Oracle® Communications Service Broker

VPN Implementation Guide

Release 6.0

E23531-02

March 2012

ORACLE®

Oracle Communications Service Broker VPN Implementation Guide, Release 6.0

E23531-02

Contents

3 Configuring Charging for VPN Services

4 Provisioning VPN Services

5 Monitoring VPN Services

6 Using Batch Operation Tools

7 Using the VPN Provisioning API

8 VPN Provisioning API Reference

Preface

This document describes how to install, configure, administer, and use the Oracle Communications Service Broker Virtual Private Network (VPN) software.

Audience

This document is intended for system administrators, developers, and system integrators who will set up or administer Service Broker VPN.

This documentation assumes that you are familiar with:

- The operating system on which your system is installed
- Telecom networks and protocols, especially SIP and SS7-based protocols
- Clustering concepts and life cycle management
- Web servers and protocols, especially RESTful

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following documents in the Oracle Communications Service Broker Release 6.0 documentation set:

- *Oracle Communications Service Broker Release 6.0 Release Notes*
- *Oracle Communications Service Broker Release 6.0 Concepts Guide*
- *Oracle Communications Service Broker Release 6.0 Installation Guide*
- *Oracle Communications Service Broker Release 6.0 Processing Domain Configuration Guide*

- *Oracle Communications Service Broker Release 6.0 Signaling Domain Configuration Guide*
- *Oracle Communications Service Broker Release 6.0 System Administrator's Guide*
- *Oracle Communications Service Broker Release 6.0 SVC Implementation Guide*

Downloading Oracle Communications Documentation

Oracle Communication Service Broker documentation is available from the Oracle software delivery web site:

http://edelivery.oracle.com/

Additional Oracle Communication documentation is available from Oracle Technology Network:

http://www.oracle.com/technetwork/index.html

1

Service Broker VPN Overview

This chapter introduces Oracle Communications Service Broker VPN (Virtual Private Network). It describes the steps for implementing Service Broker VPN services.

About Service Broker VPN

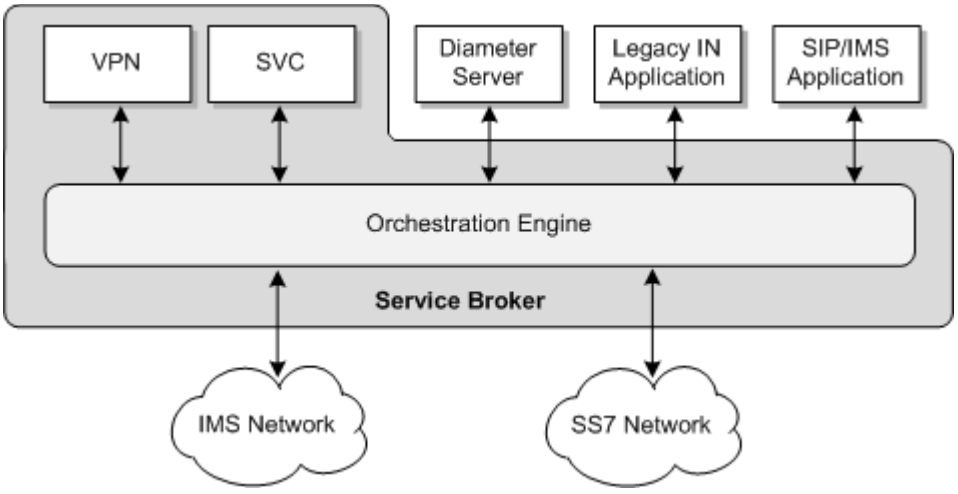
Service Broker VPN is an optional component of the Oracle Communications Service Broker that offers VPN capabilities and features traditionally associated with PBX-based voice VPN systems to mobile devices.

Service Broker VPN makes traditional PBX-based VPN features (such as private extension dialing, calling line identity presentation, and reduced charging rates) available to mobile device users even while away from their home office, city, or country.

Network operators use Service Broker VPN to offer voice VPN services to subscribing organizations. Subscribers typically benefit from reduced charging rates for on-net calls, improved employee connectivity, and a consistent user experience for their employees across mobile and desktop telephones.

The Service Broker VPN application is installed as an integrated component of the Service Broker. As shown in Figure 1–1, the VPN application resides in the Service Broker processing tier domain, where it operates alongside other application-level components such as Service Broker Social Voice Communicator (SVC).

Figure 1–1 The VPN Application in a Service Broker Deployment



The VPN application is tightly integrated with Service Broker, but in other ways, it can be regarded as any external application for which Service Broker performs mediation and orchestration.

Like other applications, the services of Service Broker VPN are integrated into the Processing Tier through an application-facing interworking module (IM). An IM is a Service Broker configuration component that serves as the mediation interface for external entities and Service Broker applications. The IM provides the mediation and session state management capabilities that, along with Signalling Tier components, make the VPN services available over multiple networks, including SS7 and IMS networks.

Feature Overview

Service Broker VPN services can be adapted according to the requirements of each subscribing organization. Features of the service can be enabled or adapted based on the requirements of the organization. For example, while one subscribing organization may choose to use call control rules extensively, another may not.

The VPN Provisioning API gives VPN administrators configuration access to the features of Service Broker VPN application.

The features of the Service Broker VPN include:

- Short code dialing for mobile and PBX-based devices
- Forced on-net call handling, which provides users who dial the public number of another on-net user with on-net call features, including differentiated charging rates.
- *N*-level user groups and profiles, which help administrators manage their VPN policies in large or small enterprises.
- Support for Diameter-based online and offline charging applications
- Line identity presentation for private number calling
- Blocking of incoming or outgoing calls based on configurable black list rules
- Call duration control
- Private calling, allowing users to bypass call restrictions or other types of call processing for personal calls
- Partner VPN relationships, which allow the provider to apply differentiated charging rates to calls between two subscribing organizations
- Virtual users, which allocates on-net extension to external users

In addition, the Service Broker VPN provides tools that ease configuration and management of the VPN, such as policy profiles, user groups, batch tools, and more. The batch tools include the Batch Loader and Batch Validator, which are command-line utilities for loading and validating VPN subscriber and policy data.

Delivering VPN Services

From the VPN subscriber's perspective, VPN services are delivered as hosted software. The subscriber can acquire VPN services without having to install and maintain software or equipment.

From the VPN provider's perspective, implementing and offering VPN services involves the following steps:

1. Install the Service Broker VPN application with Service Broker.
2. Configure the Service Broker domains and managed servers.
3. Configure the database used to store VPN application data.
4. Verify and configure the basic operating settings of the Service Broker VPN application, such as database connectivity.
5. Add an IM instance for the Service Broker VPN application to the Service Broker configuration, orchestrating the IM with the network layer and other application layer services.
6. Use the VPN Provisioning API to configure the common properties for VPN services, such as provider properties and global policy rules. The VPN Provisioning API is the RESTful programming interface for configuring the services of the Service Broker VPN application.
7. For each subscriber, use the bulk provisioning tools to populate the data store with initial user data.
8. For each subscriber, provide access to the VPN self-management portal that the subscriber will use to configure their policy, add users, assign mobile device extensions, and so on. You develop the VPN self-management portal using the operations of the VPN Provisioning API.
9. Manage and monitor the VPN services deployment.

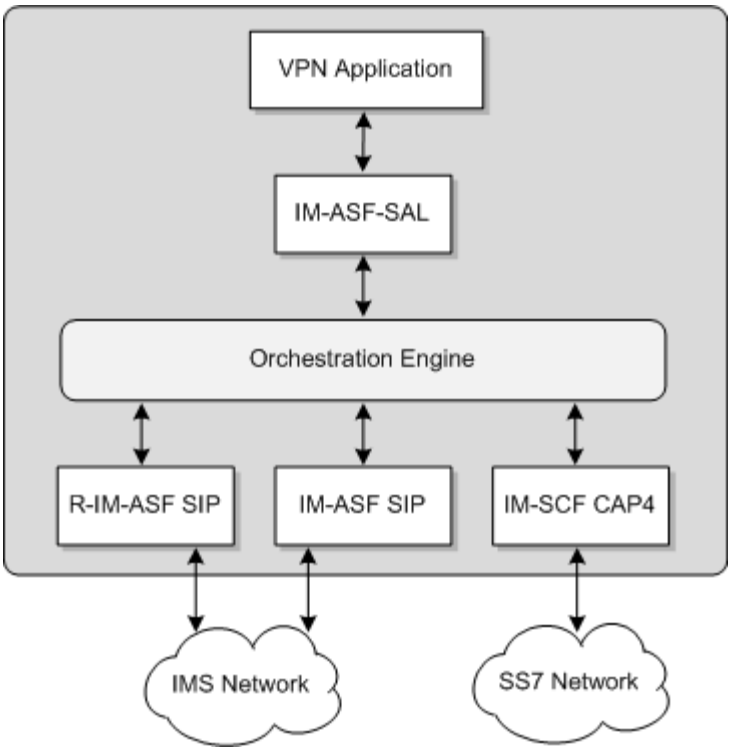
This guide provides information on each of these steps.

Functional Architecture

The VPN application integrates with the Service Broker configuration using a type of IM intended for use with Service Broker applications, IM-ASF-SAL. This type of IM provides similar functionality to the IM-ASF, which Service Broker uses to interact with external IMS applications and session control entities.

Figure 1–2 shows the Service Broker configuration components for connecting the Service Broker VPN application to the telecommunication networks on which it delivers services.

Figure 1–2 Service Broker Network Connectivity Components



For information on the complete Service Broker functional architecture, see *Oracle Communications Service Broker Concepts Guide*.

Configuration Tools and Interfaces Overview

To integrate the Service Broker VPN application with the Service Broker configuration, you use the Administration Console user interface. The Administration Console enables you to connect the VPN application to the communication networks on which it operates. You also use the Administration Console to orchestrate the VPN services with other applications for which Service Broker performs message brokering.

Like other Service Broker components, the VPN application presents its basic operating settings as JMX MBean attributes. These properties specify the application identity of the VPN application, its database connectivity settings, and how it generates accounting information for external charging servers.

You can connect to the MBean interface using any JMX client, such as JConsole or a custom MBean client application. The Service Broker scripting engine also provides configuration and read access to Service Broker MBean attributes.

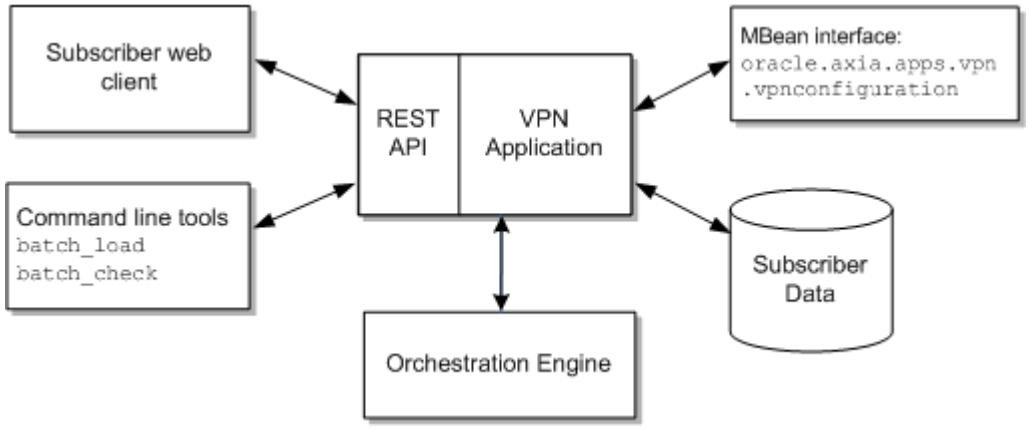
> **Note:** For complete information on the Administration Console and the MBean interface, see *Oracle Communications Service Broker System Administrator's Guide*.

The VPN Provisioning API is the primary interface for configuring the services provided by the VPN application. Using a client application, VPN administrators invoke API operations to add subscribing organizations, configure the VPN usage policy, define numbering plans, add users, and allocate device extensions, among other tasks.

The Service Broker VPN application stores the data generated by API clients in an external database accessed by JDBC connection. Statistical information generated by VPN activity can also be persisted.

Figure 1–3 shows the general access points for the configuration tools and other components in a Service Broker VPN application deployment.

Figure 1–3 Configuration Tools and Resources



2

Setting Up the VPN Application

This chapter describes how to configure the Oracle Communications Service Broker VPN application. It lists considerations for installing the VPN application software, setting up the VPN domain, and connecting VPN services to the external network.

Set Up Tasks Overview

The tasks covered in this chapter take you from installing the Service Broker VPN application software through connecting it to the communications networks on which it will run. At that point, Service Broker VPN services are ready for delivery to subscribing organizations.

Setting up the VPN application involves these tasks:

1. Install Service Broker VPN software as a Service Broker product installation option.

   See "About Service Broker VPN Software Installation" for installation considerations.

