Application Programming Interface Data Types

Published: September 23, 2015

Introduction

This chapter describes the API data types used in the Cisco Service Control Engine (SCE) Subscriber application programming interface (API).

Subscriber ID

Most methods of the Cisco SCE Subscriber APIs require the subscriber ID as an input parameter. The Subscriber ID is a string representing a subscriber name or a content manager MAC address. This section lists the formatting rules for a subscriber ID.

The subscriber name is *case-sensitive*. It can contain up to 64 characters, which include all printable characters with an ASCII code from 32 through 126 (inclusive); except for 34 ("), 39 ('), and 96 (`).

The following examples present valid subscriber names:

```
String subID1="john";
String subID2="john@yahoo.com";
```

Network ID Mappings

A Network ID is a network identifier that the Cisco SCE device associates with a specific subscriber record. A typical example of a network ID mapping is an IP address. Currently, the Cisco SCE supports mappings to an IP address, IP range, private IP address over VLAN, private IP range over VLAN, and VLAN.

The network ID class represents various types of subscriber network identification.

The API supports the following subscriber mapping types:

- IP addresses or IP ranges
- Private IP addresses or private IP ranges over VLAN
- VLAN mappings
- (Starting from Cisco Service Control Subscriber Manager, Release 3.8.5) IPv6 address

When using subscriber operations that involve network ID, the caller is requested to provide a NetworkID parameter.

NetworkID class constructors are defined as follows:

```
public NetworkID(String mapping,short mappingType) throws Exception
public NetworkID(String[] mappings,short[] mappingTypes) throws Exception
```

Parameters of the NetworkID constructors are:

- java.lang.String mapping identifier or array of mapping identifiers
- Short mapping type or array of mapping types

When passing arrays, the mapping types array must contain either the same number of elements as the mappings array or a single element:

- Use NetworkID.TYPE_IP, NetworkID.TYPE_IPV6, or NetworkID.TYPE_VPN constants if the array contains more than one element.

Note The NetworkID.TYPE_VLAN and NetworkID.ALL_VLAN_MAPPINGS constants are deprecated. Use the NetworkID.TYPE_VPN and NetworkID.ALL_VPN_MAPPINGS constants instead.

- Use one of the following mappings when a single array element is used:
  - NetworkID.ALL_IP_MAPPINGS
  - NetworkID.ALL_IPV6_MAPPINGS
  - NetworkID.ALL_VLAN_MAPPINGS

Specifying IP Address Mapping

The string format of an IP address is the common decimal notation:

```
IP-Address=[0-255].[0-255].[0-255].[0-255]
```

Example

- 216.109.118.66

  The mapping type of an IP address is provided in the NetworkID class:

  ```
  com.scms.common.NetworkID.TYPE_IP:
  ```

  com.scms.common.NetworkID.ALL_IP_MAPPINGS specifies that all the entries in the mapping identifiers array are IP mappings.

Specifying IP Range Mapping

The string format of an IP range is an IP address in decimal notation and a decimal specifying the number of **one**s in a bit mask:

```
IP-Range=[0-255].[0-255].[0-255].[0-255]/[0-32]
```

Examples

- 10.1.1.10/32 is an IP range with a complete mask, that is, a regular IP address.
- 10.1.1.0/24 is an IP range with a 24-bit mask, that is, all the addresses in the range 10.1.1.0 through 10.1.1.255.

Note The mapping type of an IP Range is identical to the mapping type of the IP address.

Specifying Private IP Address or Private IP Range over VLAN Mapping

The string format of an IP address and an IP range are described in the "Specifying IP Address Mapping" section on page 4-3 and the "Specifying IP Range Mapping" section on page 4-3. When the network ID mapping uses an IP address or range over VLAN, the string format includes the VLAN number.

Examples

- 10.1.1.10@1 is an IP address over VLAN1.
- 10.1.1.0/24@2 is an IP range with a 24-bit mask, that is, all the addresses in the range 10.1.1.0 through 10.1.1.255 over VLAN2.

Note The mapping type of an IP address or IP range over VLAN is identical to the mapping type of the IP address.

Specifying VLAN Tag Mapping

The string format for VLAN tag mapping is a decimal number in the range from 2 to 2046.

The com.scms.common.NetworkID class provides the VLAN mapping type:

- Mapping type of an IP address is provided in the class Network ID:

  `com.scms.common.NetworkID.TYPE_VPN:`

- com.scms.common.NetworkID.ALL_VLAN_MAPPINGS specifies that all the entries in the mapping identifiers array are VLAN mappings.

