでの最初のアクティベーションのセルフプロビジョニング

加入者が、建物内にコンピュータ（ブラウザ アプリケーションを持つ）を設置し、DOCSIS ケーブル モデムを購入しています。

求める結果

次のワークフローを使用して、プロビジョニングされていない新しい DOCSIS ケーブル モデムとコンピュータをオンラインにし、適切なレベルのサービスを受けられるようにします。

ステップ 1　加入者が DOCSIS ケーブル モデムを購入し、建物内に設置します。

ステップ 2　加入者がモデムの電源をオンにし、BACC が制限付きアクセス権をモデムに与えます。

ステップ 3　加入者がコンピュータのブラウザ アプリケーションを起動します。スプーフィング DNS サーバにより、ブラウザがサービス プロバイダーの登録サーバ（OSS ユーザ インターフェイスやメディエータなど）にアクセスします。

ステップ 4　加入者がサービス プロバイダーのユーザ インターフェイスを使用して、サービス クラスの選択など、登録に必要な手順を完了します。

サービス プロバイダーのユーザ インターフェイスが、加入者の情報（選択されたサービス クラスやコンピュータの IP アドレスなど）を BACC に渡します。次に、加入者のケーブル モデムとコンピュータが BACC に登録されます。

```
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 computer’s IP address (the web browser
// received this when the subscriber opened the service
// provider’s web interface). We also assume that “bostonProvGroup”
// is the provisioning group  used in that locality.

List provGroupList;
provGroupList = provGroupList.add(“bostonProvGroup”);
Map computerLease = getAllForIPAddress(
        "10.0.14.38",
        // ipAddress: restricted access computer lease
        provGroupList
        // provGroups: List containing provgroup)

// Derive the modem MAC address from the computer's network

// information.The 1,6, is a standard prefix for an Ethernet

// device.The fully qualified MAC address is required by BACC


String modemMACAddress = "1,6," +
        computerLease.GetSingleLease().get(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");

properties.put(ModemKeys.CPE_DHCP_CRITERIA, "provisionedCPE");

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

ステップ 5　ユーザ インターフェイスにより、加入者はコンピュータをリブートするように求められます。

ステップ 6　プロビジョニング クライアントが *performOperation(...)* を呼び出してモデムをリブートし、プロビジョニングされたアクセス権をモデムに与えます。

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

// AUTOMATIC is the activation mode because we are attempting
// to reset the modem so that it receives the 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

performOperation(
        DeviceOperation.RESET,
                //deviceOperation: Reset operation
        modemMACaddress,
                // macAddress:Modem's MAC address
        null
                // properties : not used
        );
```

ステップ 7　コンピュータをリブートすると、コンピュータが新しい IP アドレスを受信します。これで、ケーブル モデムがプロビジョニングされたデバイスとなります。コンピュータは BACC に登録されていませんが、サービス プロバイダーのネットワークを介してインターネットにアクセスできます。無差別モードのモデムの背後でオンラインになっているコンピュータは、引き続き、プロビジョニング API を使用して利用することができます。

無差別モードでの 100 個のモデムの一括プロビジョニング

サービス プロバイダーにおいて、サービスセンターのカスタマー サービス担当者が、配送する 100 個のケーブル モデムを事前プロビジョニングします。

求める結果

次のワークフローを使用して、すべてのモデムのモデム データを新しい加入者に割り当てます。カスタマー サービス担当者は、割り当て可能なモデムのリストを持っています。

ステップ 1　サービス プロバイダーの発送センターで、新しいケーブル モデムまたは再利用するケーブル モデムの MAC アドレス データがリストにまとめられます。

ステップ 2　特定のサービスセンターに割り当てられたモデムが BACC に一括ロードされ、そのサービスセンターの識別名が付けられます。

ステップ 3　サービスセンターでモデムを新しい加入者に割り当てる際、カスタマー　サービス担当者が新しいサービス パラメータを入力し、モデムの Owner ID フィールドを新しい加入者のアカウント番号に合せて変更します。

```
// 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 yet. A configuration can't be generated if no
// network information is present. And because the devices
// have not booted, they are not online and therefore cannot
// be reset. NO_CONFIRMATION is the confirmation mode because
// we are not attempting to reset the devices.
// Create a Map for the properties of the modem


Map properties;

// Set the property 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")

// The ModemKeys.CPE_DHCP_CRITERIA is used to specify the DHCP
// Criteria to be used while selecting IP address scopes for
// CPEs behind this modem in the promiscuous mode.

properties.put(ModemKeys.CPE_DHCP_CRITERIA, "provisionedCPE");

// for each modem MAC-address in list:

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

無差別モードでの最初のアクティベーションの事前プロビジョニング

新しい加入者がサービス プロバイダーに連絡し、サービスを要求します。加入者は、建物内にコンピュータを設置しています。

求める結果

次のワークフローを使用して、プロビジョニングされていない新しいケーブル モデムとコンピュータをオンラインにし、適切なレベルのサービスを受けられるようにします。

ステップ 1　サービス プロバイダーが、課金システムで使用する加入者のユーザ名とパスワードを選択します。

ステップ 2　サービス プロバイダーが、加入者にアクセスを許可するサービスを選択します。

ステップ 3　サービス プロバイダーが、独自のユーザ インターフェイスを使用して、デバイスを登録します。

ステップ 4　サービス プロバイダーのユーザ インターフェイスが、モデムの MAC アドレスやサービス クラスなどの情報を BACC に渡します。また、モデムが CPE DHCP 基準設定を取得します。この基準設定により、Network Registrar はモデムの背後に接続するコンピュータに対して、プロビジョニングされたアドレスを選択できます。次に、新しいモデムが BACC に登録されます。

ステップ 5　サービス プロバイダーの現場技術者が、新しい加入者の建物内に物理ケーブルを敷設し、事前プロビジョニングされたデバイスを設置して、加入者のコンピュータに接続します。

// 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 cannot be reset.

// NO_CONFIRMATION is the confirmation mode because we are not

// attempting to reset the modem.

```
// Create a map for the properties of the modem.
Map properties;
// Set the property ModemKeys.PROMISCUOUS_MODE_ENABLED to enable
```

// promiscuous mode on modem

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

properties.put(ModemKeys.CPE_DHCP_CRITERIA, "provisionedCPE");

```
add(
        DeviceType.DOCSIS,
                // deviceType: DOCSIS
        "1,6,00:11:22:33:44:55",
                // macAddress: derived from computer lease
        null,
                // hostName: not used in this example
        null,
                // domainName: not used in this example
        "0123-45-6789",
                // ownerID: here, account number from billing system
        "Silver",
                // ClassOfService
        "provisionedCM",
```

```
                // DHCP Criteria: Network Registrar uses this to
                // select a modem lease granting provisioned IP address
        properties
                // properties:
);
```

ステップ 6　技術者がケーブル モデムの電源をオンにし、BACC がプロビジョニングされたアクセス権をケーブル モデムに与えます。

ステップ 7　技術者がコンピュータの電源をオンにし、BACC　がプロビジョニングされたアクセス権をコンピュータに与えます。これで、ケーブル モデムとコンピュータはいずれもプロビジョニングされたデバイスとなります。コンピュータは、サービス プロバイダーのネットワークを介してインターネットにアクセスできます。

既存のモデムの交換

サービス プロバイダーが、故障したモデムを交換します。

（注）　モデムから別のモデムへのローミングを制限するオプションがコンピュータに設定されている場合、モデムを交換したときは、コンピュータに設定されているモデムの MAC アドレスも変更する必要があります。

求める結果

次のワークフローを使用して、加入者に提供するサービスのレベルを変更することなく、既存のケーブル モデムを新しいモデムに物理的に交換します。

ステップ 1　サービス プロバイダーが、既存のモデムの MAC アドレスを新しいモデムの MAC アドレスに変更します。

get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

