Concepts and Terms

Published: August 30, 2014,

Introduction

This chapter describes the various terms and concepts related to working with the Cisco Service Control Engine (SCE) Subscriber application programming interface (API).

It consists of the following sections:

Subscriber Characteristics

A subscriber is one of the fundamental entities in the Service Control Application for Broadband (SCA BB) solution.

The configuration of services in the SCA BB solution is based on enforcing, accounting, and monitoring the individual subscriber.

For more information about formatting and configuring subscriber characteristics, see Chapter 4, "Application Programming Interface Data Types." The following sections describe the characteristics of the subscriber in the SCA BB:

- Subscriber ID, page 2-2
- Anonymous Subscriber ID, page 2-2
- Network ID, page 2-2
- Policy Profile, page 2-2
- Quota, page 2-2
- Extended Attributes, page 2-3

Subscriber ID

A Subscriber ID is a unique identifier, such as a user name, International Mobile Subscriber Identity (IMSI), or other code that uniquely identifies a subscriber.

Anonymous Subscriber ID

For Pull Mode integration, the Cisco SCE assigns each unknown subscriber IP address a temporary, anonymous subscriber ID, until it receives the real subscriber ID from the policy server.

For more information on Pull Mode integration, see the "Subscriber Integration Modes" section on page 2-3.

Network ID

The Cisco SCE correlates a certain traffic flow to a subscriber by mapping a network identifier such as an IP address, IP range, IPv6 address, or VLAN, to the subscriber entity.

Policy Profile

A Policy Profile includes a set of parameters that the Cisco SCA BB solution uses to define the policy that is enforced on the subscriber.

Quota

A quota includes the quota-bucket values of the service quota or quotas available for the subscriber.

Extended Attributes

Extended attributes represent the data structure of the extended attributes associated with the subscriber.

Subscriber Integration Modes

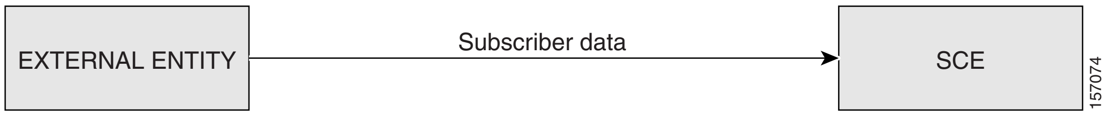
The Cisco SCE platform supports the following two modes of dynamic subscriber integration:

- Push Mode, page 2-3
- Pull Mode, page 2-3

Push Mode

In Push Mode integration, an external server introduces (pushes) the subscribers to the Cisco SCE platform as shown in Figure 2-1. This push occurs whenever a new subscriber logs in to the network or the external server presumes to know all the subscribers and introduces them to the Cisco SCE box when they connect.

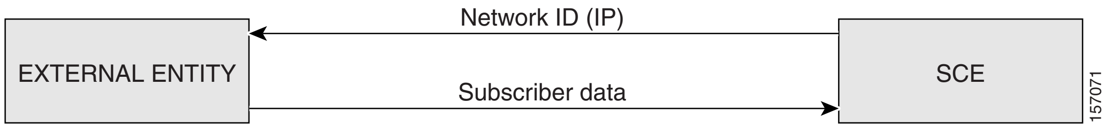
Figure 2-1 Push Mode Schematic



Pull Mode

In Pull Mode integration, the Cisco SCE platform requests subscriber data from the external entity when it encounters traffic of an unknown (anonymous) subscriber, as shown in Figure 2-2. The external entity retrieves the required subscriber information and sends it back to the Cisco SCE platform.