Specifying IPv6 Address Mapping

The string format for IPv6 address mapping is a hexadecimal string that follows the RFC 2373 standard.

Example

2005:1:2:23::/64

The mapping type of an IP address is provided in the NetworkID class:

com.scms.common.NetworkID.TYPE_IPV6:

com.scms.common.NetworkID.ALL_IPV6_MAPPINGS specifies that all the entries in the array of mapping identifiers are IPv6 mappings.

Network ID Mapping Examples

The following is a list of sample network ID mappings:

- The following example constructs a NetworkID with a single IP address:

  ```
  NetworkID nid = new NetworkID("1.1.1.1",NetworkID.TYPE_IP)
  ```

- The following example constructs a NetworkID with a range of IP addresses:

  ```
  NetworkID nid = new NetworkID("1.1.1.1/24",NetworkID.TYPE_IP)
  ```

- The following example constructs a NetworkID with multiple IP addresses:

  ```
  NetworkID nid = new NetworkID(new String[]{"1.1.1.1","2.2.2.2","3.3.3.3"},
  NetworkID.ALL_IP_MAPPINGS)
  ```

- The following example constructs a NetworkID with a single IPv6 address:

  ```
  NetworkID nid = new NetworkID("4faf:f0ff:ddde:1234::/64", NetworkID.TYPE_IPV6 )
  ```

- The following example constructs a NetworkID with a range of IPv6 addresses:

  ```
  NetworkID nid = new NetworkID("4faf:f0ff:ddde:1234::/48", NetworkID.TYPE_IPV6)
  ```

- The following example constructs a NetworkID with multiple IPv6 addresses:

  ```
  NetworkID nid = new NetworkID(new
  String[]{"4faf:f0ff:ddde:1234::/64","2005:1:2:23::/64"}, NetworkID.ALL_IPV6_MAPPINGS)
  ```

- The following example constructs a NetworkID with a single VLAN address:

  ```
  NetworkID nid = new NetworkID("23",NetworkID.TYPE_VPN)
  ```

Cisco SCA BB Subscriber Policy Profile

The Policy Profile presents the subscriber policy information. A policy profile has two main parts. The policy package identifies a statically defined policy and a set of subscriber policy properties that can be dynamic. The package ID identifies the policy package. Most of the rules enforced on the subscriber traffic are derived from the package ID.

Subscriber policy property in Cisco SCA BB is a key-value pair that affects the way the Cisco SCE analyzes and reacts to network traffic generated by the subscriber.

The Cisco SCA BB supports the subscriber properties packageId, monitor, upVlinkId, and downVlinkId. For a description of the subscriber properties, see the *Cisco Service Control Application for Broadband User Guide*.

PolicyProfile Class

The API provides a PolicyProfile class to format subscriber policy profiles that the API requires to operate.

The following method constructs the PolicyProfile class based on the array of policy properties:

```
public PolicyProfile(String[] policy)
```

The encoding of each string within the array must be as follows:

```
property_name=property_value
```

The following method enables you to add a policy property to the profile:

```
public void addPolicyProperty(String policyProperty)
```

Note This method is not optimized for performance. To optimize performance, use the PolicyProfile constructor.

Example

```
PolicyProfile pp = new PolicyProfile(new String[]{"packageId=22", "monitor=1"})
```

Subscriber Quota

To provision subscriber quotas in Cisco SCA BB, you define quota buckets. Each subscriber has 16 buckets. You define each bucket for volume or sessions. When a subscriber uses a particular service, the volume or number of sessions consumed is subtracted from one of the buckets.

Use the SCA BB Console to create a general policy definition, which specifies which bucket to use for each service. Volume consumption is measured in ubits of Layer 3 kilobytes. The number of sessions determines the consumption of session buckets. For example, it is possible to specify that the browsing and e-mail services consume quota from Bucket 1, P2P service consumes quota from Bucket 2, and that all other services are not bound to any particular bucket.

A quota bucket has the following components:

- Bucket ID—Unique identifier of the bucket **(string)** as specified by the predefined policy. Valid values are numbers in the range 1 to 16
- Bucket value—Quota bucket value (long)

Quota operation dynamically modifies the quota buckets for a subscriber. There are two types of quota operations:

- ADD_QUOTA_OPERATION—Adds the new quota value to the current value of the bucket residing on the Cisco SCE platform
- SET_QUOTA_OPERATION—Replaces the value of the quota bucket residing on the Cisco SCE platform with the new value

Examples

Current subscriber quota values at the Cisco SCE are as shown in Figure 4-1.