```
// NO_ ACTIVATION is the activation mode because we will
// not be able to reset as 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:

changeMACAddress (
        "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
);
```

ステップ 2　サービス プロバイダーが、ケーブル モデムを交換し、その電源をオンにします。コンピュータの電源もオンにする必要があります。

ステップ 3　これで、ケーブル モデムが完全にプロビジョニングされたデバイスとなり、適切なレベルのサービスを受けられるようになります。ケーブル モデムの背後にあるコンピュータも同様です。

無差別モードでの 2 台目のコンピュータの追加

加入者が、設置されているケーブル モデムの背後に 2 台目のコンピュータを接続します。

求める結果

次のワークフローを使用して、加入者の選択したサービスが複数の CPE の接続を許可すること、および接続された両方のコンピュータから加入者がネットワークにアクセスできることを保証します。

（注）　この事例では、プロビジョニング API の呼び出しは必要ありません。

ステップ 1　加入者が、ケーブル モデムの背後に 2 台目のコンピュータを接続します。

ステップ 2　加入者がコンピュータの電源を入れます。

ステップ 3　加入者の選択したサービスが複数の CPE の接続を許可すると、BACC がインターネットへのアクセス権を 2 台目のコンピュータに与えます。

NAT を使用した最初のアクティベーションのセルフプロビジョニング

大学が、Network Address Translation（NAT; ネットワーク アドレス変換）および DHCP 機能を持つ DOCSIS ケーブル モデムを購入しました。建物の利用者 5 人がそれぞれ、コンピュータにブラウザ アプリケーションをインストールしています。

求める結果

次のワークフローを使用して、プロビジョニングされていない新しいケーブル モデム（NAT を持つ）と、モデムの背後にあるコンピュータをオンラインにし、適切なレベルのサービスを受けられるようにします。

ステップ 1　加入者が、NAT および DHCP 機能を持つケーブル モデムを購入し、集合建物内に設置します。

ステップ 2　加入者がモデムの電源をオンにし、BACC が制限付きアクセス権をモデムに与えます。

ステップ 3　加入者がラップトップ コンピュータをケーブル モデムに接続し、モデム内の DHCP サーバが IP アドレスをラップトップに与えます。

ステップ 4　加入者がコンピュータのブラウザ アプリケーションを起動します。スプーフィング DNS サーバにより、ブラウザがサービス プロバイダーの登録サーバ（OSS ユーザ インターフェイスやメディエータなど）にアクセスします。

ステップ 5　加入者がサービス プロバイダーのユーザ インターフェイスを使用して、ケーブル モデムの登録に必要な手順を完了します。登録用のユーザ インターフェイスが、モデムで NAT が使用されていることを検出し、モデムを登録して、モデムが NAT 互換のサービス クラスを受けられることを確認します。

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.

LeaseResults modemLease =
        getAllForIPAddress("10.0.14.38");
        // ipAddress: restricted access.


String modemMACAddress = modemLease.
        getSingleLease().get(DeviceDetailsKeys.MAC_ADDRESS);
```

// 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

add(
        DeviceType.DOCSIS,
                // deviceType: DOCSIS
        modemMACAddress,
                // macAddress: derived from its lease
        null,
                // hostName: not used in this example
        null,
                // domainName: not used in this example
        "0123-45-6789",
                // ownerID: here, account number from billing system
        "Silver",
                // ClassOfService
        "provisionedCM",
                // DHCP Criteria: Network Registrar uses this to
                // select a modem lease granting provisioned IP address
        null
                // properties: not used
);
```

ステップ 6　ユーザ インターフェイスにより、加入者はコンピュータをリブートするように求められます。

ステップ 7　プロビジョニング クライアントが *performOperation(...)* を呼び出してモデムをリブートし、プロビジョニングされたアクセス権をモデムに与えます。

```
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


performOperation(
        DeviceOperation.RESET,
                //deviceOperation: Reset operation
        "1,6,00:11:22:33:44:55",
                // macAddress:Modem's MAC address
        null
                // properties : not used
        );
```

ステップ 8　これで、ケーブル モデムが完全にプロビジョニングされ、モデムの背後にあるコンピュータがサービス プロバイダーのネットワークにフル アクセスできるようになります。

（注）　NAT を持つ特定のケーブル モデムでは、新しいサービス クラス設定を取得するためにコンピュータをリブートするように求められる場合があります。ケーブル モデムと NAT デバイスが別々のデバイスになっている場合は、NAT デバイスもコンピュータの登録と同じように登録する必要があります。

NAT を持つモデムの背後への新しいコンピュータの追加

アパートの大家が 4 人の借家人に対し、モデムの共有と、サービス プロバイダーのネットワークへのアクセスを提供しています。この大家が新しい借家人に対し、建物のモデムを共有したインターネット アクセスを提供します。モデムには NAT および DHCP 機能が含まれています。新しい借家人が、モデムにコンピュータを接続します。

求める結果

次のワークフローを使用して、プロビジョニングされていない新しいコンピュータを、すでにプロビジョニングされているケーブル モデムでオンラインにし、新しいコンピュータが適切なレベルのサービスを受けられるようにします。

（注）　この事例では、プロビジョニング API の呼び出しは必要ありません。

ステップ 1　加入者がコンピュータの電源を入れます。

ステップ 2　これで、コンピュータはプロビジョニングされたデバイスとなり、適切なレベルのサービスにアクセスできます。

（注）　プロビジョニングされた NAT モデムは、モデムの背後にあるコンピュータをネットワークから見えなくします。

別の DHCP スコープへのデバイスの移動

サービス プロバイダーが、ネットワークに番号を再割り当てします。これに伴い、登録済みケーブル モデムには、別の Network Registrar スコープの IP アドレスが必要になります。

求める結果

プロビジョニング クライアントが DHCP 基準を変更し、ケーブル モデムが対応する DHCP スコープから IP アドレスを受信します。

ステップ 1　DOCSIS モデムの DHCP 基準を「newmodemCriteria」に変更します。

```
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.

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

changeDHCPCriteria(
        "1,6,ff:00:ee:11:dd:22"
                // Modem's MAC address or FQDN
                "newmodemCriteria"
        );
```

ステップ 2　モデムが、「newmodemCriteria」によって対象とされたスコープから IP アドレスを取得します。

イベントを使用したデバイス削除のロギング

サービス プロバイダーが、複数のプロビジョニング クライアントを保有しており、デバイス削除をログに記録します。

求める結果

どのプロビジョニング クライアントがデバイスを削除しても、プロビジョニング クライアントはイベントを 1 か所のログに記録します。

ステップ 1　デバイス削除イベントのリスナーを作成します。このクラスは、*DeviceAdapter* 抽象クラスを拡張するか、または *DeviceListener* インターフェイスを実装する必要があります。さらに、このクラスは、イベントをログに記録するために *deletedDevice(DeviceEvent ev)* メソッドを無効にする必要もあります。

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

ステップ 2　*PACEConnection* インターフェイスを使用して、イベントのリスナーと修飾子を登録します。

```
DeviceDeletionLogger deviceDeletionLogger =
       new DeviceDeletionLogger();
               // Modem's MAC address or FQDN
               "newmodemCriteria"
qualifier = new DeviceEventQualifier();
// We are interested only in device deletion.
qualifier.setDeletedDevice ();
// Add device listener using PACEConnection
connection.addDeviceListener(deviceDeletionLogger, qualifier
);
```

ステップ 3　システムからデバイスが削除されると、イベントが生成され、リスナーに通知されます。

イベントを使用した RDU 接続の監視

サービス プロバイダーが、1 つのプロビジョニング クライアントを実行しており、プロビジョニング クライアントと RDU 間の接続が切断された場合に通知されるようにします。

求める結果

次のワークフローを使用して、接続が切断された場合にイベント インターフェイスからサービス プロバイダーに通知されるように設定します。

ステップ 1　メッセージ イベントのリスナーを作成します。このクラスは、MessagingAdapter 抽象クラスを拡張するか、または MessagingListener インターフェイスを実装する必要があります。さらに、このクラスは *connectionStopped(MessagingEvent ev)* メソッドを無効にする必要もあります。

```
// Extend the service provider's Java program using the
// provisioning client to receive Messaging events.
public MessagingNotifier
        extends MessagingAdapter
        //Extend the MessagingAdapter class.
{
        public void connectionStopped(MessagingEvent ev)
                //Override connectionStopped.
        {
                doNotification(ev.getAddress(), ev.getPort());
                        //Do the notification.
        }
}
```

ステップ 2　*PACEConnection* インターフェイスを使用して、イベントのリスナーと修飾子を登録します。

```
MessagingQualifier qualifier =
        new MessagingQualifier();
qualifier.setAllPersistentConnectionsDown();
MessagingNotifier messagingNotifier = new MessagingNotifier();
connnection.addMessagingListener(messagingNotifier, qualifier
);
```

ステップ 3　接続が切断されると、イベントが生成され、リスナーに通知されます。

イベントを使用したバッチ完了のロギング

サービス プロバイダーが、複数のプロビジョニング クライアントを保有しており、バッチ完了をログに記録します。

求める結果

どのプロビジョニング クライアントがバッチを完了しても、イベントが 1 か所のログに記録されます。

ステップ 1　イベントのリスナーを作成します。このクラスは、*BatchAdapter* 抽象クラスを拡張するか、または *BatchListener* インターフェイスを実装する必要があります。さらに、このクラスは、イベントをログに記録するために *completion(BatchEvent ev)* メソッドを無効にする必要もあります。

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

ステップ 2　PACEConnection インターフェイスを使用して、イベントのリスナーと修飾子を登録します。