Figure 2-2 Pull Mode Schematic



Nonblocking Mode

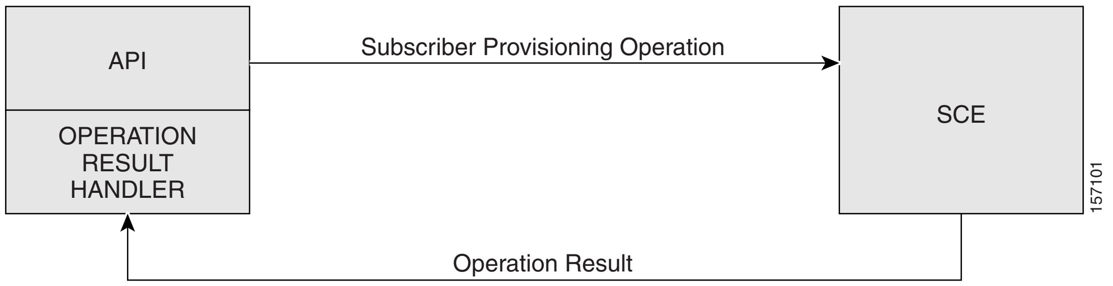
The Cisco SCE Subscriber API is implemented by using a nonblocking mode. Nonblocking methods return immediately, even before the completion of a subscriber provisioning operation. The nonblocking mode method is best suited when the operation is lengthy and involves I/O. Performing the operation in a separate thread allows the caller to continue doing other tasks and it improves overall system performance.

The operation results are either returned to an observer object (listener) or not returned at all.

The API supports retrieval of operation results through an operation result handler described in the "Result Handling" section on page 5-27.

Figure 2-3 illustrates the nonblocking mode method during a subscriber provisioning operation.

Figure 2-3 Nonblocking Mode

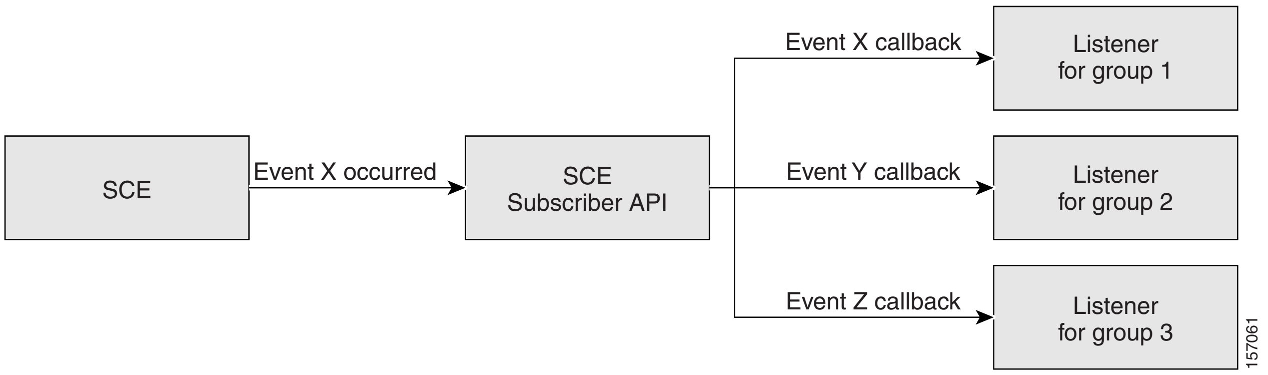


Error logging enables you to monitor the results of an operation or to inspect the parameters that the operation uses.

Indications Listeners

The API enables you to receive an indication when certain events occur on the Cisco SCE platform as shown in Figure 2-4. The API dispatches the indications received from the Cisco SCE to the interested entities, called listeners, by activating the relevant listeners callback methods. The indications are separated into several logical groups. You can define only one listener for each group of indications.