Figure 4-1 Subscriber Quota - Current Values

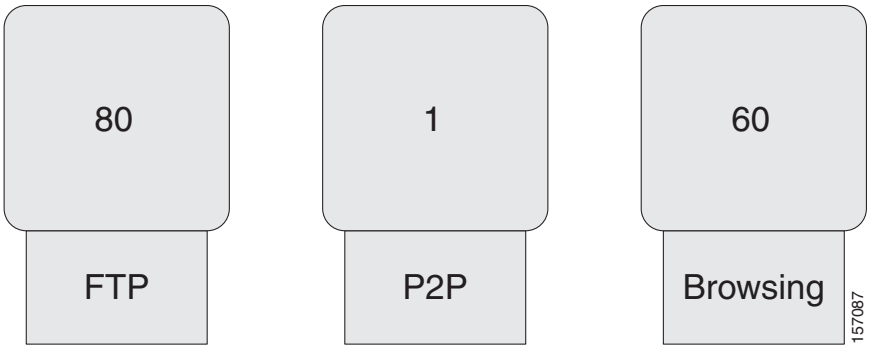
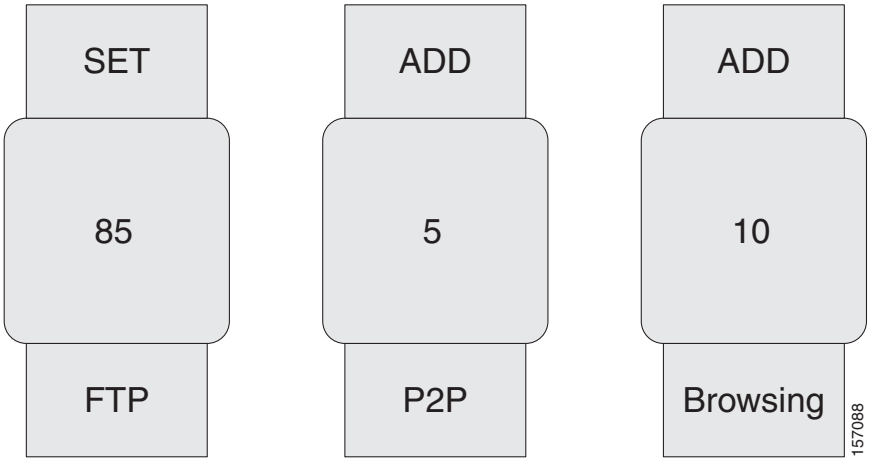


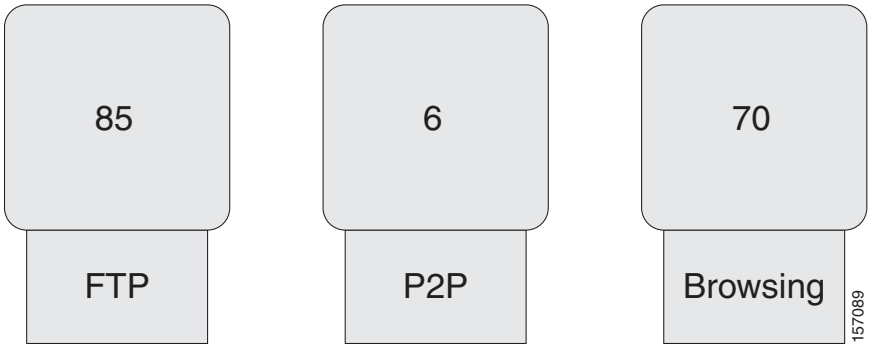
Figure 4-2 shows the application of actions to the existing quota.

Figure 4-2 Subscriber Quota - Actions to Apply



After performing the quota actions, the result is as shown in Figure 4-3.

Figure 4-3 Subscriber Quota - Results



For additional information about subscriber quota, see the *Cisco Service Control Application for Broadband User Guide*.

The following sections describe the classes the API provides for operations that include the subscriber quota management operations:

- SCAS_BB_Quota, page 4-7
- SCAS_BB_QuotaOperation, page 4-8

SCAS_BB_Quota

The SCAS_BB_Quota class implements the Quota interface, which the QuotaListenerEx interface uses in all callback functions. See the "QuotaListenerEx Interface Class" section on page 5-19.