```
BatchCompletionLogger batchCompletionLogger =
        new BatchCompletionLogger();
Qualify All qualifier = new Qualify All();
connection.addBatchListener(batchCompletionLogger , qualifier
);
```

ステップ 3　バッチが完了すると、イベントが生成され、リスナーに通知されます。

デバイスの詳細情報の取得

サービス プロバイダーが管理者に対し、特定のデバイスの詳細情報を表示することを許可します。

求める結果

サービス プロバイダーの管理アプリケーションが、特定のデバイスに関する既知の詳細をすべて表示します。この詳細には、MAC アドレス、リース情報、デバイスのプロビジョニング ステータス、およびデバイス タイプ（既知の場合）などがあります。

ステップ 1　管理者が、サービス プロバイダーの管理ユーザ インターフェイスに、クエリーするデバイスの MAC アドレスを入力します。

ステップ 2　BACC が、デバイスの詳細を組み込みデータベースにクエリーします。

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

// 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 =
        getDetails(
                "1,6,00:11:22:33:44:55",
                        // macORFqdn: unique identifier for the device
                true
                        // needLeaseInfo: yes we need it
                };
```

ステップ 3　サービス プロバイダーのアプリケーションが、デバイス データの詳細に関するページを表示します。このページには、要求されたデバイスに関する既知の情報をすべて表示できます。デバイスがサービス プロバイダーのネットワークに接続された場合、このデータにはリース情報（IP アドレスやリレー エージェントの ID など）が含まれます。データは、デバイスがプロビジョニングされたかどうかを示します。プロビジョニングされた場合、データにはデバイス タイプも含まれます。

```
// 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
```

デフォルトのサービス クラスを使用した検索

サービス プロバイダーが管理者に対し、DOCSIS デバイス タイプの *default* サービス クラスを持つすべてのモデムのデータを表示することを許可します。

求める結果

サービス プロバイダーの管理アプリケーションが、デフォルトのサービス クラスを持つ DOCSIS デバイスのリストを返します。

ステップ 1　管理者が、サービス プロバイダーの管理ユーザ インターフェイスで、検索オプションを選択します。

ステップ 2　BACC が、要求されたデフォルトのサービス クラスに一致するデバイスの MAC アドレスすべてのリストを、組み込みデータベースにクエリーします。

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

// Create a MACAddressSearchType to indicate search by
// default class of service.

MACAddressSearchType mst =
        new MACAddressSearchType.getByDefaultClassOfService(
                                        DeviceType.DOCSIS);
// Create a MACAddressSearchFilter to get 20 devices
// at a time. This indicates that we have a page size of 20.

MACAddressSearchFilter mySearchFilter =
        new MACAddressSearchFilter(
                mst,
                        // type: MAC address search type
                false,
                        // isInclusive:
                20
                        // maximumReturned:
                );
// MAC address to start the search from.
String startMac = null;

// A list containing the MAC addresses returned from
// search.
List deviceList = null;

// If the size of deviceList is equal to 20 then
// there may be devices matching the search criteria.
        while ((deviceList == null) ||
                        (deviceList.size() == 20))
        {
```

```
                // Use the provisioning API call to search BACC
                // database. The search starts from the MAC address
                // "startMac". If "startMac" is null then the search
                // starts from the very begining of the index.

                deviceList = search (mySearchFilter, startMac);

                // See Step 3 for the definition of processMACAddressList

                startMac = processMACAddressList(deviceList);
}
```

ステップ 3　サービス プロバイダーのアプリケーションが、これらのデバイスの詳細を BACC から要求し、デバイス データのページを表示します。デバイスごとに、デバイス タイプ、MAC アドレス、クライアント クラス、およびデバイスのプロビジョニング ステータスがコード表示されます。デバイスは 1 行ずつ表示されます。

```
processMACAddressList (List deviceList)
{
        Iterator iter = deviceList.iterator();

        String startMac = null;

        while (iter.hasNext())
        {
                startMac = (String) iter.next();
                // Get details for this device
                Map detailMap = getDetails (
                        startMac,
                                // MAC of current device
                        true
                                // yes, we need lease info
                        );
                // extract device detail data from each map
                // this can be used for displaying in UI
                String deviceType = (String)detailMap.get(DEVICE_TYPE);
                String macAddress = (String)detailMap.get(MAC_ADDRESS);
                String clientClass = (String)detailMap.get(CLIENT_CLASS);
                Boolean isProvisioned = (Boolean)detailMap.get(IS_PROVISIONED);
                // format and print above data in output line
        }

        // We return the last MAc address in the list
        // so that the next search can be started from here

        return startMAC;
}
```

ベンダープレフィックスに一致するデバイスの取得

サービス プロバイダーが管理者に対し、特定のベンダープレフィックスに一致するすべてのデバイスのデータを表示することを許可します。

求める結果

サービス プロバイダーの管理アプリケーションが、要求されたベンダープレフィックスに一致するデバイスのリストを返します。

ステップ 1　管理者が、サービス プロバイダーの管理ユーザ インターフェイスに、目的のベンダープレフィックスに一致する部分文字列を入力します。

ステップ 2　BACC が、要求されたベンダープレフィックスに一致するデバイスの MAC アドレスすべてのリストを、組み込みデータベースにクエリーします。

```
// Create a MACAddressPattern corresponding to the requested
// vendor prefix

MACAddressPattern pattern =
         new MACAddressPattern(
                "1,6,ff:00:ee:*",
                        // macAddressPattern: the requested vendor prefix
                );
// Create a MACAddressSearchType to indicate search by
// MAC address pattern

MACAddressSearchType mst =
        new MACAddressSearchType.getDevices(pattern);
// Create a MACAddressSearchFilter to get 20 devices
// at a time. This indicates that we have a page size of 20.

MACAddressSearchFilter mySearchFilter =
        new MACAddressSearchFilter(
                mst,
                        // type: MAC address search type
                false,
                        // isInclusive:
                20
                         // maximumReturned:
                );
// MAC address to start the search from.
String startMac = null;

// A list containing the MAC addresses returned from
// search.
List deviceList = null;

// If the size of deviceList is equal to 20 then
// there may be devices matching the search criteria.

while ((deviceList == null) ||

        (deviceList.size() == 20))
{
```

```
        // Use the provisioning API call to search BACC
        // database. The search starts from the MAC address
        // "startMac". If "startMac" is null then the search
        // starts from the very begining of the index.

        deviceList = search (mySearchFilter, startMac);

        // See Step 3 for the definition of processMACAddressList

        startMac = processMACAddressList(deviceList);
}
```

ステップ 3　サービス プロバイダーのアプリケーションが、これらのデバイスの詳細を BACC から要求し、デバイス データのページを表示します。デバイスごとに、デバイス タイプ、MAC アドレス、クライアント クラス、およびデバイスのプロビジョニング ステータスがコード表示されます。デバイスは 1 行ずつ表示されます。

```
processMACAddressList (List deviceList)
{
        Iterator iter = deviceList.iterator();

        String startMac = null;

        while (iter.hasNext())
        {
                startMac = (String) iter.next();
                // Get details for this device
                Map detailMap = getDetails (
                        startMac,
                                // MAC of current device
                        true
                                // yes, we need lease info
                        ) ;

                // extract device detail data from each map
                // this can be used for displaying in UI
                String deviceType = (String)detailMap.get(DEVICE_TYPE);
                String macAddress = (String)detailMap.get(MAC_ADDRESS);
                String clientClass = (String)detailMap.get(CLIENT_CLASS);
                Boolean isProvisioned = (Boolean)detailMap.get(IS_PROVISIONED);
                // format and print above data in output line
                }
        // We return the last MAc address in the list
        // so that the next search can be started from here

        return startMac;
}
```

PacketCable eMTA の事前プロビジョニング

新しい顧客が、サービス プロバイダーに連絡して PacketCable 音声サービスを注文します。顧客は、プロビジョニングされた組み込み型 MTA を受け取ります。

求める結果

次のワークフローを使用して、組み込み型 MTA を事前プロビジョニングし、モデムの MTA コンポーネントがオンラインになったときに適切なレベルのサービスを受けられるようにします。

（注）　次の使用例では、eMTA から電話をかけるのに必要なコール エージェントのプロビジョニングを省略しています。

ステップ 1　サービス プロバイダーが、課金システムで使用する加入者のユーザ名とパスワードを選択します。

ステップ 2　サービス プロバイダーが、モデム コンポーネントに適切なサービス クラスと DHCP 基準を選択し、コンポーネントを BACC に追加します。

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

// Let's provision the modem and the MTA component in the same
// batch. This can be done because the activation mode of this
// batch is NO_ACTIVATION. More than one device can be operated
// on in a batch if the activation mode does not lead to more
// than one device being reset.
// To add a DOCSIS modem:

add(
        DeviceType.DOCSIS,
                // deviceType: DOCSIS
        "1,6,01:02:03:04:05:06",
                // macAddress: scanned from the label
        null
                // hostName: not used in this example
        null
                // domainName: not used in this example
        "0123-45-6789",
                // ownerID: here, account number from billing system
        "Silver",
                // classOfService
        "provisionedCM",
                // DHCP Criteria: Network Registrar uses this to
                // select a modem lease granting provisioned IP address
        null
                // properties: not used
);
```