Figure 2-4 Indication Listeners



To receive certain indications, register a listener to the API that implements the required callback functions. After the listener is registered, the API can dispatch the required indications to the listener. The Cisco SCE Subscriber API provides three types of indications when separate listeners are registered to the following types of indications:

- Login-pull indications
- Logout indications
- Quota indications

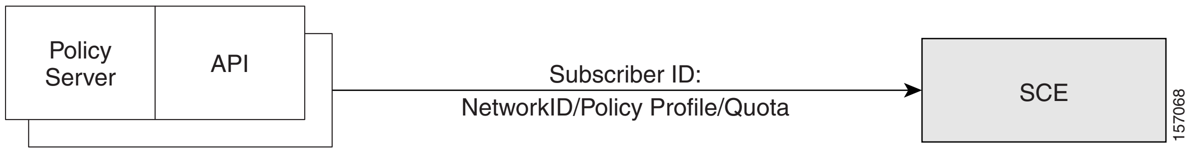
For more information about listener indications, see Chapter 3, "Application Programming Interface Events."

Supported Topologies

The Cisco SCE Subscriber API supports the following topologies:

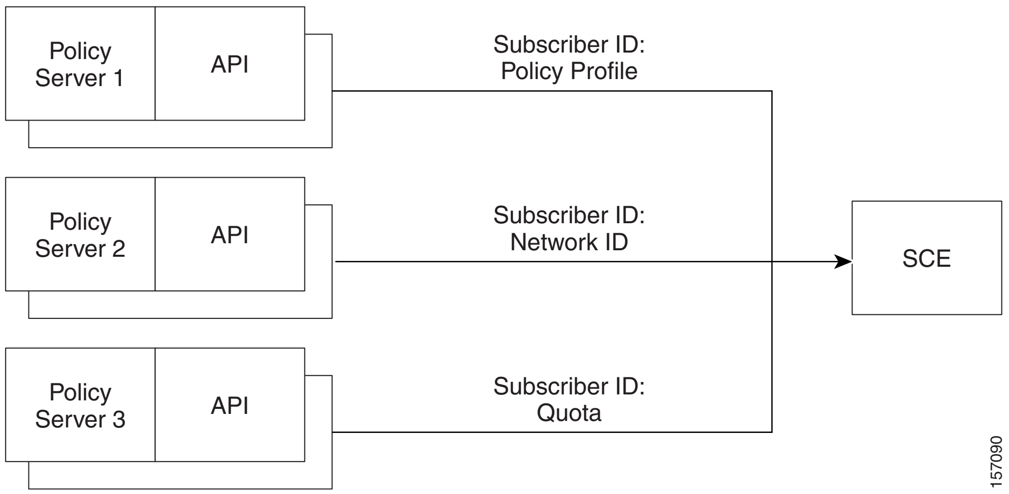
- One policy server (or one two-node cluster) that is responsible for all aspects of the subscriber provisioning process as shown in Figure 2-5.

Figure 2-5 Supported Topologies - One Policy Server



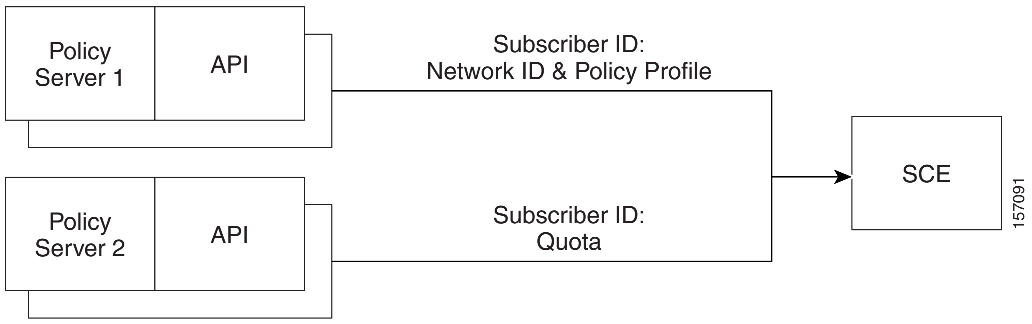
- Three policy servers (or three two-node clusters)—Every server is responsible for a different aspect of the subscriber provisioning process as shown in Figure 2-6.