The following method constructs SCAS_BB_Quota based on the array of IDs and values:

```
public SCAS_BB_Quota (String[] bucketIDs,
                      long[] bucketValues)
```

The following method constructs the SCAS_BB_Quota based on the array of IDs and values, the profile ID, the reason, and the time stamp:

```
public SCAS_BB_Quota (String[] bucketIDs,
                      long[] bucketValues,
                      int quotaProfileId,
                      int reason,
                      long timestamp)
```

The following method retrieves quota bucket IDs:

```
public String[] getBucketIDs()
```

The following method retrieves quota buckets values:

```
public long[] getBucketValues()
```

The quotaProfileId parameter identifies the quota profile, which is the package ID. The following method retrieves the quota profile ID:

```
public int getQuotaProfileId()
```

The reason parameter is relevant only for quota status events and has three possible values:

- 0—Configured time was reached, for example, every 2 minutes
- 1—A subscriber logout triggered a quota status event
- 2—A package change triggered a quota status event

The following method retrieves the reason:

```
public int getReason()
```

The timestamp parameter contains the time (in the Cisco SCE) when the event was generated. It is calculated as the number of seconds from January 1, 1970 00:00 GMT.

The following method retrieves the time stamp:

```
public long getTimestamp()
```

SCAS_BB_QuotaOperation

The SCAS_BB_QuotaOperation class implements the QuotaOperation interface, which is used for subscriber provisioning operations that include the subscribers quota, such as login operations (see the "Login Operation" section on page 5-30) and update quota operations (see the "quotaUpdate Operation" section on page 5-41).

The following method constructs the SCAS_BB_QuotaOperation based on the array of IDs, values, and actions:

```
public SCAS_BB_QuotaOperation (String[] IDs,
                               long[] values,
                               short[] actions)
```

The following method retrieves the quota bucket IDs:

```
public String[] getBucketIDs()
```

The following method retrieves the quota bucket values:

```
public long[] getBucketValues()
```

The following method retrieves the quota bucket actions:

```
public short[] getBucketActions()
```

Extended Attributes

The Extended Attributes class represents the data structure of extended attributes associated with a subscriber. Extended attributes can be Vendor Specific Attributes (VSA) or other attributes in RADIUS or other IP packets. The Cisco SCE Subscriber API sends extracted extended, attributes during the subscriber login and bulk login operations, both in Push and Pull Mode.

The following constructor is used for the esa object:

```
ExtendedAttributes(long[] esaCodeArr,
                                long[] vendorIdArr,
                                byte[] esaDataTypeArr,
                                java.lang.String[] esaDataArr)
```

Parameters

The extendend attributes parameters are as follows:

- esaCodeArr—Code or attribute ID of the extended attributes.
- vendorIdArr—Vendor ID of the extended attributes. If non-VSA, this value is zero (0).
- esaDataTypeArr—Data type of the extended attributes, which can be one of the following types: INTEGER, IP_ADDRESS, STRING, or OCTET_STRING.
- esaDataArr—Actual data for the specific extended attributes.

Note Only the Cisco SCE 8000 platform supports the Extended Attributes class.

Bulk Operations Data Types

Use bulk classes and operations when performing the same method for many subscribers, each of which has its own parameters. The API provides the bulk classes for result handling of bulk operations and for bulk indications from the Cisco SCE. The bulk classes are passed to the bulk methods such as loginBulk and logoutBulk.

Consider the following details when using the bulk operations:

- All bulk classes are inherited from the common BulkBase class.
- Because of the memory constraints of the Cisco SCE, the bulk size is limited to a maximum of 100 entries.

Bulk Iterator

The BulkBase class provides an iterator to view the data contained in the bulk.

The following is the syntax for the Bulk Iterator:

```
Iterator getIterator()
```

You can use this iterator for iterations over the bulks received from the Cisco SCE in various indications (for example, logoutBulkIndication and loginPullBulkResponseIndication) or for inspecting the data you provided to various operations when an operation fails.

The iterator provides the following methods for data retrieval:

```
public Object next()
public boolean hasNext()
```

The next() method returns a SubscriberData object.

The SubscriberData class retrieves the information of a single subscriber contained within the bulk.

SubscriberData

The SubscriberData class represents all the operations that can be performed on a specific subscriber. The SubscriberData class contains the following methods for retrieving information:

```
public String getSubscriberID()
public String getAnonymousID()
public String[] getMappings()
public short[] getTypes()
public boolean getAdditiveFlag()
```

The following sections describe various bulk data types that are available for different API operations.