ステップ 3　サービス プロバイダーが、MTA コンポーネントに適切なサービス クラスと DHCP 基準を選択し、コンポーネントを BACC に追加します。

```
// Continuation of the batch in Step2
// To add the MTA component:

add(
        DeviceType.PACKET_CABLE_MTA,
                // deviceType: PACKET_CABLE_MTA
        "1,6,01:02:03:04:05:07",
                // macAddress: scanned from the label
        null,
                // hostName: not used in this example, will be auto generated
        null,
                // domainName: not used in this example, will be auto generated.
                // The FqdnKeys.AUTO_FQDN_DOMAIN property must be set somewhere in the
                // property hierarchy.
        "0123-45-6789",
                // ownerID: here, account number from billing system
        "Silver",
                // ClassOfService
        "provisionedMTA",
                // DHCP Criteria: Network Registrar uses this to
                // select an MTA lease granting provisioned IP address
        null
                // properties: not used
);
```

ステップ 4　組み込み型 MTA が顧客に出荷されます。

ステップ 5　顧客が、組み込み型 MTA をオンラインにして、電話をかけます。

PacketCable eMTA 上での SNMP クローン作成

顧客が SNMP Element Manager を所有しており、これを使用して PacketCable eMTA にアクセスします。

求める結果

外部の Element Manager に、PacketCable eMTA へのセキュアな SNMPv3 アクセス権が与えられます。

（注）　RW MIB 変数に加えた変更は永続的なものではなく、BACC の eMTA に関する設定において更新されることはありません。eMTA MIB に書き込まれた情報は、次回 MTA を電源オフにするか、またはリセットしたときに失われます。

ステップ 1　プロビジョニング API メソッドである performOperation() を呼び出します。その際は、MTA の MAC アドレスと、MTA 上に作成する新しいユーザのユーザ名を渡します。このユーザ名は、後で Element Manager が SNMP コールを行うときに使用されます。

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

// NO_ACTIVATION is the activation mode because we don't want to
// reset the device.
// NO_CONFIRMATION is the confirmation mode because we are
// not attempting to reset the device.

// The goal here is to create a new user on the MTA indicated
// by the MAC address. The other parameter needed here is the new
// user name, which is passed in the Map.
// Create a map that contains one element - the name of
// the new user to be created on the MTA
HashMap map = new HashMap();
map.put( SNMPPropertyKeys.CLONING_USERNAME, "newUser" );
// The first param is the actual device operation to perform.
performOperation(
        DeviceOperation.ENABLE_SNMPV3_ACCESS,
                // deviceOperation : ENABLE_SNMPV3_ACCESS
        "1,6,00:00:00:00:00:99",
                // macORFqdn : MAC Address of the modem
        map
                // parameters: operation specific parameters
);
```

ステップ 2　プロビジョニング API が、ステップ 1 で渡された新しいユーザに対応するエントリを MTA 上に作成するため、SNMPv3 クローン作成操作の実行を試みます。新しいユーザのエントリ行で使用されるキーは、BACC 内で定義された 2 つのパスワードの関数です。これらのパスワードは、顧客が利用できるようになります。また、RDU コマンドが、これらのパスワード（auth および priv パスワード）をキー ローカリゼーション アルゴリズムに渡して、auth および priv キーを作成します。これらのキーは、新しいユーザとセットで、eMTA のユーザ テーブルに格納されます。

（注）　このステップに記載されている auth および priv パスワードを変更するには、rdu.properties 設定ファイル内の SNMPPropertyKeys.CLONING_AUTH_PASSWORD（/snmp/cloning/auth/password）および SNMPPropertyKeys.CLONING_PRIV_PASSWORD（/snmp/cloning/priv/password）プロパティをそれぞれ設定変更します。

ステップ 3　顧客が、上で指定されたユーザ名、パスワード、および キー ローカリゼーション アルゴリズムを使用して SNMPv3 要求を発行し、MTA とのセキュアな通信を可能にします。

PacketCable eMTA の差分プロビジョニング

顧客が、PacketCable eMTA を使用しており、その 1 つ目の回線（エンドポイント）をイネーブルにしています。この顧客が、eMTA 上の 2 つ目の電話回線（エンドポイント）をイネーブルにし、その回線に電話を接続します。

求める結果

顧客が eMTA 上の 2 つ目の回線（エンドポイント）に電話を接続し、どのサービスも中断することなく正常に電話をかけられるようになる必要があります。

（注）　eMTA 上で 2 つ目の回線を使用するには、コール エージェントを適切に設定する必要があります。コール エージェントのプロビジョニングについては、この使用例では取り上げません。

ステップ 1　サービス プロバイダーのアプリケーションが、BACC API を起動して、eMTA のサービス クラスを変更します。新しいサービス クラスは、eMTA 上の 2 つのエンドポイントをサポートします。このサービス クラスの変更は、eMTA をリセットして初めて有効になります。eMTA を中断することは好ましくありません。そのため、差分プロビジョニングが次のステップで行われます。

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

// NO_ACTIVATION is the activation mode because we don't want to
// reset the device.
// NO_CONFIRMATION is the Confirmation mode because we are not
// disrupting the device.

changeClassOfService(
        "1,6,ff:00:ee:11:dd:22" // eMTA's MAC address or FQDN
,"twoLineEnabledCOS" // This COS supports two lines.
);
```

ステップ 2　サービス プロバイダーのアプリケーションが、BACC 差分更新機能を使用して、eMTA に SNMP オブジェクトを設定します。そのため、eMTA を中断することなく、サービスがイネーブルになります。

```
// The goal here is to enable a second phone line, assuming one
// phone line is currently enabled.  We will be adding a new
// row to the pktcNcsEndPntConfigTable.

get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

// NO_ACTIVATION is the activation mode because we don't want to
// reset the device.
// NO_CONFIRMATION is the confirmation mode because we are
// not attempting to reset the device.
```

```
// Create a map containing one element - the list of SNMP
// variables to set on the MTA
HashMap map = new HashMap();

// Create an SnmpVarList to hold SNMP varbinds
SnmpVarList list = new SnmpVarList();

// An SnmpVariable represents an oid/value/type triple.

// pktcNcsEndPntConfigTable is indexed by the IfNumber, which in this
// case we will assume is interface number 12 (this is the last
// number in each of the oids below.

// The first variable represents the creation of a new row in
// pktcNcsEndPntConfigTable  we are setting the RowStatus
// column (column number 26).  The value of 4 indicates that
// a new row is to be created in the active state.
SnmpVariable variable = new SnmpVariable(
        ".1.3.6.1.4.1.4491.2.2.2.1.2.1.1.26.12",
        "4",
        SnmpType.INTEGER );
list.add( variable );

// The next variable represents the call agent id for this new
// interface, which we'll assume is 'test.com'
SnmpVariable variable = new SnmpVariable(
        ".1.3.6.1.4.1.4491.2.2.2.1.2.1.1.1.12",
        "test.com",
        SnmpType.STRING );
list.add( variable );

// The final variable represents the call agent port
SnmpVariable variable = new SnmpVariable(
        ".1.3.6.1.4.1.4491.2.2.2.1.2.1.1.2.12",
        "2728",
        SnmpType.INTEGER );
list.add( variable );

// Add the SNMP variable list to the Map to use in the API call
map.put( SNMPPropertyKeys.SNMPVAR_LIST, list );

// Invoke the BACC API to do incremental update on the eMTA.
performOperation( DeviceOperation.INCREMENTAL_UPDATE  // device operation
        ,"1,6,00:00:00:00:00:99" // MAC Address
        , map   // Parameters for the operation
        );
```

ステップ 3　eMTA で 2 つ目の電話回線を使用できるようになります。eMTA は、ステップ 1 でサービス クラスが変更された後（リセット後）から、継続的に同じサービスを受けられるようになります。

動的設定ファイルを使用した DOCSIS モデムの事前プロビジョニング

新しい顧客が、サービス プロバイダーに連絡して、DOCSIS モデムと、モデムの背後に接続された 2 つの CPE を対象とする高速な *Gold* データ サービスを注文します。

求める結果

次のワークフローを使用して、DOCSIS テンプレートを使用するサービス クラスで DOCSIS モデムを事前プロビジョニングします。テンプレートから生成される動的設定ファイルは、モデムがオンラインになるときに使用されます。

ステップ 1　サービス プロバイダーが、課金システムで使用する加入者のユーザ名とパスワードを選択します。

ステップ 2　サービス プロバイダーが、Gold サービス クラスと適切な DHCP 基準を選択し、ケーブル モデムを BACC に追加します。

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


Map properties;

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

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

// No CPE DHCP Criteria is specified.
// The CPEs behind the modem will use the default provisioned
// promiscuous CPE DHCP criteria specified in the system defaults.