Figure 2-6 Supported Topologies - Three Policy Servers



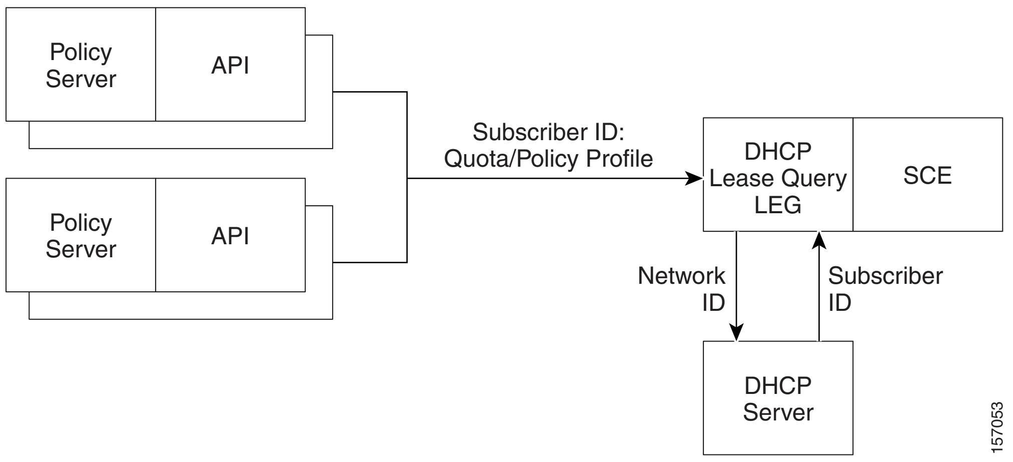
- Two policy servers (or two two-node clusters) when one of the servers is responsible for two aspects of the subscriber provisioning and the other server is responsible for only one aspect (any combination is allowed) as shown in Figure 2-7.

Figure 2-7 Supported Topologies - Two Policy Servers



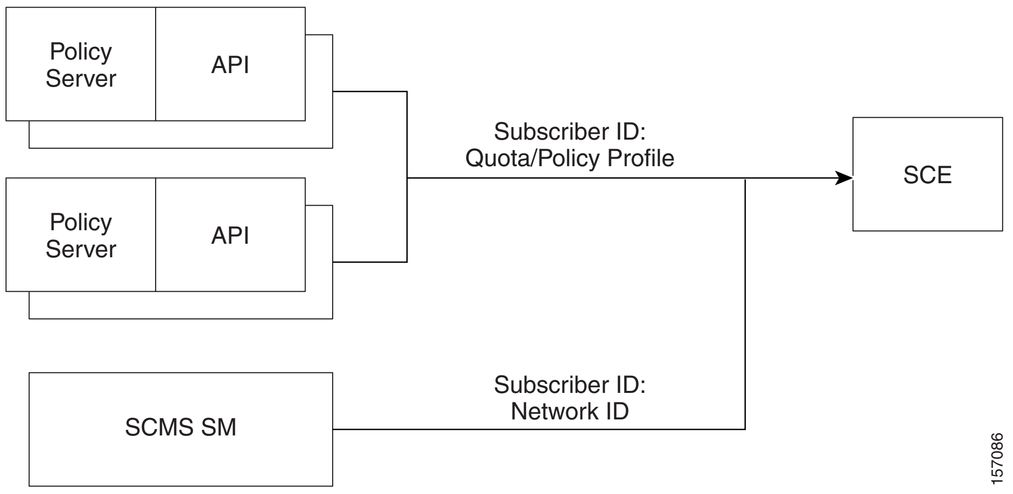
- DHCP Lease Query Login Event Generator (LEG), which is responsible for mapping a network ID to a subscriber ID, with one or more policy servers as shown in Figure 2-7. Figure 2-8 shows the DHCP Lease Query LEG.

Figure 2-8 Supported Topologies - DHCP Lease Query LEG



- The Cisco Service Control SM, which is responsible for mapping network ID to subscriber ID, with one or more policy servers as shown in Figure 2-9. The number of policy servers depends on whether the Subscriber Manager is used for policy profile provisioning in addition to the network ID.