2. Create and secure the Service Broker VPN domains.

   See "Considerations for the Service Broker VPN Domain" for information about Service Broker VPN domains.

3. Configure the data persistence mechanism. The Service Broker VPN application stores provisioning and statistical information in persistent storage.

   See "Configuring Data Persistence" for information about setting up data storage.

4. Configure basic Service Broker VPN application settings using its MBean interface.

   See "Configuring Basic VPN Application Properties"

5. Enable VPN Provisioning API connectivity by opening an HTTP port in the Signaling Tier configuration.

   See "Opening an HTTP Listening Port for the VPN Provisioning Service" for information about configuring the Signaling Tier for VPN Provisioning API connectivity.

6. Configure the application-facing IM instances used to integrate VPN services with the Service Broker functional architecture.

   See "Creating the Application-Facing IM for VPN Services" for information about configuring the modules.

7. Configure the network-facing IM instances and other settings needed to connect VPN services to the SIP and SS7-based networks.

See "Connecting VPN Services to Telecommunications Networks" for information about configuring network-facing settings.

8. Configure orchestration for routing messages from the network layer to the VPN application and related services, such as IM- or IM-OCF services.

   See "Orchestrating VPN Application Services" for orchestration information.

9. If using location-based features, such as call barring or screening based on location, configure location awareness features in Service Broker VPN.

   See "Configuring Location Interrogation through HLR/HSS Systems" for information about enabling location-based features.

10. Configure external network components to accommodate VPN services in the telecommunications network.

    See "Triggering VPN Services from the Network" and "Configuring Originating and Terminating Service Invocation" for external network considerations.

About Service Broker VPN Software Installation

To install the VPN application software, you select it as an included component of the Service Broker installation in the Oracle Universal Installer.

In the Graphical Installation Mode, you include the VPN application software by selecting it in the **Select a Product to Install** page. On the page, the VPN application component appears as **Virtual Private Network 6.0.0.0.0**.

You can also install the VPN application by selecting the **Co-deployed Virtual Private Network and Social Voice Communicator 6.0.0.0.0** component.

When using the silent installer script, include the VPN application in the installation by setting the **INSTALL_TYPE_FM** value to:

- **5** to install both the VPN and SVC application
- **6** to install the VPN application

After installing the VPN application software, you can remove it by starting the Oracle Universal Installer and clicking the **Deinstall Product** button. In the product list, choose the VPN application from the Oracle home.

For complete information on installing Service Broker software, see *Oracle Communications Service Broker Installation Guide*.

Considerations for the Service Broker VPN Domain

The Service Broker VPN application is a component of the Service Broker processing domain. Therefore, before you can start configuring and using Service Broker VPN, you must create a processing domain or basic domain, which incorporates the components of both processing and signaling domains.

Oracle Communications Service Broker System Administrator's Guide contains general information on creating and managing Service Broker domains. This section describes specific considerations for working with domains when implementing Service Broker VPN.

Creating a Secure Domain

Before you create the VPN domain, you should verify the general security settings for the domain. The settings determine the security features that are enabled in the domain you create.

In addition to common security settings applicable to other types of Service Broker products, a Service Broker VPN domain is subject to security settings specific for connections for its RESTful API. In test and evaluation systems, it may not be necessary to secure the connection. In all other cases, the connection must be secured.

The security features applicable to the API connection include Secure HTTP (HTTPS) and SSL-based client certification authentication.

> **Note:** In addition to SSL-based client certificate authentication, the VPN application authenticates client requests based on user name and password values passed in HTTP headers of API requests. See "Authentication and Authorization" for more information.

The security features are controlled by property file settings. By default, the security settings are enabled, so that the domain is configured for maximum security. You should verify and modify the settings if needed.

The following Service Broker properties control the connectivity between API clients and the Service Broker managed server, which serves the RESTful API:

- **org.eclipse.equinox.http.jetty.https.enabled**: Set to **true** to use HTTPS for the API connection.
- **org.eclipse.equinox.http.jetty.https.ports**: Set to the port number you want to use for the secured HTTP connection.

Set these properties in the properties file located in *Oracle_home***/ocsb60/admin_console/properties**:

- For a web domain, edit the **web.properties** file.
- For a hosted domain, edit the **hosting.properties** file.

You must import the client certificate of each VPN API client into each managed server's trusted certificate keystore.

In addition, to allow the client to verify the identify of the managed server when negotiating the HTTPS connection, you must import the managed server's SSL certificate into the client's trusted keystore.

For details on creating and managing security certificates, see *Oracle Communications Service Broker System Administrator's Guide*.

When creating a secured domain, you must use **https** as the protocol value in the Domain URL that you pass to the domain creation script.

See *Oracle Communications Service Broker System Administrator's Guide* for complete information on securing the Service Broker installation.

Running the Domain Creation Script

To create a domain for Service Broker VPN, you use the common domain creation script:

*Oracle_home***/ocsb60/admin_console/create_domain.sh**

When prompted for a domain type to install, select either **Virtual-Private-Network** or **Co-Deployed-Virtual-Private-Network-and-Social-Voice-Communicator**. You choose the domain type by entering the index number that corresponds to your selection, as indicated in the script output.

For a Service Broker VPN domain, the script prompts you for a password for the built-in VPN provider administrator, as follows:

```
Enter a value for parameter 'admin.password':
```

The password you enter is stored in the Service Broker credential store. For information about the credential store, see *Oracle Communications Service Broker System Administrator's Guide*.

For general information about creating domains, see *Oracle Communications Service Broker Installation Guide*.

Configuring Data Persistence

Before creating the Service Broker VPN domain, you must set up the database that Service Broker uses to store VPN application data. Application data includes the policy configuration and information for the end users of VPN services. It also includes statistical data generated by the operation of the VPN services.

The VPN application works with these Oracle Database versions:

- Oracle Database 10g Release 2
- Oracle Database 11g Release 1 or Release 2

Note that VPN application does not support Oracle Coherence in-memory storage or Oracle Berkeley DB file-based storage. You must use Oracle Database for persistence.

To configure data persistence, you first set up the database. You then configure the connection to the database in the Service Broker configuration, as described in the following sections.

Preparing the Database

The VPN application includes a database configuration script that sets up the tables and schema required for VPN services. The script itself invokes separate SQL scripts, which perform the actual configuration of the database.

A SQL script is available for the VPN service, as well as for the SVC one number and SVC voice mail services. In addition, a SQL script exists for creating the statistical data tables and schema that are common to all services.

Only run the SQL scripts applicable to your implementation. For example, if using VPN services only, just run the VPN SQL script and statistical data script.

Before configuring data persistence, make sure the database is available by network to Service Broker. Also, you should have created a user account in the database for Service Broker. Oracle recommends that you create a user account on the database specifically for the use of the Service Broker VPN service.

To set up the database:

1. Open the following properties file for editing:

   *Oracle_home***/ocsb60/admin_console/properties/create_db_table.properties**

2. Configure the profile database settings, as follows:

- **profile.db.server**: The IP address of the database server.
- **profile.db.port**: The port on which the database listens for TCP client connections, typically 1521.
- **profile.db.dbname**: The name of the database in which you want to store data, such as **orcl** for Oracle 11g. This is the database in which the script will create and format tables for the service and statistical data.
- **profile.db.user**: The name of a database user who has privileges to access this database.

For example:

```
profile.db.server=10.10.1.125
profile.db.port=1521
profile.db.dbname=orcl
profile.db.user=ocsb
```

3. Save and close **create_db_table.properties**.
4. Move any SQL script not relevant to your implementation from the SQL script directory to a backup location. The directory where the scripts are located is:

   *Oracle_home***/ocsb60/admin_console/scripts/ddl/**

   The database configuration script invokes any SQL script it finds in this directory.

   The scripts are:

   - **er_vpn.sql**: Creates database tables for VPN application services.
   - **er_stats.sql**: Creates database tables for statistical information generated by activities of the services.
   - **er_ons.sql**: Creates database tables for the SVC one number service.
   - **er_voicemail.sql**: Creates database tables for the SVC voice mail service.
5. Run the database configuration script:

   *Oracle_home***/ocsb60/admin_console/create_db_table.sh**

   The script connects to the profile database you specified in **common.properties**.
6. At the following prompt, enter the password associated with the database user you identified in **common.properties**:

   ```
   Please input password of profile database user
   ```

   The script reads and applies the SQL scripts in the **ddl** directory. Informational log messages on the operation of the script appear on the screen.

Now see "Configuring the Database Connection" to configure the connection from the Service Broker Processing Tier.

Configuring the Database Connection

After preparing the database, you can create the connection to the database in the Service Broker domain.

You can configure the database connection by using the Administration Console user interface or MBeans. The following instructions provide an overview of the procedure using the Administration Console user interface.

For complete information on configuring the database connection, see the information on data persistence in *Oracle Communications Service Broker Installation Guide*.

1. In the Administration Console user interface, expand the **OCSB** node in the navigation tree.
2. Expand **Domain Management**.
3. Expand **Data Store**.
4. Click **Persistent Stores**.
5. In the JDBC Store tab, click **New**.

   Note that the **New** button is not enabled unless the configuration is locked for editing in the user interface. For general information about using the Administration Console user interface, see *Oracle Communications Service Broker System Administrator's Guide*.
6. In the dialog box, enter the following values:
   - **CredentialKey**: Use **KEY**, the default credential key name assigned to the profile database user by the database configuration script.
   - **User**: Use **oracle_driver** for the connection name, unless you have modified the default driver reference attribute for the VPN application.
   - **Connection Factory**: In most cases, this should be the built-in driver, **oracle.jdbc.pool.OracleDataSource**.

   For the other fields, enter the values appropriate for your implementation. For more information on configuring the database connection, see *Oracle Communications Service Broker Installation Guide*.
7. Click **OK** to save your database connection definition to the working configuration.

Opening an HTTP Listening Port for the VPN Provisioning Service

To enable client access to the VPN Provisioning API, you configure an HTTP listening port in the Service Broker Signaling Tier, as follows:

1. In the Administration Console user interface, expand the **OCSB** node in the navigation tree.
2. Expand **Signaling Tier**.
3. Expand **SSU Web Services**.
4. Click the **General** item.
5. Click the **HTTP** tab.
6. In the **Server** subtab, click the **Network Access** subtab.
7. Click the **New** button.
8. In the dialog box, enter the values appropriate for your HTTP listener for the VPN Provisioning API. This includes the local server address and port that clients will use to access the VPN Provisioning API.

   For more information on configuring HTTP connectivity in the Web Services SSU, see *Oracle Communications Service Broker Signaling Domain Configuration Guide*.
9. Click **OK** to save the HTTP network access settings to the working configuration.

10. Click the **Security Context** tab.
11. For the root server context (/), change the default authentication mode to **NONE**. The default authentication mode, BASIC, should not be used with the VPN application, which uses its own authentication and authorization scheme.

After you commit the changes to the domain configuration, you must restart the managed Signaling Tier managed servers to have them listen for HTTP traffic on the new port.

Configuring Basic VPN Application Properties

After installing Service Broker with the VPN application, verify and, if needed, modify the VPN application basic operating settings. The VPN application exposes its basic operating settings as MBean attributes.

You can access the MBean interface by connecting to the Administration Console process with an MBean browser.

The MBean domain that contains the VPN application configuration is:

oracle.axia.app.vpn.vpnconfiguration

The **vpnConfig** MBean contains these three configuration categories:

- **callEngine**
- **general**
- **statConfig**

To access the MBean configuration attributes:

1. If is not running, start the Administration Console.
2. Using a JMX MBean browser, such as the Java Monitoring and Management Console (JConsole) tool, connect to the Administration Console process.

   See *Oracle Communications Service Broker System Administrator's Guide* for information on connecting to the Administration Console process to access configuration MBeans.
3. In the MBean browser, navigate to the **vpnConfig** MBean:

   oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.apps.vpn.vpnconfiguration,version=1.0.0.SNAPSHOT,name0=vpnConfig
4. Navigate to the **appAddress** MBean, which is under the **General** node, and enter the address of the VPN application. You will use this address to identify the application from the SAL IM.

   By default, it is **sip:vpn@oracle.com**.

The other **vpnconfig** attributes can typically remain at their default values. These attributes include:

- **imsList** and **inList** settings are used internally by the VPN application to identify the underlying network associated with a particular session. You should not modify these values.
- **presence** contains settings that the VPN uses to get mobile user location from HLR/HSS systems. See "Configuring Location Interrogation through HLR/HSS Systems" for more information on configuring HLR/HSS lookups with VPN services.

- **enableChargingRf** is the switch that directs Service Broker VPN to generate accounting reports for the purposes of offline Diameter-based charging. See "Configuring Offline Charging for VPN Services" for more information.
- **statConfig** lets you configure the busy period threshold value used to trigger notifications. See "Configuring CPS Threshold SNMP Traps" for more information.
- **maxNumOfRouterActors** is the number of router actors in the system. The default value is **2**. The minimum value is **1** and maximum is **100**. A router actor can be considered a dispatcher for call requests. This is a performance-related setting that in most cases should remain at its default value.