Login_BULK Class

The Login_BULK class represents the bulk of subscribers and it includes all the data required for the loginBulk operation:

- Constructor, page 4-11
- addBulkEntry Method, page 4-11
- Examples, page 4-12

Constructor

To construct the Login_BULK class filled with the data, use the following constructor:

```
public Login_BULK(String[] subscriberIDs,
                  NetworkID[] networkIDs,
                  boolean[] additive,
                  PolicyProfile[] policy,
                  QuotaOperation[] quota)
```

To support extended attributes in the Login_BULK class, a new parameter was added to the constructor starting in Cisco SCE Subscriber, Release 3.6.5:

```
public Login_BULK(String[] subscriberIDs,
                  NetworkID[] networkIDs,
                  boolean[] additive,
                  PolicyProfile[] policy,
                  QuotaOperation[] quota,
                  ExtendedAttributes[] esa)
```

To construct an empty Login_BULK class, use the following method:

```
public Login_BULK()
```

Parameters

The Login_BULK class parameters are as follows:

- subscriberIDs—Unique ID of the subscriber. For a description of the subscriber ID format, see the "Subscriber ID" section on page 4-1.
- networkIDs—Network identifier of the subscriber. For more information, see the "Network ID Mappings" section on page 4-2.
- additive—If this parameter is set to TRUE, the specified networkID is added to the existing networkIDs of the subscriber. Otherwise, the specified networkID replaces the existing networkIDs.
- policy—Policy profile of the subscriber. For more information, see the "Cisco SCA BB Subscriber Policy Profile" section on page 4-5.
- quota—Quota of the subscriber. For more information, see the "Subscriber Quota" section on page 4-6.
- esa—Extended attributes associated with the subscriber and passed in the login operation.

addBulkEntry Method

To add entries to the bulk, use the following method:

```
public void addBulkEntry(String subscriberID,
                         NetworkID networkID,
                         boolean networkIdAdditive,
                         PolicyProfile policy,
                         QuotaOperation quota)
```

To support extended attributes in the addBulkEntry class, a new parameter was added to the constructor starting in Cisco SCE Subscriber, Release 3.6.5:

```
public void addBulkEntry(String subscriberID,
                         NetworkID networkID,
                         boolean networkIdAdditive,
                         PolicyProfile policy,
```

```
                                QuotaOperation quota,
                                ExtendedAttributes esa)
```

Parameters

The addBulkEntry method parameters are as follows:

- subscriberID—Unique ID of the subscriber. For a description of the subscriber ID format, see the "Subscriber ID" section on page 4-1.
- networkID—Network identifier of the subscriber. See the "Network ID Mappings" section on page 4-2.
- networkIdAdditive—If you set this parameter to TRUE, the supplied NetworkID is added to the existing networkIDs of the subscriber. Otherwise, the supplied networkID replaces the existing networkIDs.
- policy—Policy profile of the subscriber. See the "Cisco SCA BB Subscriber Policy Profile" section on page 4-5.
- quota—Quota of the subscriber. See the "Subscriber Quota" section on page 4-6.
- esa—Extended attributes associated with the subscriber and passed in the login operation.

Examples

Login_BULK Object Usage

```
// Prepare all data for the bulk construction
String[] names = new String[5];
NetworkID[] mappings = new NetworkID[5];
boolean[] additive = new boolean[5];
PolicyProfile[] policy = new PolicyProfile[5];

for (int i=0; i<5; i++)
{
    names[i]="sub_"+i;
    mappings[i] = new NetworkID("1.1.1."+i,NetworkID.TYPE_IP);
    additive[i] = true;
    policy[i] = new PolicyProfile(new String[]{"packageId="+(i+1)});
}

// construct the bulk object
Login_BULK bulk = new Login_BULK(names,mappings,additive,policy,null);
// Now it can be used in loginBulk operation
sceApi.loginBulk(bulk,null);
```

Manipulating Login_BULK

```
// Construct the empty bulk
Login_BULK bulk = new Login_BULK ();

// Fill the bulk using addBulkEntry method:
for (int i=0; i<20; i++)
{
    String name ="sub_"+i;
    NetworkID mappings = new NetworkID(i+1);
    boolean additive = true;
    PolicyProfile policy = new PolicyProfile(new String[]{"packageId="+(i+1)});
    QuotaOperation quota = new SCAS_BB_QuotaOperation(
           new String[]{"1","2","3"},
           new long[]{80,80,0}
           new short[]{SCAS_BB_QuotaOperation.ADD_QUOTA_OPERATION,
                       SCAS_BB_QuotaOperation.ADD_QUOTA_OPERATION,
                       SCAS_BB_QuotaOperation.SET_QUOTA_OPERATION});
    bulk.addBulkEntry(name,mappings,additive,policy,quota);
}
// Now it can be used in loginBulk operation
sceApi.loginBulk(bulk,null);
```

SubscriberID_BULK Class

The logoutBulkIndication callback function, which requires you to enter only subscriber IDs, uses the SubscriberID_BULK class. See the "logoutBulkIndication Callback Method" section on page 5-19.