// This custom property corresponds to a macro variable in the
// DOCSIS template for “gold” class of service indicating the
// maximum number of CPEs allowed behind this modem. We set it
// to two CPEs from this customer.

properties.put(“docsis-max-cpes”, "2");

// To add a DOCSIS modem:

add(
        DeviceType.DOCSIS,
                // deviceType: DOCSIS
        “1,6,01:02:03:04:05:06”,
                // macAddress: scanned from the label
        null,
                // hostName: not used in this example
        null,
                // domainName: not used in this example
        "0123-45-6789",
                // ownerID: here, account number from billing system
        "gold",
                // classOfService:
        "provisionedCM",
                        // DHCP Criteria: Network Registrar uses this to
                // select a modem lease granting provisioned IP address
        properties
                // properties:
        );
```

ステップ 3　ケーブル モデムが顧客に出荷されます。

ステップ 4　顧客がケーブル モデムをオンラインにし、モデムの背後にコンピュータを接続します。

オプティミスティック ロッキング

サービス プロバイダー アプリケーションのインスタンスが、同じアプリケーションの別のインスタンスによって行われた変更を上書きしないことを保証します。

求める結果

次のワークフローを使用して、BACC API が提供するオプティミスティック ロッキング機能のデモを行います。

（注）　オブジェクトのロッキングは、マルチユーザ　システムで変更の完全性を維持するために実行されます。その結果、ユーザの変更が、別のユーザによって不用意に上書きされなくなります。オプティミスティック ロッキングを使用してプログラム記述する場合、コミットしようとするオブジェクトの少なくとも 1 つが、処理の開始後に別のユーザによって変更されていたときは、コミットが失敗する可能性があります。

ステップ 1　サービス担当者が、サービス プロバイダーのユーザ インターフェイスで検索オプションを選択し、ケーブル モデムの MAC アドレスを入力します。

ステップ 2　BACC が、組み込みデータベースにクエリーし、デバイスの詳細を取得します。次に、その情報が MSO ユーザ インターフェイスに表示されます。

```
// MSO admin UI calls the provisioning API to query the details
// for the requested device.
Map deviceDetails =
        getDetails(
                "1,6,00:11:22:33:44:55",
                        // macORFqdn: unique identifier for the device
                true
                        // needLeaseInfo: yes we need it
                };
// 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);
// service provider admin UI displays this information.

// Let's save the OID_REVISION_NUMBER property so that we can se in
// step 3.
String oidRevisionNumber = (String)deviceDetails.get(OID_REVISION_NUMBER);
```

ステップ 3　サービス担当者が、ユーザ インターフェイスを使用して、モデムのサービス クラスと DHCP 基準を変更します。その結果、BACC API が起動します。

```
// We need a reference to Batch instance so that ensureConsistency()
// method can be invoked on it.
Batch batch = conn.newBatch() ;

List oidList = new ArrayList();
// Add the oid-rev number saved from step 2 to the list
oidList.add(oidRevisionNumber);

// Sends a list of OID revision numbers to validate before processing the
// batch.  This ensures that the objects specified have not been modified
// since they were last retrieved.
batch.ensureConsistency(oidList);
       batch.changeClassOfService (
              "1,6,00:11:22:33:44:55",
              // macORFqdn: unique identifier for the device.
       "gold"
              // newCOSName : Class of service name.
              )
batch.changeDHCPCriteria (
       "1,6,00:11:22:33:44:55",
              // macORFqdn: unique identifier for the device.
       "specialDHCPCriteria"
              // newDHCPCriteria : New DHCP Criteria.
              )
// This batch fails with BatchStatusCodes.BATCH_NOT_CONSISTENT,
// in case if the device is updated by another client in the mean time.
// If there is a conflict occurs, then the service provider client
// is responsible for resolving the conflict by querying the database
// again and then applying changes appropriately.
```

ステップ 4　これで、ユーザが Gold サービス クラスと適切な DHCP 基準を受けられるようになります。

加入者の帯域幅の一時的なスロットリング

MSO が、加入者に許可するダウンロードのデータ量を 1 か月あたり 10 MB に制限するサービスを提供します。その限度に達すると、加入者のダウンストリーム帯域幅が 1 MB から 56 KB に制限されます。月が替わると 1 MB に戻ります。

（注） ピアツーピア ユーザや Web サイトを運営するユーザはアップロード帯域幅の使用率が非常に高くなる傾向があるため、アップストリーム帯域幅の変更も考慮することが必要になる場合があります。

求める結果

次のワークフローを使用して、加入者の帯域幅をその契約条件に従って増減させます。

ステップ 1　MSO が、*NetFlow* などのレート追跡システムを使用します。このシステムは、MAC アドレスに基づいて各顧客の使用率を追跡します。最初に、顧客は、1 MB のダウンストリームを持つ *Gold* サービス クラスのレベルでプロビジョニングされます。

ステップ 2　レート追跡ソフトウェアが、加入者が 10 MB の限度に達したことを特定すると、OSS に通知します。次に、OSS が BACC API を呼び出して、加入者のサービス クラスを *Gold* から *Gold-throttled* に変更します。

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

// AUTOMATIC is the activation mode because we are
// attempting  to reset the modem so that it
// receives low bandwidth service.
// NO_CONFIRMATION is the confirmation mode
// because we do not want the batch to fail if we cannot
// reset the modem. If the modem is off, when it will
// be disabled when it is turned back on.

// Let's change the COS of the device so that it restricts
// bandwidth usage of the modem.
        changeClassOfService(
                Device.MAC_ADDRESS,
                        // macAddress: unique identifier for this modem
                "Gold-throttled");
                        // newClassOfService: restricts bandwidth usage to 56k
```

ステップ 3　支払い請求周期の期末になると、OSS が BACC API を呼び出して、加入者のサービス クラスを *Gold* に戻します。

CableHome WAN-Man の事前プロビジョニング

新しい顧客が、サービス プロバイダーに連絡してホーム ネットワーキング サービスを注文します。顧客には、プロビジョニングされた CableHome デバイスが出荷されます。

求める結果

次のワークフローを使用して、CableHome デバイスを事前プロビジョニングし、ケーブル モデムとその WAN-Man コンポーネントがオンラインになったときに適切なレベルのサービスを受けられるようにします。

ステップ 1　サービス プロバイダーが、課金システムで使用する加入者のユーザ名とパスワードを選択します。

ステップ 2　サービス プロバイダーが、モデム コンポーネントに適切なサービス クラスと DHCP 基準を選択し、コンポーネントを BACC に追加します。

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

// Let's provision the modem and the WAN-Man component in the same
// batch.
// To add a DOCSIS modem:

add(
        DeviceType.DOCSIS,
                // deviceType: DOCSIS
        "1,6,01:02:03:04:05:06",
                // macAddress: scanned from the label
        null,
                // hostName: not used in this example
        null,
                // domainName: not used in this example
        "0123-45-6789",
                // ownerID: here, account number from billing system
        "Silver",
                // classOfService
        "provisionedCM",
                // DHCP Criteria: Network Registrar uses this to
                // select a modem lease granting provisioned IP address
        null
                // properties: not used
);
```

ステップ 3　サービス プロバイダーが、WAN-Man コンポーネントに適切なサービス クラスと DHCP 基準を選択し、コンポーネントを BACC に追加します。

```
// Continuation of the batch in Step2
        // To add the WAN-Man component:
add(
        DeviceType.CABLEHOME_WAN_MAN,
                // deviceType: CABLEHOME_WAN_MAN
        “1,6,01:02:03:04:05:07”,
                // macAddress: scanned from the label
        null,
                // hostName: not used in this example.
        null,
                // domainName: not used in this example.
        "0123-45-6789",
                // ownerID: here, account number from billing system
        "silverWanMan",
                // ClassOfService
        "provisionedWanMan",
                // DHCP Criteria: Network Registrar uses this to
                // select an MTA lease granting provisioned IP address
        null
                // properties: not used
        );
```

ステップ 4　CableHome デバイスが顧客に出荷されます。

ステップ 5　顧客が CableHome デバイスをオンラインにします。

ファイアウォール設定を持つ CableHome

顧客が、サービス プロバイダーに連絡して、ファイアウォール機能がイネーブルになったホーム ネットワーキング サービスを注文します。顧客は、プロビジョニングされた CableHome デバイスを受け取ります。

求める結果

次のワークフローを使用して、CableHome デバイスを事前プロビジョニングし、ケーブル モデムとその WAN-Man コンポーネントがオンラインになったときに適切なレベルのサービスを受けられるようにします。

ステップ 1　サービス プロバイダーが、課金システムで使用する加入者のユーザ名とパスワードを選択します。

ステップ 2　サービス プロバイダーが、ケーブル モデム コンポーネントに適切なサービス クラスと DHCP 基準を選択し、コンポーネントを BACC に追加します。

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

// Let's provision the modem and the WAN-Man component in the same
// batch.
// To add a DOCSIS modem:

add(
        DeviceType.DOCSIS,
                // deviceType: DOCSIS
        "1,6,01:02:03:04:05:06",
                // macAddress: scanned from the label
        null,
                // hostName: not used in this example
        null,
                // domainName: not used in this example
        "0123-45-6789",
                // ownerID: here, account number from billing system
        "Silver",
                // classOfService
        "provisionedCM",
                        // DHCP Criteria: Network Registrar uses this to
                // select a modem lease granting provisioned IP address
        null
                // properties: not used
        );
```