Creating the Application-Facing IM for VPN Services

The services of the VPN application are integrated with the Service Broker configuration through an IM in the processing tier. The VPN application works with the IM that serves as the interface to Service Broker applications, IM-ASF-SAL.

Service Broker routes call sessions initiated in external SS7 or IMS networks to the VPN application through an IM-ASF-SAL instance. The Service Broker VPN application initiates requests to external HSS or HLR system for user location interrogation through the IM as well. This section explains how to create the IM-ASF-SAL. See "Configuring Location Interrogation through HLR/HSS Systems" for information on creating the IM instance for the HLR/HSS system.

The IM-ASF-SAL is an application-facing IM specifically intended for use with Service Broker applications, such as Service Broker VPN and SVC.

To create the IM-ASF-SAL instance by using the Administration Console, follow these steps:

1. Lock the configuration for editing.
2. In the Processing Tier navigation tree, expand the **Interworking Modules** node.
3. Select **IM Management**.
4. Click the **New** button.
5. In the New dialog box, select **IMASFSAL** as the type of the module and type a name for the new module.
6. Click **OK** and, as indicated by the dialog box, click the **Commit** icon before modifying the module settings.
7. Lock the configuration again.
8. Select the IM instance you created from the navigation tree.
9. In the **SAL Application Address** field, specify the application address for the VPN application in SIP address format, such as **sip:vpn@oracle.com**.

   This is the address you configured in "Configuring Basic VPN Application Properties" as the **appAddress** for the VPN application.

   The other IM settings, which include session keep-alive and general SAL settings, are used to control connectivity between other Service Broker components and the VPN application component. In general, you can leave these settings at their default value unless you have specific requirements related to these settings in your network.

   For more information on the settings, see *Oracle Communications Service Broker Processing Domain Configuration Guide*.

10. Commit your changes to save the configuration and deploy it to the managed servers.

You can now configure orchestration for the VPN application.

Connecting VPN Services to Telecommunications Networks

The services provided by the VPN application work over SS7-based and IMS networks. To expose VPN services on those networks, you connect the VPN application to the network-facing IMs that connect to the external networks, as described in the following sections.

See "Configuration Walkthrough" for a sample walkthrough of the configuration steps described in this section.

Connecting to SIP Networks

The general steps for connecting VPN services to SIP networks through Service Broker are:

- Create, activate, and configure an R-IM-ASF instance to exchange SIP traffic with external SIP networks.
- Configure the SIP Signaling Server Unit (SSU) to accept calls from your SIP network nodes and route them to the R-IM-ASF that you created.

See "Configuration Walkthrough" for an example scenario of setting up SIP network connectivity.

Connecting to SS7 Networks

The VPN application works with CAMEL Application Part v4 (CAP Phase 4) networks. It does not work with CAP Phase 3 or earlier versions. Therefore, the network-facing IM type to use with the VPN application is the IM-SCF CAP Phase 4.

The general steps for connecting VPN services to SS7 networks through Service Broker are:

- Create, activate, and configure an IM-SCF-CAP Phase 4 instance to exchange SS7 traffic with external SS7 networks.
- Configure the SSU SIGTRAN to accept traffic from your SS7 network nodes and route them to the IM-SCF-CAP Phase 4 that you created.

By default, Service Broker invokes originating and terminating functions for a session in separate dialogs. This is the recommended mode, particularly if the VPN service is available over both IN and IMS networks, because IMS networks operate in this manner.

To have originating and terminating functions invoked in a single IN dialog (or transaction) instead, set the MTC suppression code parameter. In this case, instead of separately initiated originating and terminating service transactions, the VPN application invokes originating and terminating service in a single transaction. It also adds to the network the MTC suppression code value to the response message.

The external Service Switching Point (SSP) must check messages for the suppression code, and, if present, avoid invoking the terminating functions.

See "Configuration Walkthrough" for an example scenario of setting up SS7 network connectivity.

Orchestrating VPN Application Services

As with other application-facing IMs, to route messages from network-facing IM to the VPN applications, you must add a filtering criteria for the IM to the orchestration rule.

In the orchestration rule, identify the IM-ASF-SAL you created for the VPN application. Specifically, given an IM named **imasfsal_vpn**, you edit an SM-LSS module and add an initial filter criteria as shown in Example 2–1:

Example 2–1 OE IFC rule to the IM

```
<?xml version="1.0" encoding="UTF-8"?>
<Sh-Data>
 <Sh-IMS-Data>
  <IFCs>
   ...
     <InitialFilterCriteria>
      <Priority>1</Priority>
      <TriggerPoint>
       <ConditionTypeCNF>0</ConditionTypeCNF>
       <SPT>
        <ConditionNegated>0</ConditionNegated>
        <Group>0</Group>
        <Method>INVITE</Method>
       </SPT>
       </TriggerPoint>
      <ApplicationServer>
       <ServerName>sip:imasfsal_vpn.IMASFSAL@oracle.com</ServerName>
       <DefaultHandling>0</DefaultHandling>
      </ApplicationServer>
     </InitialFilterCriteria>
  ...
  </IFCs>
 </Sh-IMS-Data>
</Sh-Data>
```

See "Configuration Walkthrough" for a complete orchestration example.

Configuring Location Interrogation through HLR/HSS Systems

The VPN application can make call barring and call screening decisions based on the location of the called or calling party.

To enable the VPN service to make location-based decisions, you must enable HLR or HSS location lookups for the VPN services. The VPN application submits a location request when needed. For instance, for a call attempt on an SS7 network, if the VPN application cannot retrieve the location number of the called party from the IDP message, it submits a MAP Any Time Interrogation (ATI) command to the HLR through the IM-PSX configuration you specify.

Service Broker VPN assumes that the location number from a MAP ATI response uses the *Country_CodeArea_Code* format. For example, it interprets the value 861390 as country code: 86 and area code: 1390. If the MAP element cannot provide responses in this format, the internal Service Broker component that interprets the location information must be customized. For more information, contact Oracle.

To configure HLR or HSS lookups, follow these steps:

1. Create an IM-R-ASF-SAL instance for the VPN application to use to initiate location requests routed through the Service Broker.

2. Create an IM-PSX MAP instance that provides the interface to the external HLR or HSS system.
3. Configure orchestration so that messages to the R-IM-ASF SAL are routed to the IM-PSX you created.
4. Configure the presence attributes in the **vpnconfiguration** MBean.

   In the **oracle.axia.app.vpn.vpnconfiguration** MBean, navigate to the **presence** attributes.

   To access the settings, expand the following nodes: **vpnConfig**, **callEngine**, and then **presence**.
5. Verify and if needed modify the values for the following attributes:

Table 2–1 presence MBean Attributes

Name	Description
impsxDomain	The domain identifier of the PSX domain for the Service Broker. The default value is **ocsb-psx.net**.
notifyResFormat	The content type of HLR interrogation requests. The default value is **application/map-phase3+xml**.
ocsbHandlingModuleAddress	The identifier, or alias, of the IM-ASF SAL instance that will accept location requests initiated by the VPN application. The default value is: **sip:imasfsal_2.IMRASFSAL@ocsb.com**
subsinfoType	The type of information to be requested from the HLR/HSS system. The default value is: **presence;requested-info=mobile-location**
subsReqinitor	The SIP address of the request initiator. In this case, it should be the SIP address of the installation. The default value is **sip:vpn@ocsb.com**.
useHSSInIMS	Set to **true** if interrogating an HSS system instead of an HLR. This affects the format of the request generated by the VPN application. The default value is **false**.

For details on HLR/HSS integration, see the discussion of IM-PSX in *Oracle Communications Service Broker Processing Domain Configuration Guide*.

Triggering VPN Services from the Network

The mechanism used to trigger VPN application services for a particular call varies between mobile-originated and PBX-originated calls. The VPN application data model configuration includes an attribute to facilitate a network triggering mechanism.

For mobile-originated calls, the VPN service is typically triggered using the O-CSI record of the subscriber end user. This is the standard subscription-based triggering mechanism in which an O-CSI parameter identifies the services associated with a subscriber in the HLR/VLR for that subscriber.

For PBX-originated calls, the provider may use a dialed number trigger or a trunk-based trigger to trigger VPN services.

In trunk-based triggering, all calls that traverse a specific line between the VPN subscriber's PBX and the MSC trigger the VPN service. This requires that the line is used exclusively by the subscribing organization, and thus all calls on that line should trigger VPN services.

In dialed-number triggering, the SSP triggers the VPN service based on a numeric prefix in the destination number. To use this method, configure the PBX that is local to the subscriber to add an identifying numeric prefix to dialed numbers. The SSP should use this prefix to trigger the VPN service for those calls. When the VPN service receives the call, it removes the prefix from the dialed number before continuing its call handling procedure.

An attribute of the VPN service provider configuration named **pbxPrefix** identifies the prefix value to the VPN application.

See "Modify VPN Service Provider" in "VPN Provisioning API Reference" for information on how to set the **pbxPrefix** value.

Configuring Originating and Terminating Service Invocation

The call processing activities of the VPN application comprise both originating (O-CSI) and terminating (T-CSI) services. An originating service applies to on-net users who are initiating calls. A terminating service applies to on-net user who are receiving calls. For example, outgoing call barring is an originating service because it applies to the call originator. Incoming call screening is a terminating service because it applies to the called party.

Terminating services can be invoked either in the same IN dialogue as the originating services or in separate IN dialogues. By default, the VPN application is configured to invoke the originating and terminating services in separate dialogs. This matches the mode typically used in IMS networks and allows the greatest compatibility between SS7 and IMS session processing by Service Broker.

To have the session occur in a single dialog, you set the mobile terminated calls (MTC) suppression code in the VPN service provider configuration. This is controlled by the **mtcSuppressionPrefix** in the service provider configuration settings.

To have the session occur in a single dialog, you must also configure the SSP so that it does not redundantly invoke the terminating services. You can configure the SSP to suppress T-CSI based on the presence of the **MTC-Suppression-Prefix** header, which the VPN service adds as a prefix to the address in its response messages. The SSP should recognize this code, skip terminating service invocation, and strip the code from the address.

See "Modify VPN Service Provider" in "VPN Provisioning API Reference" for information on how to set the **mtcSuppressionPrefix** value.

Configuration Walkthrough

The procedures in this section list the steps for performing an end-to-end configuration of a new Service Broker installation to implement VPN services.

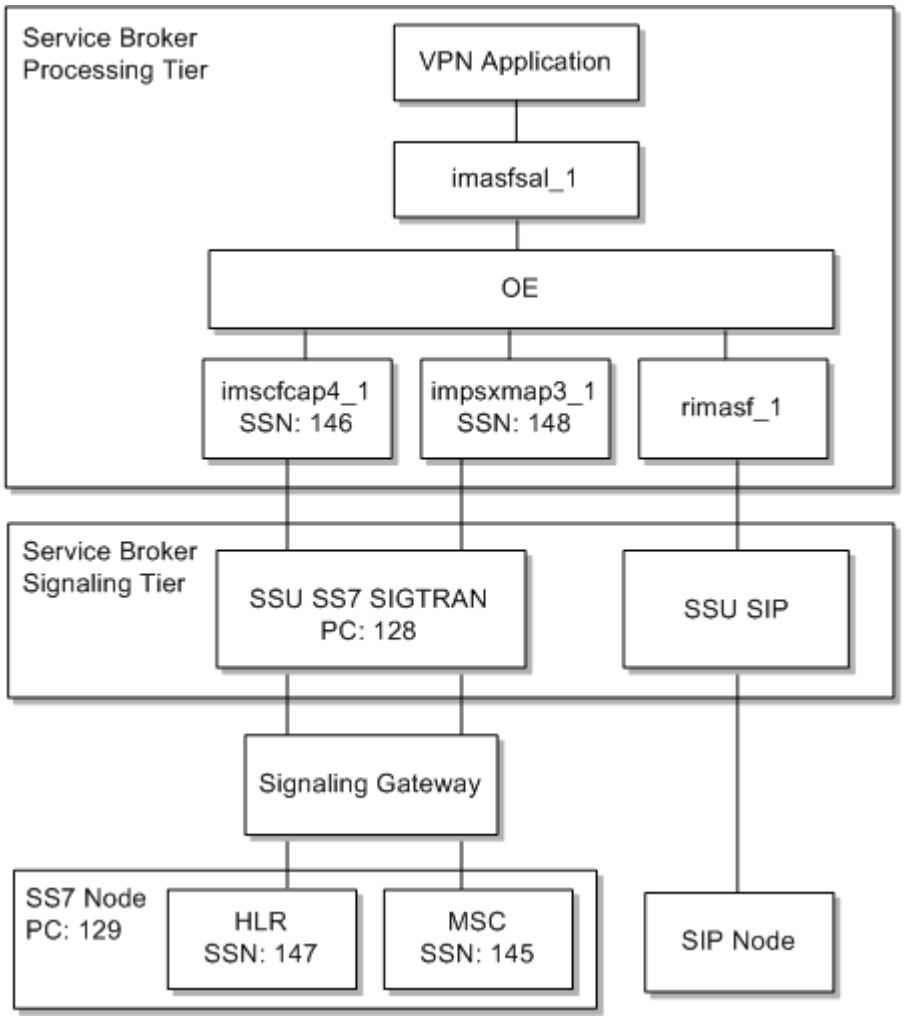
The steps take you through the complete Service Broker configuration from an initial, empty state through VPN service orchestration. The configuration integrates the services of the VPN application with IMS and SS7 networks, along with an external HLR system used to provide location lookup services.