Figure 2-9 Supported Topologies - Subscriber Manager


Policy Server
API
Policy Server
API
Subscriber ID: Quota/Policy Profile
SCE
SCMS SM
Subscriber ID: Network ID
157086

Caution The API itself does not limit the use of any topology; however, the Cisco SCE platform does not correlate among all the entries (policy servers) that perform subscriber provisioning. Therefore, be careful when using more than one policy server for the **same provisioning purpose** (for example, network ID/subscriber ID correlation). If you are not careful when using more than one policy server, the Cisco SCE platform can receive different information for the same subscriber from the two policy servers responsible for the same aspect of the subscriber provisioning. This can cause a loss of synchronization with at least one policy server. For example, if two policy servers correlate subscriber ID to network ID for the same subscriber, the Cisco SCE always synchronizes with the policy server that performed the last update for this subscriber.

Multithreading Support

The API supports an unlimited number of threads calling its methods simultaneously. (Only available memory limits the number of threads.)

Note In a multithreaded scenario, the order of invocation is **guaranteed**. The API performs operations in the same order in which they were called.

Autoreconnect Support

The API supports autoreconnection to the Cisco SCE in case of connection failure. When this option is activated, the API can determine when the connection to the Cisco SCE is lost. When the connection is lost, the API activates a reconnection task that repeatedly tries to reconnect to the Cisco SCE following a configurable time interval, until reconnection is successful.

Reliability Support

The Cisco SCE Subscriber API is implemented as a reliable API. The API ensures that no requests to the Cisco SCE are lost and no indication from the Cisco SCE is lost. The API maintains internal storage for all API requests that are sent to the Cisco SCE. Only after receiving an acknowledgment from the Cisco SCE that the request was processed does it consider the request as committed**.** The API can then remove the request from its internal storage. If a connection failure occurs between the API and the Cisco SCE, the API accumulates all requests in its internal storage until the connection to the Cisco SCE is re-established. On reconnection, the API resends all noncommitted requests to the Cisco SCE, ensuring that no requests are lost.

Note The order of resending requests is guaranteed. The API resends the requests in the same order in which they were called.

High-Availability Support

The API provides high-availability support. It assumes that the high-availability scheme of the policy server is a two-node cluster type in which only one server is active at any time. The other server, in standby, is not connected to the Cisco SCE. For more information, see the "Implementing High Availability" section on page 5-52.

Synchronization

The Cisco SCE and policy server must be kept synchronized regarding the subscribers for which the Cisco SCE manages internal parameters. Otherwise, the Cisco SCE can confuse the traffic of one subscriber with the traffic of another; or the service level agreement (SLA) of the subscriber might not be enforced because of a change in the policy that did not reach the Cisco SCE. For more information, see the "Cisco SCE-API Synchronization" section on page 5-45.

Practical Tips

When implementing the code that integrates the API with your application, consider the following practical tips:

- Connect once to the Cisco SCE and maintain an open API connection to the Cisco SCE at all times, using the API many times. Establishing a connection is a timely procedure, which allocates resources on the Cisco SCE side and the API client side.
- Share the API connection among your threads—it is better to have one connection per policy server. Multiple connections require more resources on the Cisco SCE and the client side.
- Do not implement synchronization of the calls to the API. The client automatically synchronizes calls to the API.
- If the policy server application has bursts of logon operations, increase the internal buffer size accordingly to hold these bursts (nonblocking flavor).
- During the integration, use the logging capabilities that are described in the "Cisco SCE Logging" section on page 6-2 and in the "API Client Logging" section on page 6-6. View the API operations in the Cisco SCE client logs to troubleshoot problems during the integration.
- Use debug mode for the policy server application that logs or prints the return values of the operations.
- Use the automatic reconnect feature to improve the resiliency of the connection to the Cisco SCE.