Constructors

To construct SubscriberID_BULK with subscriber IDs data, use the following constructor:

```
public SubscriberID_BULK(String[] subscriberIDs)
```

To construct an empty SubscriberID_BULK, use the following method:

```
public SubscriberID_BULK()
```

Parameter

subscriberID—Unique ID of the subscriber. For a description of the subscriber ID format, see the "Subscriber ID" section on page 4-1.

addBulkEntry Method

To add entries to the SubscriberID bulk, use the following method:

```
addBulkEntry(String subscriberID)
```

Parameter

subscriberID—Unique ID of the subscriber. For a description of the subscriber ID format, see the "Subscriber ID" section on page 4-1.

NetworkAndSubscriberID_BULK Class

Use the NetworkAndSubscriberID_BULK class in bulk operations that require subscriber IDs and NetworkIDs:

- getSubscribersBulkResponse callback (see the "LoginPullListener Interface Class" section on page 5-15)
- logoutBulk operation (see the "logoutBulk Operation" section on page 5-36)
- networkIDUpdateBulk operation (see the "networkIdUpdateBulk Operation" section on page 5-39)

Constructors

To construct NetworkAndSubscriberID_BULK with the subscriberID and networkID data, use the following constructor:

```
public NetworkAndSubscriberID_BULK(String[] subscriberIDs,
                                   NetworkID[] networkIDs,
                                   boolean[] netIdAdditive)
```

To construct an empty NetworkAndSubscriberID_BULK, use the following method:

```
public NetworkAndSubscriberID_BULK()
```

Parameters

The NetworkAndSubscriberID_BULK class parameters are as follows:

- subscriberID—Unique ID of the subscriber. For a description of the subscriber ID format, see the "Subscriber ID" section on page 4-1.
- networkID—Network identifier of the subscriber. See the "Network ID Mappings" section on page 4-2.
- networkIDAdditive—If this parameter is set to TRUE, the supplied NetworkID is added to the existing networkIDs of the subscriber. Otherwise, the supplied networkID replaces the existing networkIDs.

addBulkEntry Method

To add entries to the bulk, use the following method:

```
addBulkEntry(String subscriberID,
             NetworkID networkID,
             boolean netIdAdditive)
```

Parameters

The addBulkEntry method parameters are as follows:

- subscriberID—Unique ID of the subscriber. For a description of the subscriber ID format, see the "Subscriber ID" section on page 4-1.
- networkID—Network identifier of the subscriber. See the "Network ID Mappings" section on page 4-2.
- networkIDAdditive—If this parameter is set to **TRUE**, the supplied NetworkID is added to the existing networkIDs of the subscriber. Otherwise, the supplied networkID replaces the existing networkIDs.

LoginPullResponse_BULK Class

This class represents a bulk of subscribers and includes all data required for the loginPullResponseBulk method.

To construct an empty LoginPullResponse_BULK table, use the following method:

```
public LoginPullResponse_BULK()
```

Constructors

To construct the LoginPullResponse_BULK table containing the relevant data, use the following constructor:

```
public LoginPullResponse_BULK(String[] anonymousIDs,
                              String[] subscriberIDs,
                              NetworkID[] networkIDs,
                              boolean[] additive,
                              PolicyProfile[] policy,
                              QuotaOperation[] quota)
```

To support extended attributes in the LoginPullResponse_BULK class, a new parameter was added to the constructor starting in Cisco SCE Subscriber, Release 3.6.5:

```
public LoginPullResponse_BULK(String[] anonymousIDs,
                              String[] subscriberIDs,
                              NetworkID[] networkIDs,
                              boolean[] additive,
                              PolicyProfile[] policy,
                              QuotaOperation[] quota,
                              ExtendedAttributes[] esa)
```

Parameters

The LoginPullResponse_BULK class parameters are as follows:

- anonymousIDs—Identifier of the anonymous subscriber. The Cisco SCE sends this parameter within the loginPullRequest/loginPullBulkRequest indication (see the "loginPullRequest Callback Method" section on page 5-16 and the "loginPullRequestBulk Callback Method" section on page 5-17). See the "Subscriber Integration Modes" section on page 2-3.
- subscriberIDs—Unique ID of the subscriber. For a description of the subscriber ID format, see the "Subscriber ID" section on page 4-1.
- networkIDs—Network identifier of the subscriber. See the "Network ID Mappings" section on page 4-2.
- additive—If this parameter is set to TRUE, the supplied NetworkID is added to the existing networkIDs of the subscriber. Otherwise, the supplied networkID replaces the existing networkIDs.
- policy—Policy profile of the subscriber. See the "Cisco SCA BB Subscriber Policy Profile" section on page 4-5.
- quota—Quota of the subscriber. See the "Subscriber Quota" section on page 4-6.
- esa—Extended attributes associated with the subscriber and passed in the login operation.