The steps provide sample values for variable data such as module instance names and connectivity settings.

The exact steps, instance names, connection settings, and other values that you use for your implementation may vary based on your network environment and requirements. Nevertheless, the following steps can provide a guideline for the steps needed to perform your own implementation.

Figure 2–1 shows the network components and the Service Broker configuration that results from the sample configuration.

Figure 2–1 Sample Configuration Environment



The walkthrough is organized into three parts:

- Configuring the Processing Tier
- Configuring the Signaling Tier
- Configuring the Orchestration Engine Logic

Configuring the Processing Tier

To configure the Processing Tier by using the Administration Console:

1. Create an R-IM-ASF instance named **rimasf_1**. This IM connects the Service Broker Orchestration Engine layer to the external SIP network.
2. Create an IM-ASF-SAL instance named **imasfsal_1**. This IM instance serves as the application-facing IM between the Orchestration Engine layer and the VPN application.

3. Create an additional IM-ASF-SAL instance named **imasfsal_2**. The VPN application uses this IM to initiate location (presence) requests to external HLR or HSS systems.
4. Using a JMX client such as JConsole, set the **ocsbHandlingModuleAddress** attribute of the **presence** MBean to the name of the IM you created, **imasfsal_2**. See "Configuring Location Interrogation through HLR/HSS Systems" for more information.
5. Create an IM-PSX MAP Phase 3 instance named **impsxmap3_1**. This IM provides the interface to the external HLR or HSS system.

   In the case of VPN services, the VPN application uses location lookups to make call screening and barring determinations that are based on location, for example, if the policy bars international calling.
6. Create an IM-SCF CAP Phase 4 instance named **imscfcap4_1**. This IM integrates the Orchestration Engine layer with the CAP4 SS7 network.
7. Update the **imasfsal_1** configuration to refer to the default application identity for the VPN application:
   a. Go to **imasfsal _1** in the list of interworking modules.
   b. Set the **SAL Application Address** field to **sip:vpn@oracle.com**. If this address has been modified from the default, set to the value appropriate for your case. See "Configuring Basic VPN Application Properties" for more information about setting the application identity.
8. After committing the new **impsxmap3_1** instance, configure its settings as follows:
   a. In the **General** tab, set the **Alias** to **psx_map3**.
   b. In the **General** subtab of the **SIP Subscription** tab for the IM, set **PSX SIP Domain** to the domain appropriate for your SIP network, in the form shown by the default value, **ocsb-psx.net**. The **Generate pending NOTIFY** value can remain at the default, **TRUE**.
   c. In the **Map Handling** tab, set **gsmSCF Address** to the address that identifies the IM in the network.
9. In the General tab, update the **imscfcap4_1** configuration by setting the **Alias** value to **imscf_ss7**.

Configuring the Signaling Tier

To configure the Signalling Tier by using the Administration Console:

1. In the Signaling Tier navigation tree, select the **SSU SIP** node.
2. In the **SIP Server** tab, update the **Globally Routable User Agent URI** value to the value appropriate for the managed servers in your signaling domain by using this syntax:

   sip:*host*:*port*

   Where *host* is the externally routable host or IP address of the managed server or managed server cluster in your environment, and *port* is the port number at which the managed server listens for SIP traffic.
3. In the **Incoming Routing Rule** tab, configure an incoming routing rule to the **rimasf_1** IM instance. For example:
   - **Name**: **irr_1**

- **IP Address**: **any**
- **Alias**: **sip:rimasf_1.RIMASF@ocsb**

4. In the Signaling Tier navigation tree, click the **SSU SS7 SIGTRAN** node.
5. Click the **M3UA** tab.
6. In the **Local Point Code** tab, set the **Local Point Code** value to the point code in the network for the SSU node, for example, **128**.
7. Click the **Connectivity** tab.
8. In the **Local Systems** subtab, create a new route definition with the following values:
   - **Name**: An identifying name for the local system instance, such as **local_sys**.
   - **Routing Context**: **128**, or the local point code for the SSU.
   - **SS7 Mode**: **ITU14**
   - **Traffic Mode**: **loadshare**
   - **IP Address 1**: The externally routable IP address or host name for the managed server or managed server cluster.
9. In the **Remote Systems** subtab of the **Connectivity** tab, create a new route definition with the following values:
   - **Name**: An identifying name for the remote route, such as **remote_sys**.
   - **Type**: **sg**
   - **IP Address 1**: The IP address or host name of the external SS7 network entity to which Service Broker routes traffic.
10. Click the **M3UA** tab.
11. In the **Network Mapping** subtab, create a new SCTP association mapping with the settings appropriate for your system, as in the following example:
    - **Name**: An identifying name for the mapping, such as **to_sg**.
    - **Side**: **client**
    - **Type**: **m3ua**
    - **Local Port**: **2905**
    - **Remote Side**: **sg**
    - **Remote port**: **2905**
12. In the **Network Routing** subtab of the **M3UA** tab, create a new default network route with the settings appropriate for your network, as in the following example.

    In this case, it is assumed that both the MSC and HLR system are hosted on the same remote node, identified by the remote point code number of 129:
    - **Name**: An identifying name for the default SS7 network route, such as **to_msc**.
    - **Remote Point Code**: The SS7 point code for the remote element, such as **129**.
    - **Primary Remote SIGTRAN System**: The identifier of the mapping you created earlier, **to_sg** in this example.
    - **Secondary Remote SIGTRAN System**: To specify a backup system, enter the identifier of a secondary system to use if the primary is unavailable.

13. Expand the **SSU SS7 SIGTRAN** node and then click the **SCCP** tab.
14. In the **Local SSNs** subtab, create a new SSN definition for the CAP Phase 4 network with the following values:
    - **Name**: An identifying name for the subsystem definition, such as **imscf**.
    - **SSN**: The subsystem number associated with the local SCCP application, such as **146**.
    - **Description**: An optional textual description of the local subsystem definition, such as **imscf ssn**.
    - **Alias**: The alias given for the Service Broker subsystem, which in this case should match the alias you specified earlier for the **imscfcap4_1** IM instance (**imscf_ss7**).
15. In the **Local SSNs** subtab, create an additional SSN definition for the CAP Phase 4 network with the following values:
    - **Name**: An identifying name for the HLR subsystem definition, such as **local_hlr**.
    - **SSN**: The subsystem number associated with the local SCCP application, which in this case represents the HLR, such as **148**.
    - **Description**: An optional textual description of the local subsystem definition, such as **impsx ssn**.
    - **Alias**: The alias given for the Service Broker subsystem, which in this case should match the alias for the **impsxmap3_1** IM instance (**psx_map3**).
16. In the **Remote PC and SSN Addresses** subtab, create a new address definition for the remote MSC subsystem with the following values:
    - **Name**: An identifying name for the remote subsystem definition, such as **msc**.
    - **Network Indicator**: **National Network**
    - **SSN**: The subsystem number for the remote subsystem, in this case, for the MSC, such as **145**.
    - **Point Code**: The point code for the remote SS7 node that provides the MSC services, such as **129**.
    - **Description**: An optional textual description of the remote subsystem.
    - **Alias**: An alias for the remote subsystem, such as **msc**.
17. In the **Remote PC and SSN Addresses** subtab, create an additional new address definition for the remote HLR subsystem with the following settings:
    - **Name**: An identifying name for the remote subsystem definition, such as **hlr**.
    - **Network Indicator**: **National Network**
    - **SSN**: The subsystem number for the remote subsystem, in this case, for the HLR, such as **147**.
    - **Point Code**: The point code for the remote SS7 node that provides the HLR services, such as **129**.
    - **Description**: An optional textual description of the remote subsystem.
    - **Alias**: An alias for the remote subsystem, such as **hlr**.
18. Under the **SSU SS7 SIGTRAN** node, click the **Routing** tab and then click the **Incoming Routing Rules** subtab.

19. In the **Incoming Routing Rules** subtab, create two rules that route traffic between the SSU and the IM instances you created earlier, as follows:
    - The routing rule for the IM-SCF CAP Phase 4:
      - **Name**: An identifying name for the rule, such as **imscf_rule**.
      - **Priority**: **0**
      - **Module Instance**: The identifier of the IM SCF CAP4 IM instance you created earlier, **imscfcap4_1.IMSCFCAP4@ocsb**.
    - The routing rule for the IM-PSX MAP Phase 3:
      - **Name**: An identifying name for the rule, such as **map3_rule**.
      - **Priority**: **0**
      - **Module Instance**: The identifier of the IM SCF CAP4 IM instance you created earlier, **impsxmap3_1.IMPSXMAP3@ocsb**.
20. In the **Incoming Routing Criteria** subtab, create a criteria for each IM, as follows:
    - Select **imsf_rule** from the **Parent** list and create a new routing criteria with the following values:
      - **Name**: An identifying name for the criteria, such as **imscf_criteria**.
      - **Session Key**: **DEST_ADDRESS_ALIAS**
      - **Value**: The alias of the IM SCF CAP4 IM instance you created earlier, **imscf_ss7**.
    - Select the **map3_rule** from the **Parent** list and create a new routing criteria with the following values:
      - **Name**: An identifying name for the criteria, such as **map3_criteria**.
      - **Session Key**: **DEST_ADDRESS_ALIAS** application
      - **Value**: The alias of the IM-SCF CAP Phase 4 IM instance you created earlier, **psx_map3**.

Configuring the Orchestration Engine Logic

To configure the Orchestration Engine logic by using the Administration Console:

1. In the Processing Tier navigation tree, click the **Orchestration Engine** node.
2. Change the default value of **Subscriber Profile Receiver** to **OlpLSSInfoReceiver**. In this case, the subscriber profile is served locally, based on the orchestration profile you configure next.
3. In the Processing Tier navigation tree, click the **Supplementary Module** node and then **SM-LSS**.
4. Select the **default** orchestration profile and click **Update**.

   The **Update** dialog box appears. It contains the local iFC logic that controls the application chain applied by Service Broker. The **OLP Data** field contains the actual iFC text that specifies the orchestration logic.
5. Add the appropriate iFC text to the **OLP Data** field. Given the IM names used in this sample walkthrough, the iFC would be as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Sh-Data>
   <Sh-IMS-Data>
```

```
        <IFCs>
           <InitialFilterCriteria>
              <Priority>1</Priority>
              <TriggerPoint>
                 <ConditionTypeCNF>0</ConditionTypeCNF>
                 <SPT>
                    <ConditionNegated>0</ConditionNegated>
                    <Group>0</Group>
                    <Method>SUBSCRIBE</Method>
                 </SPT>
              </TriggerPoint>
              <ApplicationServer>
                 <ServerName>sip:impsxmap3_1.IMPSXMAP3@ocsb.com</ServerName>
                 <DefaultHandling>0</DefaultHandling>
                 <Extension>
                    <ForceB2B/>
                 </Extension>
              </ApplicationServer>
           </InitialFilterCriteria>
           <InitialFilterCriteria>
              <Priority>2</Priority>
              <TriggerPoint>
                 <ConditionTypeCNF>0</ConditionTypeCNF>
                 <SPT>
                    <ConditionNegated>0</ConditionNegated>
                    <Group>0</Group>
                    <Method>INVITE</Method>
                 </SPT>
              </TriggerPoint>
              <ApplicationServer>
                 <ServerName>sip:imasfsal_1.IMASFSAL@ocsb.com</ServerName>
                 <DefaultHandling>1</DefaultHandling>
                 <Extension>
                    <ForceB2B/>
                 </Extension>
              </ApplicationServer>
           </InitialFilterCriteria>
        </IFCs>
     </Sh-IMS-Data>
  </Sh-Data>
```

This completes the sample configuration of the VPN application. The VPN application is now integrated with Service Broker and its services and external network components. Network messages addressed to the managed server or managed server cluster that hosts the VPN domain should now reach the VPN application.

The remaining chapters in this book explain how configure and provision VPN services for your subscribers, how to enable your subscribers to use those VPN services, and how to charge them for the services they use.

3

Configuring Charging for VPN Services

This chapter describes how to integrate Oracle Communications Service Broker VPN services with external charging applications. It describes the charging information that Service Broker VPN services generate in the form of attribute-value pairs (AVPs).

About Charging

Service Broker VPN work with offline-based and online-based Diameter charging applications. Specifically, the Service Broker VPN application acts as a Diameter client (or Charging Trigger Function (CTF)) that reports session and event information related to VPN activities to external Diameter servers.

The VPN application reports charging information in the form of attribute-value pairs (AVPs). Each AVP contains an item of information about a call, such as the type of call, the public number of a called or calling party, and the private numbers of call participants.

The following listing shows a sample Diameter accounting request message generated by the VPN application. It shows the AVPs that are generated for a typical on-net to on-net call.