ステップ 3　サービス プロバイダーが、WAN-Man コンポーネントに適切なサービス クラスと DHCP 基準を選択し、コンポーネントを BACC に追加します。

```
// Continuation of the batch in Step2
// To add the WAN-Man component:

// Create a Map to contain WanMan's properties

Map properties;

// The fire wall configuration for the Wan Man component is specified
// using the CableHomeKeys.CABLEHOME_WAN_MAN_FIREWALL_FILE property.
// This use case assumes that the firewall configuration file named
// "firewall_file.cfg" is already present in the RDU database and the
// firewall configuration is enabled in the Wan Man configuration file
// specified with the corresponding class of service.

properties.put(CableHomeKeys.CABLEHOME_WAN_MAN_FIREWALL_FILE,
"firewall_file.cfg");

add(
        DeviceType.CABLEHOME_WAN_MAN,
                // deviceType: CABLEHOME_WAN_MAN
        "1,6,01:02:03:04:05:07",
                // macAddress: scanned from the label
        null,
                // hostName: not used in this example.
        null,
                // domainName: not used in this example.
        "0123-45-6789",
                // ownerID: here, account number from billing system
        "silverWanMan",
                // ClassOfService
        "provisionedWanMan",
                // DHCP Criteria: Network Registrar uses this to
                // select an MTA lease granting provisioned IP address
        properties
                // properties: contains the firewall config file
        );
```

ステップ 4　CableHome デバイスが顧客に出荷されます。

ステップ 5　顧客が CableHome デバイスをオンラインにします。次に、ケーブル モデムと WAN-Man コンポーネントが、プロビジョニングされた IP アドレスと正しい設定ファイルを取得します。

CableHome WAN-Man のデバイス機能の取得

サービス プロバイダーが管理者に対し、CableHome WAN-Man デバイスの機能の情報を表示することを許可します。

求める結果

サービス プロバイダーの管理アプリケーションが、特定の CableHome WAN-Man コンポーネントに関する既知の詳細をすべて表示します。この詳細には、MAC アドレス、リース情報、プロビジョニング ステータス、およびデバイス機能の情報などがあります。

ステップ 1　管理者が、サービス プロバイダーのユーザ インターフェイスに、クエリーする WAN-Man の MAC アドレスを入力します。

ステップ 2　BACC が、入力された MAC アドレスを使用して特定したデバイスの詳細を、組み込みデータベースにクエリーします。

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

// MSO admin UI calls the provisioning API to query the details
// for the requested device.

Map deviceDetails =
        getDetails(
                1,6,00:11:22:33:44:55","
                        // macORFqdn: unique identifier for the device
                true
                        // needLeaseInfo: yes we need it
                };
```

ステップ 3　サービス プロバイダーのアプリケーションが、デバイス データの詳細に関するページを表示します。このページには、要求されたデバイスに関する既知の情報をすべて表示できます。デバイスがサービス プロバイダーのネットワークに接続された場合、このデータにはリース情報（IP アドレスやリレー エージェントの ID など）が含まれます。このデータは、デバイスがプロビジョニングされているかどうかを示します。プロビジョニングされている場合、データにはデバイス タイプとデバイス機能の情報も含まれます。

```
// extract device details information from the map
String deviceType      = (String)   deviceDetails.get(DeviceDetailsKeys.DEVICE_TYPE);
String macAddress      = (String)   deviceDetails.get(DeviceDetailsKeys.MAC_ADDRESS);
String ipAddress       = (String)   deviceDetails.get(DeviceDetailsKeys.IP_ADDRESS);
String relayAgentID    = (String)   deviceDetails.get(
                                          DeviceDetailsKeys.RELAY_AGENT_ID);
Boolean isProvisioned  = (Boolean)  deviceDetails.get(
                                          DeviceDetailsKeys.IS_PROVISIONED);
String formation       = (String)    deviceDetails.get(
                                          IPDeviceCapabilities.FORMATION);
String deviceList      = (String)    deviceDetails.get(
                                          IPDeviceCapabilities.DEVICE_LIST);
String serNum          = (String)   deviceDetails.get(
                                          IPDeviceCapabilities.SERIAL_NUMBER);
String hwVer           = (String)   deviceDetails.get(
                                          IPDeviceCapabilities.HARDWARE_VERSION);
String swVer           = (String)   deviceDetails.get(
                                          IPDeviceCapabilities.SOFTWARE_VERSION);
String brVer           = (String)   deviceDetails.get(
                                          IPDeviceCapabilities.BOOT_ROM_VERSION);
String vendorOui       = (String)   deviceDetails.get(
                                          IPDeviceCapabilities.VENDOR_OUI);
String modelNum        = (String)   deviceDetails.get(
                                          IPDeviceCapabilities.MODEL_NUMBER);
String vendorNum       = (String)   deviceDetails.get(
                                          IPDeviceCapabilities. VENDOR_NAME)
String sysDesc         = (String)    deviceDetails.get(
                                          IPDeviceCapabilities.SYSTEM_DESCRIPTION);
String fwVer           = (String)   deviceDetails.get(
                                          IPDeviceCapabilities.FIRMWARE_VERSION);
String fwVer           = (String)   deviceDetails.get(
                                          IPDeviceCapabilities.FIREWALL_VERSION);
// The admin UI now formats and prints the detail data to a view page
```

CableHome WAN-Man のセルフプロビジョニング

加入者が、建物内にブラウザ アプリケーションを持つコンピュータを設置し、組み込み CableHome デバイスを購入しています。

求める結果

次のワークフローを使用して、プロビジョニングされていない新しい組み込み CableHome デバイスをオンラインにし、適切なレベルのサービスを受けられるようにします。さらに、組み込み CableHome デバイスに接続されたコンピュータから加入者がインターネットにアクセスできるようにします。

ステップ 1　加入者が、組み込み CableHome デバイスを購入し、建物内に設置します。

ステップ 2　加入者が、組み込み CableHome デバイスの電源をオンにします。BACC が、制限付きのアクセス権を組み込みケーブル モデムに与えます。つまり、アクセス権が 2 つの CPE（CableHome WAN-Man とコンピュータ）に制限されます。

（注）　この使用例は、プロビジョニングされていない DOCSIS モデムの背後に 2 つの CPE を接続できることを前提としています。特に設定しない限り、BACC では、プロビジョニングされていない DOCSIS モデムの背後に接続できるデバイスは 1 台のみです。この動作を変更するには、2 つの CPE をサポートする適切なサービス クラスを定義してから、そのサービス クラスを DOCSIS デバイスのデフォルト サービス クラスとして使用します。

ステップ 3　BACC が CableHome WAN-Man を設定します。この設定には、IP 接続や、デフォルトの CableHome ブート ファイルのダウンロードなどがあります。デフォルトの CableHome ブート ファイルは、CableHome デバイスをパススルー モードに設定します。まだ、CableHome デバイスはプロビジョニングされません。

ステップ 4　加入者が、コンピュータを CableHome デバイスに接続します。コンピュータが、プロビジョニングされていない（制限された）IP アドレスを取得します。加入者がコンピュータ上でブラウザ アプリケーションを起動します。スプーフィング DNS サーバにより、Web ブラウザがサービス プロバイダーの登録サーバ（OSS ユーザ インターフェイスやメディエータなど）にアクセスします。

ステップ 5　加入者がサービス プロバイダーのユーザ インターフェイスを使用して、ケーブル モデムの登録に必要な手順（サービス クラスの選択など）を完了します。また、加入者は CableHome のサービス クラスも選択します。

ステップ 6　サービス プロバイダーのユーザ インターフェイスが、加入者の情報（ケーブル モデムと CableHome に対して選択されたサービス クラスやコンピュータの IP アドレスなど）を BACC に渡します。次に、加入者が BACC に登録されます。

```
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). We also assume that “bostonProvGroup”
// is the provisioning group  used in that locality.