addBulkEntry Method

To add entries to the bulk, use the following method:

```
public addBulkEntry(String anonymousSubscriberID,
                    String subscriberID,
                    NetworkID networkID,
                    boolean networkIdAdditive,
                    PolicyProfile policy,
                    QuotaOperation quota)
```

To support extended attributes in the addBulkEntry class, a new parameter was added to the constructor starting in Cisco SCE Subscriber, Release 3.6.5:

```
public addBulkEntry(String anonymousSubscriberID,
                    String subscriberID,
                    NetworkID networkID,
                    boolean networkIdAdditive,
                    PolicyProfile policy,
                    QuotaOperation quota,
                    ExtendedAttributes esa)
```

Parameters

The addBulkEntry method parameters are as follows:

- anonymousSubscriberID—Identifier of the anonymous subscriber. The Cisco SCE sends this parameter within the loginPullRequest/loginPullBulkRequest indication (see the "loginPullRequest Callback Method" section on page 5-16 and the "loginPullRequestBulk Callback Method" section on page 5-17). See the "Subscriber Integration Modes" section on page 2-3.
- subscriberID—Unique ID of the subscriber. For a description of the subscriber ID format, see the "Subscriber ID" section on page 4-1.
- networkID—Network identifier of the subscriber. See the "Network ID Mappings" section on page 4-2.
- networkIDAdditive—If this parameter is set to TRUE, the supplied NetworkID is added to the existing networkIDs of the subscriber. Otherwise, the supplied networkID replaces the existing networkIDs.
- policy—Policy profile of the subscriber. See the "Cisco SCA BB Subscriber Policy Profile" section on page 4-5.
- quota—Quota of the subscriber. See the "Subscriber Quota" section on page 4-6.
- esa—Extended attributes associated with the subscriber and passed in the login operation.

PolicyProfile_BULK Class

The updatePolicyProfileBulk operation uses this class, which represents a bulk of subscriber IDs and subscriber policy profiles.

Constructors

To construct the PolicyProfile_BULK table containing the relevant data, use the following constructor:

```
public PolicyProfile_BULK(String[] subscriberIDs, PolicyProfile[] policy)
```

To construct an empty PolicyProfile_BULK table, use the following method:

```
public PolicyProfile_BULK()
```

Parameters

The PolicyProfile_BULK class parameters are as follows:

- subscriberID—Unique ID of the subscriber. For a description of the subscriber ID format, see the "Subscriber ID" section on page 4-1.
- policy—Policy profile of the subscriber. See the "Cisco SCA BB Subscriber Policy Profile" section on page 4-5.

addBulkEntry Method

To add entries to the bulk, use the following method:

```
public addBulkEntry(String subscriberID, PolicyProfile policy)
```

Parameters

The addBulkEntry Method parameters are as follows:

- subscriberID—The unique ID of the subscriber. See the "Subscriber ID" section on page 4-1 for the subscriber ID format description.
- policy—Policy profile of the subscriber. See the "Cisco SCA BB Subscriber Policy Profile" section on page 4-5.

Quota_BULK Class

The following operations use this class, which represents a bulk of subscriber IDs and subscriber quota buckets:

- getQuotaStatusBulk operation (only the bucket IDs are provided)
- quotaStatusBulkIndication callback method
- quotaDepletedBulkIndication callback method
- quotaBelowThresholdIndication callback method

Constructors

To construct the Quota_BULK table containing the relevant data, use the following constructor:

```
public Quota_BULK(String[] subscriberIDs, Quota[] subscribersQuota)
```

To construct an empty Quota_BULK table, use the following method:

```
public Quota_BULK()
```

Parameters

The Quota_BULK class parameters are as follows:

- subscriberID—Unique ID of the subscriber. For a description of the subscriber ID format, see the "Subscriber ID" section on page 4-1.
- quota—Quota of the subscriber. See the "Subscriber Quota" section on page 4-6.

addBulkEntry Method

To add entries to the bulk, use the following method:

```
public addBulkEntry(String subscriberID, Quota quota)
```

Parameters

The addBulkEntry method parameters are as follows:

- subscriberID—Unique ID of the subscriber. For a description of the subscriber ID format, see the "Subscriber ID" section on page 4-1.
- quota—Quota of the subscriber. See the "Subscriber Quota" section on page 4-6.

QuotaOperation_BULK Class

The QuotaUpdateBulk operation and the login operation use this class, which represents a bulk of subscriber IDs and subscriber quota operations.

Constructors

To construct the QuotaOperation_BULK table containing the relevant data, use the following constructor:

```
public QuotaOperation_BULK(String[] subscriberIDs,
                           QuotaOperation[]quotaOperations)
```

To construct an empty QuotaOperation_BULK table, use the following method:

```
public QuotaOperation_BULK()
```

Parameters

The QuotaOperation_BULK class parameters are as follows:

- subscriberID—Unique ID of the subscriber. For a description of the subscriber ID format, see the "Subscriber ID" section on page 4-1.
- quotaOperation—Quota operation to perform on the quota of the subscriber. See the "Subscriber Quota" section on page 4-6.

addBulkEntry Method

To add entries to the bulk, use the following method:

```
addBulkEntry(String subscriberID, QuotaOperation quotaOperation)
```

Parameters

The addBulkEntry method parameters are as follows:

- subscriberID—Unique ID of the subscriber. For a description of the subscriber ID format, see the "Subscriber ID" section on page 4-1.
- quotaOperation—Quota operation that is to be performed on the quota of a subscriber. See the "Subscriber Quota" section on page 4-6.

Virtual Link Data Types

This section describes the various data types related to the virtual link API.

VLink Class

The VLink class represents the data structure of the virtual link associated with a subscriber. When a virtual link is created, a channel too gets created by default for this virtual link.

The following method adds channels to a virtual link by passing a channel object:

```
addChannel(Channel channel)
```

The following method adds channels to a virtual link by passing an array of channel objects:

```
addChannel(Channel[] channels)
```

Constructors

To construct a VLink class based on vlinkId, vlinkName, direction, agcOffset, vlinkPIR, vlinkCIR, and vlinkA1Value, use the following constructor:

```
public VLink(int vlinkId, String vlinkName, int direction, int agcOffset, int vlinkPIR,
int vlinkCIR, int vlinkALValue)
```

To construct a VLink class based on vlinkId, vlinkName, direction, agcOffset, vlinkPIR, and vlinkCIR, use the following constructor:

```
public VLink(int vlinkId, String vlinkName, int direction, int agcOffset, int vlinkPIR,
int vlinkCIR)
```

Parameters

The VLink class parameters are as follows:

- vlinkId—Describes the unique ID of a virtual link.
- vlinkName—Describes the name of a virtual link.
- direction—Describes the upstream and downstream directions that are represented by the integer constants in Cisco SCE. The value is 0 for upstream direction and 1 for downstream direction.
- agcOffset—Describes the offset value of Aggregate Global Control (AGC) that is configured.
- vlinkPIR—Describes the peak information rate (PIR) of a virtual link.
- vlinkCIR—Describes the committed information rate (CIR) of a virtual link.
- vlinkA1Value—Describes the Assurance Level (AL) value of a virtual link.

Channel Class

A channel class represents the data structure of the channel associated with a subscriber. It also contains the direction and vlinkId parameters of the virtual link to which the corresponding channel is associated.

Constructors

To construct a channel class based on a channelId, channelName, channelPIR, channelCIR, and channel AlValue, use the following constructor:

```
public Channel(int channelId, String channelName, int channelPIR, int channelCIR, int
channelAlValue)
```

To construct a channel class based on channelId, channelName, channelPIR,, and channelCIR use the following constructor:

```
public Channel(int channelId, String channelName, int channelPIR, int channelCIR)
```

Parameters

The channel class parameters are as follows:

- channelId—Describes the unique ID of a channel class.
- channelName—Describes the name of a channel class.
- channelPIR—Describes the PIR of a channel class.
- channelCIR—Describes the CIR of a channel class.
- channelAlValue—Describes the Assurance Level value of a channel class.

VLinkDisabledException Class

VLinkDisabledException is a user-defined exception in com.scms.api.sce.prpc. This exception is displayed when a user tries to use the virtual link APIs with the virtual link being disabled in the Cisco SCE.

VLinkModeException Class

VLinkModeException is a user-defined exception that is displayed when virtual link APIs are not used in the user mode. To enable the user mode, use the following method:

```
enableVlinkMode()
```

Note We recommend that in order to enable the user mode, you use the **enableVlinkMode()** command before the Virtual Link API gets connected to the Cisco SCE platform.