List provGroupList;
provGroupList = provGroupList.add(“bostonProvGroup”);
Map computerLease = getAllForIPAddress(

            "10.0.14.38",
        // ipAddress: restricted access computer lease

                provGroupList

        // provGroups: List containing provgroup)

// 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.getSingleLease().get(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");

properties.put(ModemKeys.CPE_DHCP_CRITERIA, "provisionedCPE");

add(
        DeviceType.DOCSIS,
                // deviceType: DOCSIS
        modemMACaddress,
                // macAddress: derived from computer lease
        null,
                // hostName: not used in this example
        null,
                // domainName: not used in this example
        "0123-45-6789",
                // ownerID: here, account number from billing system
        "Silver",
                // ClassOfService
```

```
        "provisionedCM",
                // DHCP Criteria: Network Registrar uses this to
                // select a modem lease granting provisioned IP address
        properties
                // properties:
);
```

ステップ 7　ユーザ インターフェイスにより、加入者はコンピュータをリブートするように求められます。

ステップ 8　プロビジョニング クライアントが performOperation(…) を呼び出してモデムをリブートし、プロビジョニングされたアクセス権をモデムに与えます。

```
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

performOperation(
        DeviceOperation.RESET,
                //deviceOperation: Reset operation
        modemMACaddress,
                // macAddress:Modem's MAC address
        null
                // properties : not used
        );
```

ステップ 9　コンピュータをリブートすると、コンピュータが CableHome デバイスの DHCP サーバから新しい IP アドレスを受信します。これで、ケーブル モデムと CableHome デバイスはいずれもプロビジョニングされます。この結果、加入者は CableHome デバイスのイーサネット ポートに多数のコンピュータを接続し、インターネットにアクセスすることができます。

（注）　WAN-Man コンポーネントに提供された設定ファイルによってボックスの WAN-Data コンポーネントがイネーブルになる場合、コンポーネントは無差別モードでプロビジョニングされます。無差別モードは DeviceType.CABLEHOME_WAN_DATA デバイス タイプのテクノロジー デフォルト レベルでイネーブルになることが前提となっています。

リース予約の使用例

この項では、リース予約機能の使用を主とした使用例について説明します。サービス クラス、DHCP 基準、およびライセンスの管理など、標準的な管理操作についてはここでは取り上げません。リース予約の使用例には、次のものがあります。

リース予約機能の影響を受ける API コール

次の API コールの実装が、リース予約機能をサポートします。

- IPDevice.add(DeviceType DeviceType, String macAddress, String hostName, String domainName, String ownerID, String cosName, String dhcpCriteria, Map Properties)
- IPDevice.changeProperties(String macORFqdn, Map newPropToAdd, List propToDelete)
- IPDevice.changeMACAddress(String macORFqdn, String newMAC)
- IPDevice.delete(String macORFqdn, Boolean deleteDevicesBehind)

サービス プロバイダーの IP アドレスを使用したデバイスのオンライン化

デバイスを BACC システムのデータベースに追加するときに、サービス プロバイダーが BACC プロパティ（IPDeviceKeys.IP_RESERVATION）を使用して、そのデバイスにデバイス固有の IP アドレスを設定します。

求める結果

サービス プロバイダーによってプロパティに設定されたとおりの IP アドレスで、デバイスをオンラインにします。付与された IP アドレスが、そのデバイスに対して予約されます（リース予約）。

ステップ 1　サービス プロバイダーが、BACC システムに新しいデバイスを追加します。サービス プロバイダーは、新しいデバイスに固有の IP アドレス 10.10.10.1 を設定します。

ステップ 2　サービス プロバイダーのユーザ インターフェイスが、デバイス情報（MAC アドレス、FQDN、およびサービス クラスなど）を BACC に渡します。BACC プロパティ（IPDeviceKeys.IP_RESERVATION）が使用され、そのデバイスに対して固有の IP アドレス（10.10.10.1）が予約されます。

```
Map properties;

        // Set the property IPDeviceKeys.IP_RESERVATION to a specific
        // IP address

        properties.put( IPDeviceKeys.IP_RESERVATION, "10.10.10.1");

        // To add a DOCSIS modem:

        get-new-batch(NO_ACTIVATION, NO_CONFIRMATION)

        add(
                 DeviceType.DOCSIS,
                        // deviceType: DOCSIS
                "1,6,01:02:03:04:05:06",
                        // macAddress: scanned from the label
                null,
                        // hostName: not used in this example
                null,
                        // domainName: not used in this example
                "0123-45-6789",
                        // ownerID: here, account number from billing system
                "gold",
                        // classOfService:
                "provisionedCM",
                         // DHCP Criteria: Network Registrar uses this to
                        // select a modem lease granting provisioned IP address
                properties
                        // properties:
                );
```

ステップ 3　新しいデバイスが BACC に登録されます。また、BACC/Network Registrar システムで、MAC アドレス「01:02:03:04:05:06」に対する 10.10.10.1 の予約も作成されます。

ステップ 4　デバイスをリブートすると、デバイスが、サービス プロバイダーによってプロパティに設定されたとおりの IP アドレス（10.10.10.1）を受信します。

API コマンドの処理中にエラーが発生した場合、または変更が RDU データベースにコミットできなかった場合は、必要に応じてロールバックが実行されます。MAC アドレス「01:02:03:04:05:06」に対応する 10.10.10.1 のリース予約がコマンド処理中に作成されていた場合は、コマンド実装によりその予約が BACC/Network Registrar から削除されます。

予約の削除と再作成

予約済みの IP アドレスでデバイスを BACC システムに登録した後で、サービス プロバイダーが、BACC プロパティ（IPDeviceKeys.IP_RESERVATION）を使用して、デバイスに別の IP アドレスを再割り当てします。

求める結果

デバイスが、サービス プロバイダーによってプロパティに設定されたとおりの IP アドレスでリブートされます。予約が削除され、再作成されます。そのデバイスに対して以前割り当てられた IP アドレスは予約解除され、新しい IP アドレスがそのデバイスに与えられ、予約されます（リース予約）。

ステップ 1　デバイスは、予約済みの IP アドレス 10.10.10.1 で BACC に登録されます。

ステップ 2　サービス プロバイダーのアプリケーションが、BACC で次の API コールを行い、予約済みの IP を 10.10.10.5 に変更します。

```
get-new-batch(AUTOMATIC, NO_CONFIRMATION);
// AUTOMATIC is the Activation mode because we are attempting
// to reset the device
// NO_CONFIRMATION is the Confirmation mode because we don't
// want the batch to fail if we can't reset the modem.

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

        Map properties;
                // Set the property IPDeviceKeys.IP_RESERVATION to a specific
                // IP address

                properties.put( IPDeviceKeys.IP_RESERVATION, "10.10.10.5");
                // To reassign a different IP to:


                    changeProperties(

                        "1,6,01:02:03:04:05:06",
                        // macAdd
                        properties, null


                     );
```

ステップ 3　MAC アドレス「01:02:03:04:05:06」に対する 10.10.10.1 の予約が、BACC/Network Registrar システムから削除されます。MAC アドレス「01:02:03:04:05:06」に対する 10.10.10.5 の予約が、BACC/Network Registrar システムで作成されます。

ステップ 4　PACE　ディスラプタによるデバイス中断の結果としてデバイスがリブートされた場合（アクティベーション モードには AUTOMATIC を使用）、デバイスは、サービス プロバイダーによってプロパティに設定されたとおりの IP アドレス（10.10.10.5）を受信します。

API コマンドの処理中にエラーが発生した場合、または変更が RDU データベースにコミットできなかった場合は、必要に応じてロールバックが実行されます。コマンドの実装により、デバイスに対して以前の動作設定へのロールバックが試行されます。このとき、MAC アドレス「01:02:03:04:05:06」に対する 10.10.10.1 の予約が削除されていた場合は、その予約が BACC/Network

Registrar システムに再度追加されます。一方、MAC アドレス「01:02:03:04:05:06」に対する 10.10.10.5 の予約がコマンド処理中に作成されていた場合は、その予約が BACC/Network Registrar システムから削除されます。

新しいデバイスに対する古いデバイスの IP アドレスの割り当て

予約済みの IP アドレスでデバイスを BACC システムに登録した後で、サービス プロバイダーが、故障したデバイスを新しい MAC アドレスを持つデバイスに交換します。

求める結果

新しいデバイスが、故障した古いデバイスに与えられたものと同じ IP アドレスでブートされます。デバイスの IP アドレスが変更された場合と同様に、予約が削除され、再作成されます。

ステップ 1　デバイスは、予約済みの IP アドレス 10.10.10.5 で BACC に登録されます。

ステップ 2　サービス プロバイダーが、BACC システムで、既存のデバイスの MAC アドレス（「01:02:03:04:05:06」）を新しいデバイスの MAC アドレス（「01:02:03:04:05:07」）に変更します。

```
get-new-batch(AUTOMATIC, NO_CONFIRMATION);
// NO_ ACTIVATION is the activation mode because we will
// not be able to reset as the new device has not booted
// on the network.
// NO_CONFIRMATION is the confirmation mode because we are
// not trying to reset the device

        // To change the MAC address of a device:

        changeMACAddress (
                "1,6,01:02:03:04:05:06",
                        // old macAddress: unique identifier for the old device
                 "1,6,01:02:03:04:05:07"
                        //// new macAddress: unique identifier for the new device
                 );
```

ステップ 3　MAC アドレス「01:02:03:04:05:06」に対する 10.10.10.5 の予約が、BACC/Network Registrar システムから削除されます。MAC アドレス「01:02:03:04:05:07」に対する 10.10.10.5 の予約が、BACC/Network Registrar システムで作成されます。

ステップ 4　MAC アドレス「01:02:03:04:05:07」を持つデバイスの電源がオンになると、デバイスは、故障した古いデバイスに与えられたものと同じ IP アドレス（10.10.10.5）を受信します。

API コマンドの処理中にエラーが発生した場合、または変更が RDU データベースにコミットできなかった場合は、必要に応じてロールバックが実行されます。コマンドの実装により、デバイスに対して以前の動作設定へのロールバックが試行されます。このとき、MAC アドレス「01:02:03:04:05:06」に対する 10.10.10.5 の予約が削除されていた場合は、その予約が BACC/Network Registrar システムに再度追加されます。一方、MAC アドレス「01:02:03:04:05:07」に対する 10.10.10.5 の予約がコマンド処理中に作成されていた場合は、その予約が BACC/Network Registrar システムから削除されます。

予約の削除と新しい IP アドレスの割り当て

予約済みの IP アドレスでデバイスを BACC システムに登録した後で、デバイスに対する予約済み IP の割り当てが不要になったため、サービス プロバイダーが、固有の IP アドレスの予約を削除します。

求める結果

予約がシステムから削除されます。次に、Network Registrar/BACC が、選択された DHCP 基準に基づいて、適切な IP プールで次に利用可能な IP アドレスを選択し、デバイスに割り当てます。

ステップ 1　デバイスは、予約済みの IP アドレス 10.10.10.5 で BACC に登録されます。

ステップ 2　サービス プロバイダーのアプリケーションが、BACC で次の API コールを行い、予約を削除します。

```
 get-new-batch(AUTOMATIC, NO_CONFIRMATION);
// AUTOMATIC is the Activation mode because we are attempting
// to reset the device
// NO_CONFIRMATION is the Confirmation mode because we don't
// want the batch to fail if we can't reset the modem.


// Add the property IPDeviceKeys.IP_RESERVATION the list to be removed

list.add( IPDeviceKeys.IP_RESERVATION);

// To reassign a different IP to:

changeProperties(
        "1,6,01:02:03:04:05:07",
        // macAdd
        null, list
);
```

ステップ 3　MAC アドレス「1,6,01:02:03:04:05:07」に対する 10.10.10.5 の予約がシステムから削除されます。

ステップ 4　デバイス中断の結果としてデバイスがリブートされた場合は（アクティベーション モードには AUTOMATIC を使用）、動的アドレス割り当てが発生します。Network Registrar/BACC が、デバイスに対して選択および設定された DHCP 基準に基づいて、適切な IP アドレス プールで次に利用可能な IP アドレスを選択し、デバイスに与えます。

API コマンドの処理中にエラーが発生した場合、または変更が RDU データベースにコミットできなかった場合は、必要に応じてロールバックが実行されます。コマンドの実装により、デバイスに対して以前の動作設定へのロールバックが試行されます。このとき、MAC アドレス「01:02:03:04:05:07」に対応する 10.10.10.5 の予約がコマンド処理中に削除されていた場合は、その予約が BACC/Network Registrar システムに再度追加されます。

同一の IP アドレスを使用したデバイスのリブート

BACC/Network Registrar によって与えられたプロビジョニング済みの IP アドレス（動的 IP 割り当てであって予約済み IP ではない）でデバイスを BACC システムに登録した後で、サービス プロバイダーが、BACC プロパティ（IPDeviceKeys.IP_RESERVATION）を使用して、デバイスに別の IP アドレスを再割り当てします。

求める結果

デバイスが、サービス プロバイダーによってプロパティ（IPDeviceKeys.IP_RESERVATION）に設定されたとおりの IP アドレスでリブートされます。その結果、付与された新しい IP アドレスが、そのデバイスに対して予約されます（リース予約）。

ステップ 1　動的 IP アドレスでデバイスが BACC に登録されます。この IP アドレスは、Network Registrar/BACC が、デバイスに対して選択された DHCP 基準に基づいて、適切な IP プールで選択し、デバイスに与えたものです。

ステップ 2　サービス プロバイダーのアプリケーションが、BACC で次の API コールを行い、IP アドレスを再割り当てします。

```
get-new-batch(AUTOMATIC, NO_CONFIRMATION);
// AUTOMATIC is the Activation mode because we are attempting
// to reset the device
// NO_CONFIRMATION is the Confirmation mode because we don't
// want the batch to fail if we can't reset the modem.

Map properties;

// Set the property IPDeviceKeys.IP_RESERVATION to a specific
// IP address

properties.put( IPDeviceKeys.IP_RESERVATION, "10.10.10.1");

// To reassign a different IP to:

changeProperties(

        "1,6,01:02:03:04:05:08",
                // macAdd
                properties, null


        ):
```

ステップ 3　MAC アドレス「01:02:03:04:05:08」に対する 10.10.10.1 の予約が BACC/Network Registrar で作成されます。

ステップ 4　デバイス中断の結果としてデバイスがリブートされた場合（アクティベーション モードには AUTOMATIC を使用）、デバイスは、サービス プロバイダーによってプロパティに設定されたとおりの IP アドレス（10.10.10.1）を受信します。

API コマンドの処理中にエラーが発生した場合、または変更が RDU データベースにコミットできなかった場合は、必要に応じてロールバックが実行されます。コマンドの実装により、デバイスに対して以前の動作設定へのロールバックが試行されます。MAC アドレス「01:02:03:04:05:08」に対する 10.10.10.1 の予約がコマンド処理中に追加されていた場合は、その予約が BACC/Network Registrar システムから削除されます。デバイスをリブートすると、BACC/Network Registrar が、選択された DHCP 基準に基づいて、適切な IP プールで次に利用可能な IP アドレスを選択し、デバイスに与えます。

BACC からのデバイスの削除

サービス プロバイダーが、予約済みの IP を持つ加入者のデバイスを BACC システムから削除します。

求める結果

加入者のデバイスを BACC システムから完全に削除します。付与された IP アドレスが、そのデバイスに対して予約解除されます。

ステップ 1　デバイスは、予約済みの IP アドレス 10.10.10.5 で BACC に登録されます。

ステップ 2　サービス プロバイダーが独自のユーザ インターフェイスを使用して、デバイスを BACC システムから削除します。サービス プロバイダーのユーザ インターフェイスは、BACC クライアントとして機能し、情報を BACC に渡します。BACC がデバイス情報を更新し、デバイスを削除します。

```
delete(
        "1,6,01:02:03:04:05:07",
        // macAdd
        // deleteDevicesBehind ress: unique identifier for this device
        true: deletes CPEs behind this modem.
):
```

ステップ 3　MAC アドレス「01:02:03:04:05:07」に対する 10.10.10.5 の予約も、BACC/Network Registrar システムから削除されます。

API コマンドの処理中にエラーが発生した場合、または変更が RDU データベースにコミットできなかった場合は、必要に応じてロールバックが実行されます。コマンドの実装により、デバイスに対して以前の動作設定へのロールバックが試行されます。MAC アドレス「01:02:03:04:05:07」に対する 10.10.10.5 の予約がコマンド処理中に削除されていた場合は、その予約が BACC/Network Registrar システムに再度追加されます。

BACC が CCM を使用するときに送信済みバッチが失敗する

BACC が、CCM を使用するように設定されています。OSS が、予約を追加または削除するために API 要求を送信しますが、CCM が利用不能になっています（接続ダウン、CCM の設定ミス、CCM ライセンスの問題など）。

求める結果

送信バッチが失敗した場合は、BACC/Network Registrar で予約の追加または削除が行われず、定義済みかつ周知のエラー コードが正常に返されます。

ステップ 1　OSS が、予約を追加または削除するために、P.C-58 の「リース予約機能の影響を受ける API コール」にリストされている API コールを含むバッチを作成し、送信します。

ステップ 2　RDU がバッチを受信し、それを処理します。RDU は処理中に、リース予約関連のタスクに関する外部 CCM コールを行います（P.C-58 の「リース予約機能の影響を受ける API コール」にリストされている API コマンドを使用）。

ステップ 3　CCM が利用不能であるため、エラーが発生します。API コールからの戻りステータス コードが、CommandStatusCodes .CMD_ERROR_CCM_UNREACHABLE に設定されます。

BACC が CCM を使用しないときに送信済みバッチが失敗する

BACC は、CCM を使用するように設定されて**いません**。OSS が、予約を追加または削除するために API 要求を送信します。

求める結果

送信済みバッチが失敗します。BACC/Network Registrar システムで予約の追加または削除が行われず、定義済みかつ周知のエラー コードが正常に返されます。

ステップ 1　OSS が、予約を追加または削除するために、P.C-58 の「リース予約機能の影響を受ける API コール」にリストされている API コールを含むバッチを作成し、送信します。

ステップ 2　RDU がバッチを受信し、それを処理します。RDU は処理中に、CCM がリース予約機能について設定されていないことを検出します（P.C-58 の「リース予約機能の影響を受ける API コール」にリストされている API コマンドを使用）。

ステップ 3　エラーが発生します。API コールからの戻りステータス コードが、CommandStatusCodes .CMD_ERROR_CCM_NOT_CONFIGURED に設定されます。