```
Dual-Invoicing: 1. CORPORATE_CALL
Service-Access-Type: 1. Originating
IN-Calling-Party-Address: 861391001001
Calling-Party-Private-Number: 1001
IN-Called-Party-Address: 1002
Real-Called-Number: 861391001002
Call-Type: 1: ONNET-TO-ONNET
```

About Online Charging

The VPN application's support for online charging (Diameter Ro) relies upon the common online charging functionality built into Service Broker.

For information about the Diameter Ro support in Service Broker, see the discussions related to the Online Charging Function (OCF) in *Oracle Communications Service Broker Processing Domain Configuration Guide*.

About Offline Charging

The VPN application supports offline charging (Diameter Rf) as defined by the following specifications:

- RFC 3588 (Diameter Base Protocol)

- 3GPP TS 32.299 [770] (Diameter Charging Applications)

In addition to standards-based AVPs, the VPN application generates custom AVPs that provide information specific to VPN services. See "Customized VPN Diameter Rf AVPs" for a list of the AVPs.

The service-specific information in the AVPs enables providers to apply differentiated charging rates to the different types of calls on the VPN.

The charging information distinguishes between these types of calls:

- Calls between on-net members
- Calls between PBX and mobile on-net members
- Calls between on-net members and virtual users
- Calls between on-net members and members of a partner VPN subscriber
- Calls between on-net and off-net members

The VPN application reports VPN activities as either session-based or event-based incidents. A session-based incident is one that occurs over time, such as successful phone call. An event-based incident is short-lived, such as a failed call attempt. Unlike a session-based incident, an event-based incident does not have a meaningful beginning, middle, or end for charging purposes.

The VPN application generates session-based Diameter Accounting Requests (ACRs) for a successful call at:

- The start of a session (ACR START). The VPN application considers a call session started when the call is established successfully.
- Periodic intervals during the session (ACR INTERRIM).
- The end of the session (ACR STOP). The VPN application considers a call ended when it receives a disconnect message.

Event-based incidents are reported as ACR EVENT messages.

Configuring Offline Charging for VPN Services

The Service Broker VPN application uses connections in the Signalling Tier to exchange messages with external Diameter servers. Therefore, to enable offline charging, you must configure the Diameter server connection settings in the Signaling Tier. You also configure the network-facing IMs and orchestration rules in the Processing Tier that integrate the VPN application with the external Diameter server.

The Diameter AVPs that are generated by the VPN application are defined by MBean attributes. To implement Rf charging, you must add the AVP definitions that are specific to VPN services to the common set of Service Broker AVP definitions.

To add the VPN AVPs to the common set of AVP definitions, you use the Service Broker scripting engine. The scripting engine is a configuration tool that allows you to access and modify Service Broker MBeans from a shell script.

You use the scripting engine by running the `script.sh` script, passing it the command file appropriate for your deployment. The domain configuration must not be locked by another process when you run the script, such as by the Administration Console user interface.

> **Note:** For complete information about using the Service Broker scripting engine, see *Oracle Communications Service Broker System Administrator's Guide*.

To configure offline charging for VPN services:

1. Copy the AVP definitions appropriate for your deployment to the common signalling tier MBean using the `script.sh` script, passing it the appropriate command file, as follows:

   - For VPN only, run the script with the **define_vpn_avps.xml** command file, as follows:

     ```
     ./script.sh scripts/define_vpn_avps.xml
     ```

   - For SVC and VPN, run the script with the **define_svc_vpn_avps.xml** command file, as follows:

     ```
     ./script.sh scripts/define_svc_vpn_avps.xml
     ```

   The script copies AVP definitions to the common Service Broker MBean:

   oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.protocol.diameter.common,version=1.0.0.1,name0=diameterCommonConfig

   > **Note:** For a Social Voice Communicator (SVC)-only implementation, you use the AVP script command file named **define_svc_avps.xml**.
   >
   > Note that running both the VPN-specific and SVC-specific script command files does not constitute a substitute for running the **define_svc_vpn_avps.xml** script command file, because the service-specific scripts overwrite each other's settings.
   >
   > That is, if deploying both SVC and VPN services, you must run the script with the **define_svc_vpn_avps.xml** file only.

2. At the following prompt, enter the local path to the Service Broker processing domain:

   ```
   Enter a value for parameter 'domain.path':
   ```

   The path should be the directory in which the **initial.zip** file and other domain artifacts are located. For example

   ```
   Enter a value for parameter 'domain.path': /home/oracle/ocsb60/domains
   ```

3. Start and connect to the Administration Console process with a JMX MBean client.
4. Configure event-based reporting by navigating to the following MBean:

   oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.apps.vpn.vpnconfiguration,version=1.1.0.0,name0=vpnConfig,name1=general,name2=enableChargingEventRf

5. Set the **enableChargingEventRf** attribute to **true** to enable event reporting or **false** to suppress event reporting.
6. Configure session-based reporting by navigating to the following MBean:

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.apps.vpn.vpnconfiguration,version=1.1.0.0,name0=vpnConfig,name1=general,name2=enableChargingSessionRf

7. Set the **enableChargingSessionRf** attribute to **true** to enable session reporting or **false** to suppress session reporting.
8. In the Service Broker Signaling Domain, configure connectivity to the external Diameter charging server to which you want to send accounting reports.

   For instructions on how to perform this configuration, see the information about configuring Diameter Signaling Server Units in *Oracle Communications Signaling Domain Configuration Guide*.

The Service Broker VPN application now generates accounting request messages based on the service activities, which are transmitted to the charging server specified in the Service Broker Signalling Tier configuration.

Customized VPN Diameter Rf AVPs

In addition to the standard AVPs generated for Rf charging (as listed in "About Offline Charging"), the VPN application generates custom AVPs that provide information specific to VPN services.

Table 3–1 lists the custom AVPs.

Table 3–1 Attribute-Value Pairs Specific for VPN

AVP Name	Code	Type	Description
Service-Information.IN-Information	30200	Grouped	The custom AVP group that provides IN service information not covered by 3GPP 32.299. The other AVPs in this table belong to this group.
Service-Access-Type	30201	Unsigned32	The service access type: ■ 1: ORIGINATING ■ 2: TERMINATING
Service-Key	30202	Unsigned32	The service key of the VPN service.
IN-Calling-Party-Address	30203	UTF8String	The calling party public number, for example, MSISDN or fixed line number.
Calling-Party-Private-Number	30204	UTF8String	The private extension number of the calling party.
Calling-Party-VPNIdentifier	30205	UTF8String	The VPN subscriber identifier of calling party.
IN-Called-Party-Address	30206	UTF8String	The number that the VPN user dialed.
Real-Called-Number	30207	UTF8String	The actual number called, that is, the translated number. For example, if the user dials a private extension number, then this value shows the public number.
Called-Party-Private-Number	30208	UTF8String	The extension number of the called party.
Called-Party-VPNIdentifier	30209	UTF8String	The VPN subscriber identifier of the called party.
Calling-Vlr-Number	30210	UTF8String	VLR number of calling party.

Table 3–1 (Cont.) Attribute-Value Pairs Specific for VPN

AVP Name	Code	Type	Description
Called-Vlr-Number	30211	UTF8String	The VLR number of called party.
EventType-BCSM	30212	Unsigned32	The event type of the basic call state model (BCSM).
Call-Type	30213	Unsigned32	The type of call on the VPN, with one of the following values: ■ 1: ONNET_TO_ONNET ■ 2: ONNET_TO_PBX_ONNET ■ 3: ONNET_TO_VIRTUAL_ONNET ■ 4: ONNET_TO_PARTNER_ONNET ■ 5: ONNET_TO_OFFNET ■ 6: PBX_ONNET_TO_ONNET ■ 7: VIRTUAL_ONNET_TO_ONNET ■ 8: OFFNET_TO_ONNET ■ 9: PARTNER_ONNET_TO_ONNET
Dual-Invoicing	30214	Unsigned32	If private calling is enabled for the VPN subscriber, identifies the type of the call: ■ 1: CORPORATE_CALL ■ 2: PRIVATE_CALL
Network-Cause-Code	30215	UTF8String	The mapping of the cause code received from the network.
Reject-Cause	30216	Unsigned32	The reason for a rejected call. Used when the VPN service triggers incoming call screening, outgoing call barring, or call duration control, as identified by the attribute's value: ■ 1: INCOMING_CALL_SCREENING ■ 2: OUTGOING_CALL_BARRING ■ 3: CALL_DURATION_CONTROL ■ 4: NUMBER_TRANSLATION_ERROR

Standards-Based VPN Diameter Rf AVPs

The following sections list the standards-based AVPs generated by the VPN service.

VPN Diameter Rf AVPs from RFC 3588

The AVPs defined by RFC 3588 (Diameter Base Protocol) that the Service Broker VPN application generates are listed below.

Except where otherwise noted, the definitions of the attributes as generated by the VPN application conform to their definitions as stipulated by public specification.

For complete information about the definition or purpose of a specific AVP, see RFC 3588.

- **Session-Id**
- **Origin-Host**
- **Origin-Realm**

- **Destination-Host**
- **Destination-Realm**
- **Accounting-Record-Type**
- **Accounting-Record-Number**
- **Acct-Application-Id**
- **Acct-Interim-Interval**
- **User-Name**: For a mobile user, the value of this attribute is the International Mobile Subscriber Identity (IMSI) number for that user. For a PBX-based on-net user, the **User-Name** value is the user's public number.
- **Event-Timestamp**

VPN Diameter Rf AVPs from 3GPP TS 32.299

The AVPs defined by 3GPP TS 32.299 that the VPN application generates include the following.

For information about the definition or purpose of a specific AVP, refer to the 3GPP TS 32.299 specification.

- **Service-Context-Id**
- **Service-Information**
- **Service-Information.Subscription-Id**
- **Service-Information.Subscriber-Id.Subscription-Id-Type**
- **Service-Information.Subscriber-Id.Subscription-Id-Data**

4

Provisioning VPN Services

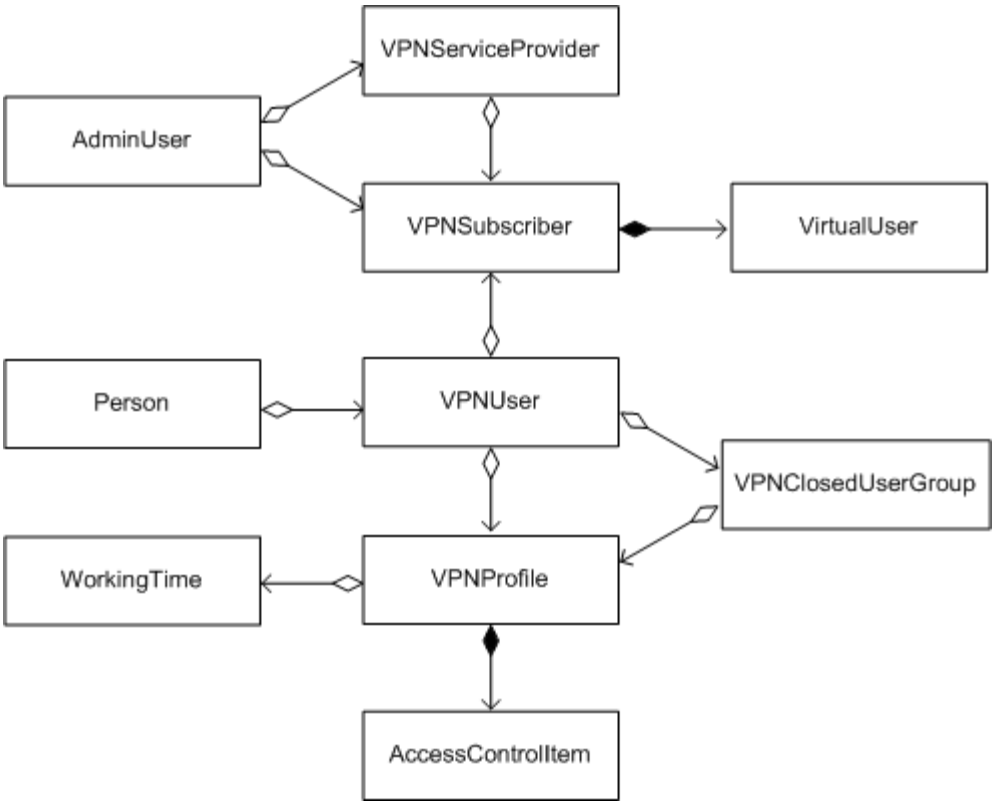
This chapter describes how to configure and provision the services of the Oracle Communications Service Broker VPN. It describes how to populate the VPN data model to configure and provision VPN services.

About the VPN Data Model

After installing the Service Broker VPN software and performing the initial setup, you configure and provision VPN services by populating the VPN data model by using the VPN Provisioning API.

Figure 4–1 shows an overview of the VPN data model.

Figure 4–1 Data Model Overview



The objects in the data model represent the logical components of the system, such as subscribers, user devices, end users, and policies. Collectively, the state of the data objects determines the features, behavior, and availability of VPN services.

In terms of the data model, the general steps for implementing VPN services are:

- Configure the settings of the VPN service provider.
- Add an administrative user as an **AdminUser** object for each subscriber organization to which VPN services will be deployed. Create additional provider and subscriber administrators as appropriate.
- Add a **VPNSubscriber** object for each VPN subscribing organization.
- Configure the private numbering plan and VPN features for a VPN subscriber as attributes of the **VPNSubscriber** object.
- To specify the policy, configure the usage policy for the subscriber by using **VPNProfile**, **AccessControlItem**, and **WorkingTime** objects.
- Add users for each subscriber by creating **VPNUser**, **Person**, and **VPNClosedUserGroup** objects.

In a production deployment, the tasks would be partitioned between the provider administrator and subscriber administrators. For example, provider administrators typically configure global VPN settings and create administrator accounts, whereas subscriber administrators typically add and manage end users, user devices, groups, and policies for their VPN subscription.

Subscriber administrator tasks should be exposed in the web application created for the purpose.

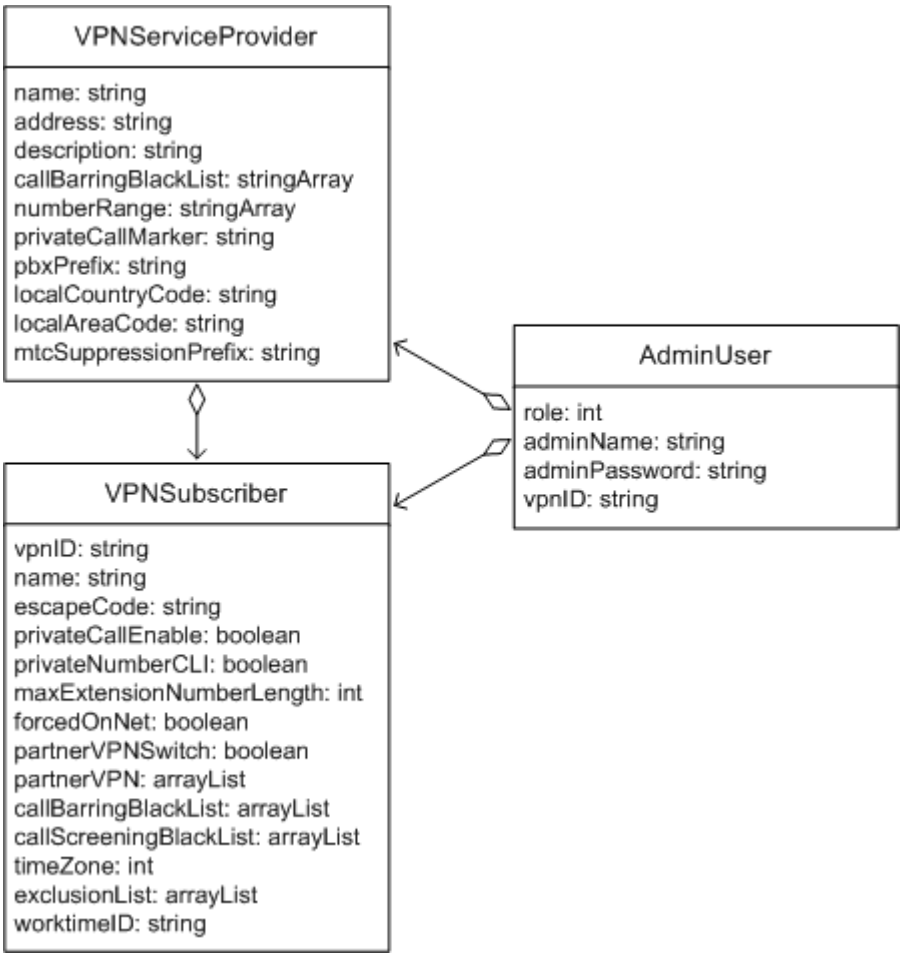
Configuring Provider Settings

The **VPNServiceProvider** object defines the provider organization, and includes information such as the provider name, address, and description. It also includes settings that have global scope, such as a global call barring list that applies to users across all VPN subscribers in the system.

Only one **VPNServiceProvider** instance exists in the data model. The service provider object is created automatically; you only need to configure it.

Figure 4–2 shows the attributes of the **VPNServiceProvider** and related objects.

Figure 4–2 VPNServiceProvider, VPNSubscriber, Administrator Objects



A service provider can have any number of **VPNSubscriber** object instances. The provider and each subscriber instance are associated with one or more **AdminUser** objects. A subscriber administrator identifies the subscriber for which it is the administrator by its **vpnID** attribute value.

To view the existing attributes of the provider, use the "Get VPN Service Provider" operation in the VPN Provisioning API. You can modify the provider settings using the "Modify VPN Service Provider" operation.

Managing Administrators

An administrative account represents an individual or application that performs the configuration, provisioning, and maintenance tasks for the VPN. When security is enabled for the VPN Provisioning API, requests to the API must contain the valid credentials of an administrative user. To create an administrator account, use the **AdminUser** object.

The data model defines two types of administrative accounts:

- Provider administrator (role 1)
- Subscriber administrator (role 4)

The administrator's role determines the privileges associated with the account, as described in the following sections.

A provider administrator can create additional administrative user accounts using the "Create Administrator" operation. Operations also exist for getting and modifying existing administrator account information.

About Provider Administrators

Provider administrators have full access to the operations in the VPN Provisioning API. The API operations can affect any subscriber instance in the system. For example, a provider administrator can add VPN users for any subscribing organization.

The VPN application services includes a built-in provider administrator named **admin**. After installing the VPN application software, you use this account to create additional administrator accounts.

Note: The admin user password is set when you configure the database used to store application data for the VPN application. See "Configuring Data Persistence" for more information.

The built-in **admin** user has several characteristics that user-created provider or subscriber administrators do not:

- It is created automatically.
- It cannot be deleted.
- Its attributes control whether API access requires authentication and whether the VPN Provisioning API service validates the data submitted with the Batch Loader tool.

To create additional provider administrators, you use the "Create Administrator" operation, specifying a **role** value of **1** for the new user. Note that the **role** value of an existing administrative user account cannot be modified.

About Subscriber Administrators

Subscriber administrative accounts are intended to be used by the individuals or applications in the subscribing organization that perform the subscriber-specific configuration and administration tasks for the VPN. For example, this administrator can create and manage policy profiles, configure subscriber settings, and add users and groups.

While provider administrators have full access to the operations in the VPN Provisioning API, subscriber administrators can access only a subset of those operations.

Furthermore, subscriber administrators can access the operations only in the context of their own VPN subscriber instance. That is, the API ensures that an administrator for one VPN subscriber cannot get or modify the data associated with other subscribers.

Before you create a VPNSubscriber instance, an administrator for the subscriber must exist. Therefore, the first step in provisioning VPN services to a newly subscribing organization is to create the administrator for the new subscriber.

To create a subscriber administrator, you invoke the "Create Administrator" operation, passing **4** as the new administrator's **role** value. Note that the **role** value of an existing administrative user account cannot be modified.

Adding and Configuring VPN Subscribers

A **VPNSubscriber** object contains the configuration settings and user data for a single organization. This object defines the private numbering plan, usage policy, user devices, and other VPN properties for the organization. You must create a **VPNSubscriber** object for each organization to which you are deploying VPN services.

Figure 4–2 shows the data model of the **VPNSubscriber** and adjacent objects in the VPN data model.

The VPN subscriber also defines the scope of on-net calling within the VPN. Calls between users within the subscriber organization are subject to number translation, reduced charging rates, and other on-net calling features.

To create a subscriber, first create the administrator for the **VPNSubscriber** instance, specifying the unique ID of the VPN subscription as the **vpnID** attribute of the administrator object. You cannot create a VPNSubscriber instance unless an AdminUser for it exists.

You create the **VPNSubscriber** object by using the "Create VPN Subscriber" operation, passing as its **vpnID** the same value you specified for the corresponding attribute of the administrator. Additional subscriber-related operations exist for getting all subscribers in the VPN, getting information on a specific subscriber, and deleting a subscriber.

After creating the subscriber object, the provider or subscriber administrator can configure the private numbering plan for the subscriber, and other features of the VPN, as described in the subsequent sections.

Partner VPNs

A partner VPN relationship defines an association between two or more subscribing organizations. Members of the partner organizations can be subject to reduced charging rates for calls between them. Also, subscriber administrators can define call screening or blocking conditions based on the partner relationship.

While charging differentiation may apply to partner calls, other features of the VPN, such as short code dialing, do not apply to partner VPNs.

To specify a partner relationship for a subscriber, you use the following attributes of the **VPNSubscriber** object:

- **partnerVPNSwitch**: Determines whether the VPN Partner feature is enabled for the subscriber.
- **partnerVPN**: Identifies the partner organization by VPN ID.

The VPN application distinguishes calls between partner organization users for purposes of call charging. The VPN application identifies calls from partner on-net users by using the call type PARTNER_ONNET_TO_ONNET.

Defining the Private Numbering Plan

The private numbering plan describes the schema for allocating device extensions, and other dialing properties in the VPN. Device extensions, or short codes, are the abbreviated numbers that on-net users dial to call other on-net users.

The subscriber administrator assigns an extension to a mobile device by creating a **VPNUser** object that represents the device. This object contains the private number of

the device (or on-net extension), as well as the actual, public number of the device. The Service Broker VPN uses this mapping to perform number translation for the device.

For a PBX device, the **VPNUser** object maps a range of public numbers to private extensions. In this case, the VPN user actually represents multiple devices.

In general, the design of the private numbering plan varies by subscriber, and is based upon factors such as the size of the organization and its existing numbering plan and network services. For example, a small organization can use shorter extension numbers, whereas a larger organization must use longer extension numbers.

An organization may choose to allocate mobile device extensions based upon its employees' existing desktop telephone extensions, for example, by adding a numeric prefix to the existing extension. Thus, an end user with a desktop number of 445 would get a mobile device extension of 8445.

Whereas the VPN self-management application contains the actual logic that allocates the user extensions, the **VPNSubscriber** object can specify restrictions on what those extension numbers can be. Specifically, the subscriber administrator can restrict the length of extensions and specify an exclusion list that blocks certain numbers or patterns of numbers from being allocated.

Restricting Extension Number Length

You restrict the length of extension numbers by using the **maxExtensionNumberLength** attribute of the **VPNSubscriber** object. The maximum extension length can be from 2 through 10 digits. The VPN application uses this value to validate new extension requests and to determine whether a dialed number is a public call or a private call.

When an on-net user dials a number that is longer than the maximum extension length, the VPN application immediately regards the number as a public number, without checking the private number store. If the dialed number is the same length or shorter than the maximum extension length, the VPN application first checks the store of allocated extensions, and regards the call as public only if the extension does not match one in the store.

Managing Extension Number Conflicts

A number allocated as a private extension must be unique within the organization's private telephone network. The subscriber should also avoid allocating numbers that conflict with public numbers.

The maximum extension length setting can help prevent conflicts between an organization's private extensions and public telephone numbers. However, telephone services may exist on the network with numbers that are the same length or shorter than the maximum extension length value. For example, an internal voice mail service or a public number such as the 911 emergency service in the U.S., present potential extension number conflicts.

The VPN Provisioning API provides the following mechanisms that help prevent extension number conflicts:

- Exclusion list: Specifies numbers and number patterns that cannot be allocated as extensions. If the API client application attempts to allocate an extension in the exclusion list, an error is generated and the attempt fails.
- Escape code: A numeric prefix which, if dialed by a user, directs the VPN application to forego number translation of the subsequently dialed number.

Using an Exclusion List

You specify an exclusion list using the **exclusionList** attribute of the **VPNSubscriber** object. The attribute can contain specific numbers or number ranges. To specify a range, use a wildcard symbol (*). For example, if an exclusion list includes **111***, the Service Broker VPN will not allocate any extension that starts with 111, such as 1112 or 1114. If the client application attempts to allocate an excluded number, the API returns the following error message:

```
Phone extension[1114] is not allowed because it conflicts with [111*] in
VPN's Exclusion List
```

Using an Escape Code

An escape code enables end users to make direct calls from an on-net device. By dialing an escape code before a number, the end user directs the VPN application to bypass the usual number translation logic.

For example, a network has a preexisting voice mail service at the number 113 and a mobile user at the same extension. Given an escape code of 8 in the subscriber's dialing plan, an end user can reach the voice mail service by dialing 8113. Alternatively, the end user can reach a device assigned the extension by dialing 113 without the escape code.

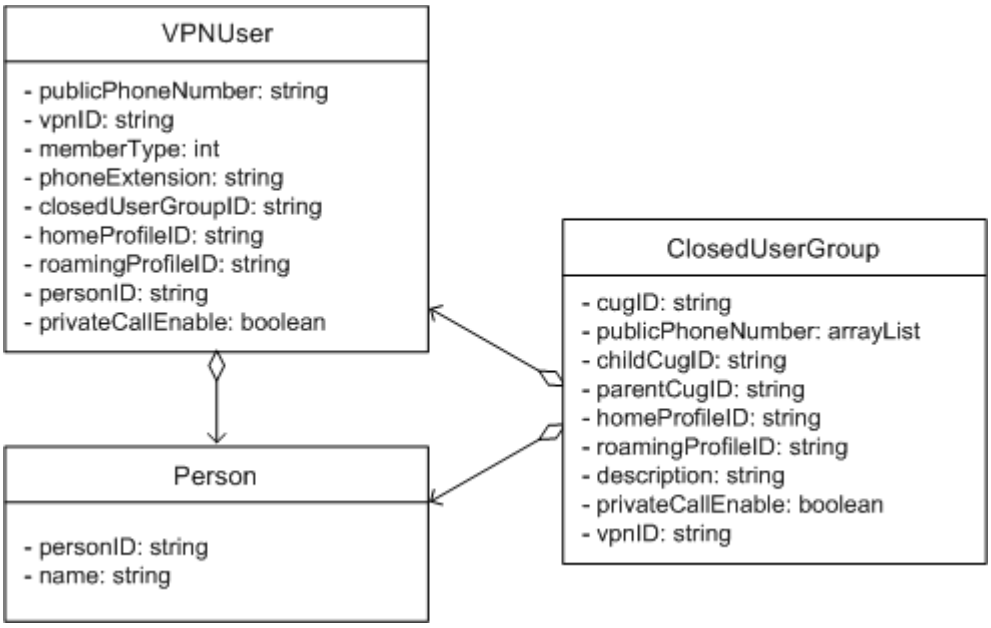
To specify an escape code, use the **escapeCode** attribute of the **VPNSubscriber** object. An escape code can consist of one or two digits. If you configure an escape code, be sure to configure an exclusion list that prevents the allocation of extensions that begin with the escape code digits.

Note that if the subscriber requires an escape code for off-net calls and permits private calling, users who make a call using the private call marker do not also enter the escape code. See "Enabling Private Calling" for information about private calling and the private call marker attribute.

Provisioning End Users

To provision an end user on the VPN, the administrator identifies the user's mobile device as an on-net device by using the **VPNUser** object. Figure 4–3 shows the data model for **VPNUser** objects and related objects.

Figure 4–3 Data Objects Related to Users



Each **VPNUser** object represents an end user's device (whereas the **Person** object represents the end user). The device represented by the VPN user can be a mobile device or a PBX device (or more typically, a range of PBX devices). Both types of users are represented by the **VPNUser** object. The **memberType** attribute determines whether the user is a mobile device or a PBX device.

A VPN user for a mobile device may optionally be associated with a **Person** object. The **Person** object represents a human user on the VPN. A VPN user can reference only one Person, but multiple VPN users can reference the same **Person** object. In other words, an end user can have multiple devices on the network, but a device can have only one owner.

A closed user group is a collection of VPN users. A group provides a convenient way to manage the policy for multiple users at a time. You specify a group with the **ClosedUserGroup** object.

You create users, virtual users, and other objects related to end users in the context of a particular subscriber. The subscriber to which the user belongs is determined by the URL path used to invoke the create operation; the subscriber is not specified as a parameter in the operation body.

The following sections provide more information about user-related objects.

Mobile VPN Users

The subscriber administrator must create a **VPNUser** object for each mobile VPN on-net device.

The VPN user maps a private extension on the VPN to a device's public telephone number. The VPN application uses this mapping to perform the number translation for calls to the device.

The VPN user can define the usage policy for the device (in the form of the home and roaming profile), or the device can be subject to the policy specified for its group.

Use the "Create VPN User" operation to create a **VPNUser** object for the mobile device, specifying **0** as the value of the **memberType** attribute.

PBX-Based VPN Users

A PBX-based user represents a set of PBX-connected telephones that, depending on the policy, the VPN service can regard as on-net devices.

Configuring a PBX-based VPN user is similar to configuring a mobile VPN user. In both cases, the **VPNUser** object maps a public number to a private extension. Also the same policy restrictions that apply to mobile devices can apply to PBX on-net devices. However, unlike a mobile device, the PBX user typically represents multiple devices.

To configure PBX devices with a VPN user, you use wildcards in its public number and private extension values. The VPN application uses the wildcarded portion of the number to match a private number to its public number.

For example, given a VPN user with a private number value of **3*** and a public number of **001415553***, the VPN application would route calls made to the extension 37722 to the public number 0014155537722. All other extensions belonging in the PBX system identified by 3 would be similarly mapped.

To create a PBX-based VPN user, invoke the "Create VPN User" operation, specifying **1** as the value of the **memberType** attribute.

Persons

The **Person** object represents a human user in the VPN. The end user attributes include the person's name, unique ID, and VPN instance.

An instance of a **Person** object is associated with a device through the **personID** attribute of the **VPNUser** object. A **Person** object can have any number of on-net devices. By associating devices to human users in this manner, the **Person** object enables administrators to list devices in the VPN by end user.

Use the "Create Person" operation to create a **Person** object. Use the **personID** attribute of the VPN user object to map the end user to a mobile device.

Closed User Groups

A **VPNClosedUserGroup** object represents a collection of one or more VPN users. User groups provide a convenient mechanism for managing policy for multiple devices at a time.

User groups can also provide conditions for on-net calling. That is, you can define an access control rule that permits or blocks a call depending on whether the call is between members of the same group.

User groups in the data model can be nested, providing inheritance for the restrictions defined in groups. A child group that defines its own restrictions also inherits the restrictions defined in the parent group, which may in turn inherit from its parent, and so on.

A user who belongs to a child user group is subject to the rules of both the user's immediate group and any parent group in the hierarchy to which the child belongs.

The VPN application applies group rules in a bottom-to-top manner. When evaluating a call, the VPN application first checks the call restriction rules in the user's immediate group. If the call does not match a rule in the user's group, it checks the rules in the groups immediate parent, and so on. When a rule matches the call, the rule action is applied without evaluating additional rules in a parent group.

A group cannot have more than one parent group, but a parent can have multiple child groups.

A general approach for designing nested groups involves defining parent groups that contain restrictions that are appropriate for a wide range of users, and child groups that specialize the restrictions for a subset of users.

You can create a closed user group using the "Create User Group" operation. You then add members to the group by identifying the group in the **closedUserGroupID** attribute in each **VPNUser** object to be associated with the group.

Virtual User

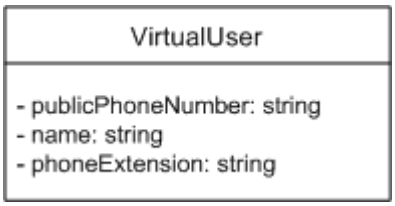
A **VirtualUser** object represents an external number that has a private extension in the VPN. The virtual user role is useful for external parties that VPN users must frequently call, such as a corporate travel agent or health care insurance provider.

Depending on the provider's charging plan implementation, calls to and from virtual users may be subject to reduced charging rates. The VPN application identifies calls involving virtual users in call accounting records with the call types VIRTUAL_ONNET_TO_ONNET and ONNET_TO_VIRTUAL_ONNET.

In addition to charging differentiation, the virtual user role can serve as a basis for access control decisions. The administrator may define access control rules that permit or block calls to virtual users based on call type, time of day, and other factors.

The **VirtualUser** object contains the settings for the virtual user. The object maps a private extension to the public number of the external user, as shown in Figure 4–4.

Figure 4–4 VirtualUser Object



You create a virtual user for a subscriber by invoking the "Create Virtual User" operation. The VPN Provisioning API also provides operations for getting all virtual users for a subscriber and for getting detailed information for a specific virtual user.

Controlling Calls

Administrators can control activities on the VPN using call control rules. For example, rules can limit the length of calls, block international calls, or permit calls only during working hours.

The VPN data model provides two primary mechanisms for specifying call control rules:

- Blacklists identify telephone numbers to be blocked either as outgoing call destinations or incoming call sources.
- Profiles can block or permit calls based on a variety of conditions, such as the time of day or the off-net status of the called party.

> **Note:** An end user can bypass a policy restriction by using the private call feature, if enabled for the subscriber. See "Enabling Private Calling" for more information.

Rule Evaluation Priority

Various objects in the VPN data model can define call control rules. The VPN application evaluates the rules by objects in the following order:

1. **VPNServiceProvider**
2. **VPNSubscriber**
3. **VPNUser**
4. **ClosedUserGroup**
5. Parent **ClosedUserGroup**

If the VPN application evaluates all rules without encountering a match, it permits the call. If the VPN application encounters a matching rule that specifies a blocking action, the call is immediately blocked without further rule evaluation.

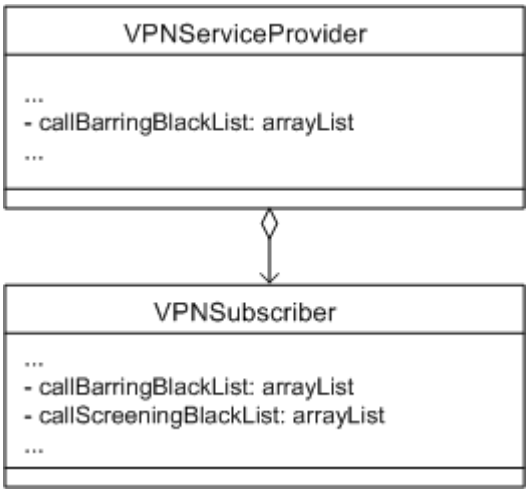
Note that if the VPN application encounters a matching rule that allows the call, subsequent rules are evaluated.

Using Call Barring and Call Screening Blacklists

A blacklist specifies numbers that are blocked as either call destinations or call sources. A call screening blacklist controls incoming calls, while a call barring blacklist controls outgoing calls.

As shown in Figure 4–5, the administrator can configure call barring blacklist items in either the **VPNServiceProvider** or **VPNSubscriber** object. The call screening blacklist is an attribute of the subscriber only.

Figure 4–5 Blacklist Attributes



The administrator can apply a blacklist by specifying a telephone number or number range as the blacklist attribute value. Number ranges are specified with wildcards (**"*"**). The following examples cause the blacklisting results indicated:

- A value of **"*"** blocks all calls. For example:

```
"callBarringBlackList":["*"]
```

- Prefixes that end in wildcards block calls that start with those prefixes (for example, "4321*" blocks "43211234" and "2281*" blocks "228138").

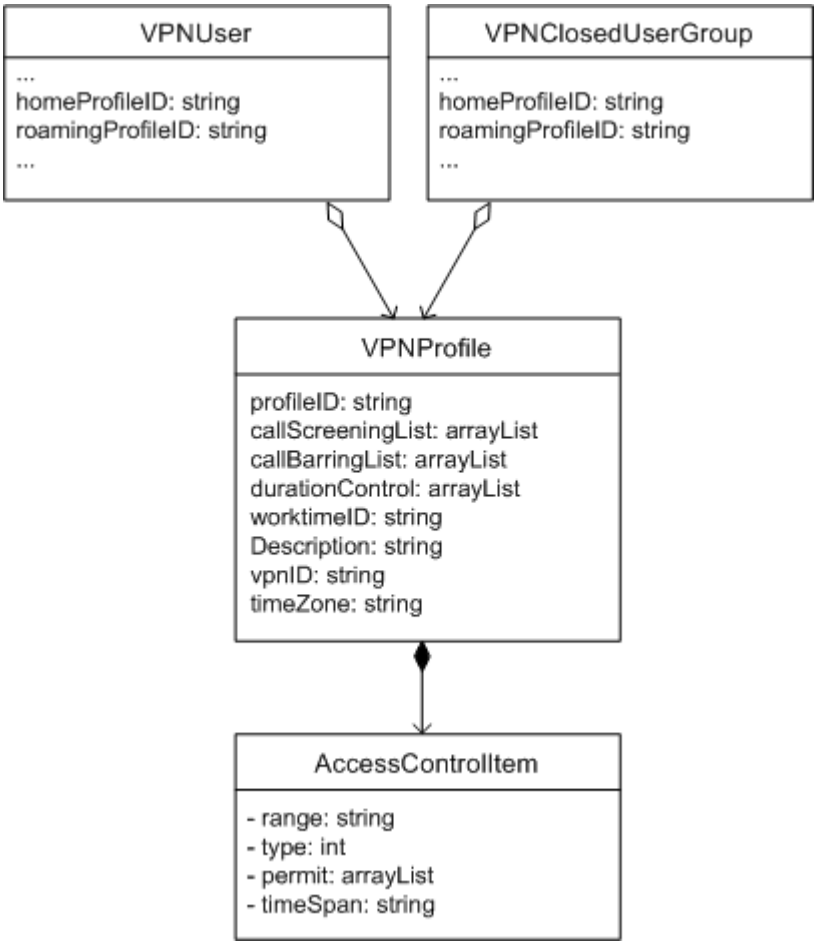
- Complete phone numbers (for example, "55523940" or "55523941") block calls specifically to those numbers.

Using Home and Roaming Profiles

A **VPNProfile** object defines a VPN usage policy. The policy can limit calls by duration or restrict calls by call type, time of day, or other conditions.

An administrator can apply different policies based on whether users are in-network or roaming. As shown in Figure 4–6, the **VPNUser** and **VPNClosedUserGroup** objects can apply profiles.

Figure 4–6 Data Model Related to Profiles



The **callScreeningList** and **callBarringList** attributes of the **VPNProfile** object contain and array of access control items. An access control item defines an access rule, which specifies a condition and the result of a match: whether the matched call is blocked or permitted.

Note that call screening and call barring blacklists, which are attributes of **VPNServiceProvider** and **VPNSubscriber** objects, define blocking conditions based on the called or calling number only. Access control rules, however, can block or permit calls based on a variety of conditions. See "Using Access Control Rules" for more information about access control rules.

When an access rule is matched, the server event log indicates the name of the matching profile and the reason for the match. The event log is written to the **server.log** file located in the managed server directory. The event log provides detailed information on activities of the managed server.

Forcing On-Net Status

By default, if an on-net user calls another on-net user by dialing that user's public phone number rather than the user's extension, the VPN application treats the call as an on-net to off-net call. The administrator can direct the VPN application to regard such calls as on-net calls by setting the **forcedOnNet** attribute of the **VPNSubscriber** object to true.

If forced functionality is enabled, instead of immediately treating a dialed number that exceeds the maximum extension length as a public number, the VPN application checks the data store for the number. If it finds that the dialed number is the public number of a user in the data store, the call is forced to on-net status.

All call processing functions and policies that apply to other types of on-net calls apply to forced on-net calls as well.

Enabling Private Calling

If private calling is enabled for a subscriber, end users can enter a private call marker before dialing a number. A private call marker is one or two numbers or symbols, such as an asterisk (*), and distinguishes the call as a private call.

A private call is a mobile-originated call that bypasses policy restrictions applicable to on-net calls. The VPN application identifies the call as a private call in charging records, allowing the provider to implement dual invoicing. In dual invoicing, the provider applies charges for privates calls to the private account of the end user.

In charging records, a private call is identified by the **Dual-Invoicing** attribute. For a private call, the value of this attribute is PRIVATE_CALL. Providers should use this value to implement alternate charging, in which an end user's personal use of the network is charged separately from the user's business-related use.

To initiate a private call, the end user dials the private call marker digits before dialing the number. The provider administrator specifies the value of the private call marker as the value of the **privateCallMarker** attribute of the **VPNServiceProvider** object. The private call marker value specified for the provider applies to all subscribers for which private calling is enabled.

Call restriction policies, caller identification, and other VPN features are bypassed for the call.

If the subscriber configuration requires the use of an escape code to dial off-net numbers, use the private call marker to preempt that requirement. That is, when making a call using the private call marker, the end user does not precede the public number with the escape code to reach the external destination.

Administrators can enable private calling for a subscriber by setting the **privateCallEnable** attribute to **true** in the **Subscriber** object.

Setting the Calling Line Identity Presentation Mode

Calling line identity presentation is a telephone service that presents the caller number to the called party's device, if the device is capable of displaying this type of

information. This enables the called party to identify the caller before answering the call.

The VPN application can present private numbers or public numbers as the call source to on-net call recipients. You can configure the identity presentation mode by using the **privateNumberCLI** property, which is an attribute of the **VPNSubscriber** object.

When **privateNumberCLI** is set to **true**, private identity presentation varies depending on the call type:

- For an on-net to on-net call and for an on-net to PBX on-net call, the VPN application presents the private number of the caller.
- For an on-net to virtual on-net, partner on-net, or off-net call, the VPN application presents the public number of the caller.
- For a PBX on-net to on-net call, the VPN application presents the private number (PBX extension) of the caller's fixed-line telephone.

When **privateNumberCLI** is set to **false**, the VPN application always presents the caller's public number to the called party.

5

Monitoring VPN Services

This chapter explains the monitoring features of the Oracle Communications Service Broker VPN. It describes how to capture and view statistics generated by activities of the VPN application.

About Monitoring

Service Broker VPN generates and collects information you can use to monitor VPN activities. The information include counter-based statistics, such as the number of calls originated by VPN users, the number of calls received by VPN users, and the number of calls to partner organizations.

In addition to counter-based statistics, the VPN application generates audit information in the form of the calls per second (CPS) log. The CPS serves as the charging basis for the VPN application.

Every 24 hours, the VPN application calculates the *busy hour* CPS value. This value represents the average number of successful calls per second during the busiest hour of the previous 24-hour period. It is derived by determining the busiest hour in the period, and then dividing the total call count by 3600.

The VPN application can generate SNMP trap notifications based on the CPS value. If the CPS threshold you configure is exceeded, the VPN application sends a trap to the monitoring agent.

You configure the CPS threshold and other settings that control how the VPN application performs monitoring-related tasks by using JMX-based configuration MBeans. To access configuration MBeans, connect to the Administration Console runtime process from a JMX browser.

Runtime statistics are available by connecting to the Processing Server process and invoking MBean operations to retrieve runtime statistics.

> **Note:** For more information about connecting to the Service Broker processes from a JMX client, see *Oracle Communications Service Broker System Administrator's Guide*.

Runtime information reflects the current statistics for the specific managed server you access. By default, every 5 minutes the VPN application collects the runtime information from all managed servers in the cluster and aggregates it in the statistics database.

The database preserves the data for a period of one year by default, enabling you to access the historical records of VPN application activities.

Accessing Historical Monitoring Data

The VPN application stores historical monitoring data in a database you configure for the purpose. The statistics information is stored in three tables:

- **stats_collect_info**: Stores the most recently collected statistical data. The collection time value in this table indicates when the data was collected.
- **stats_history**: Stores historical statistic data. The service writes to this table every 5 minutes by default. The information includes the CPS count, the calls per second count that is used for the busy hour CPS value.
- **stats_audit**: Stores the busy hour CPS value for each day.

See "Configuring Data Persistence" for information about connecting and configuring the external database used to store monitoring data.

Configuring Monitoring

You can configure how the VPN application performs monitoring related tasks by using the **statConfig** configuration MBean. The MBean interface is available by accessing the Administration Server process from a JMX client.

There are two MBeans relevant to statistics: one is common to both VPN and SVC and the other contains the attribute specific for VPN.

The object name of the common **statConfig** MBean is:

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.apps.common.statistic,version=1.1.0.0,name0=statConfig

The **statConfig** MBean has the following attributes:

- **archivePeriod**: The time span, in days, for which statistics are kept in data storage. This default value is 365. Statistics older than 365 days are purged from the database.
- **busyHourAverageCPSCheckTimeHour**: The daily hour at which the VPN application performs the CPS calculation. Can be from 0 to 24. By default, the calculation occurs at 2:00 AM.
- **busyHourAverageCPSCheckTimeMinute**: The daily minute at which the VPN application performs the CPS calculation. Together with the hour, this value determines the time of day the calculation occurs. Can be from 0 to 60. By default, the calculation occurs at 2:00 AM.
- **collectionDelay**: The collection delay, in seconds, between synchronizations of the VPN application statistics with the underlying Service Broker framework statistics. This ensures that data in memory awaiting collection by the platform data collection process is not lost. This value should match the **collectionPeriod** attribute in the **oracle.axia.logging.stats.statisticsConfig.collectionPeriod** MBean. Note that **collectionDelay** is in seconds (60, by default), whereas **collectionPeriod** is in minutes (1, by default).
- **collectionInterval**: How often, in minutes, the VPN application collects statistics from the managed servers in the cluster. The default is 5 minutes. The minimum value is 1. Collecting VPN statistics from managed servers complements the collection process performed by the underlying Service Broker framework by aggregating VPN application statistics from multiple managed services.

The object name of the VPN-specific **statConfig** MBean is:

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.apps.vpn.vpnconfiguration,version=1.1.0.0,name0=vpnConfig,name1=statConfig

This **statConfig** MBean contains the **busyHourAverageCPSThreshold** attribute. This attribute specifies the CPS value that, if exceeded, causes the VPN application to generate an SNMP trap notification. See "Configuring CPS Threshold SNMP Traps" for more information.

Viewing Runtime Monitoring Data

You can view runtime monitoring data by using operations in runtime MBeans exposed by the managed server process.

The runtime values represent the statistics of that particular managed server. They do not represent cluster-wide statistics, which are stored in the VPN application statistics database.

You can use operations in **VPNRuntimeStatisticsMBean** to view runtime statistics for the VPN application.

The object name of the runtime statistics MBean is:

oracle:type=oracle.axia.apps.vpn.statistic.mangement.VPNRuntimeStatisticsMBean oracle.axia.apps.vpn.statistic.management.

Table 5–1 lists the operations that return runtime statistics.

Table 5–1 VPN Runtime Statistics MBean Operations

Operation	Description
retrieveTotalCallCount	Total number of incoming and outgoing calls attempted.
retrieveSuccessfulCallCounts	Number of successful calls established.
retrieveTotalOriginatingCallCounts	Total number of call attempts that were originated by VPN users.
retrieveSuccessfulOriginatingCallCounts	Number of successful calls originated by VPN users.
retrieveTotalTerminatingCallCounts	Total number of incoming calls attempted to VPN users.
retrieveSuccessfulTerminatingCallCounts	Number of successful calls made to VPN users.
retrieveTotalIntraVPNCallCounts	Total number of call attempts between users of VPN subscriber organizations.
retrieveSuccessfulIntraVPNCallCounts	Number of successful calls between users of VPN subscriber organizations.
retrieveTotalPartnerVPNCallCounts	Total call attempts from VPN users to partner VPN users.

Generating the Runtime Audit Log

The audit log provides a historical view of the CPS rate for the cluster. It shows the CPS rate for each 24-hour period in the time span you specify.

You can access the audit log using the following runtime MBean:

oracle:type=oracle.axia.apps.vpn.statistic.mangement.VPNCPSAuditLogMBean

To view the log, connect to the managed server process and invoke the **genCPSAuditLog** operation in the **VPNCPSAuditLogMBean** MBean.

As parameters to the operation, pass the name for the audit log file and the time range of the records to report. The operation creates the audit log file with the name you specify in this location:

*oracle_home***/ocsb60/managed_server/***filename*

To specify the time range, indicate the start and end date of the range using the format "*yyyy-mm-dd*", for example, 2011-05-01.

The following example illustrates the format of the audit log file:

```
<calls_limit_log>
  <start>2011-04-01<start>
  <end>2011-06-01</end>
  <log_entry group="VPN" start="2011-04-23 16:00" end="2011-04-23 17:00"
                                  cps="0.05" limit="-1" exceeded="false"/>
  <log_entry group="VPN" start="2011-04-24 16:00" end="2011-04-24 17:00"
                                  cps="0.07" limit="-1" exceeded="false"/>
  <checksum>1a909c2b3420850e1e16c6b51f32dfcb</checksum>
</calls_limit_log>
```

In the log, each audit log item is represented by a **log_entry** element. The element represents the audit information for a given time period. It contains the following attributes:

- **group**: Indicates the Service Broker application for which this log entry was created, either VPN or Social Voice Communicator (SVC).
- **start**: The start date and hour of the time span represented by the log entry.
- **end**: The end date and hour of the time span represented by the log entry.
- **cps**: The busy hour calls per second for the time period.
- **limit**: Not currently used. Always set to **-1**.
- **exceeded**: Not currently used. Always set to **false**.

Resetting Statistics

You can use the **reset** operation to clear the current the statistics count for a managed server. This resets the **StatsBeginTime** value to the current time.

The operation is in the following runtime MBean:

oracle:type=oracle.axia.apps.vpn.statistic.mangement.VPNRuntimeStatisticsMBean

Configuring CPS Threshold SNMP Traps

You can configure a CPS threshold that, if exceeded, causes the VPN application to generate an SNMP trap. The SNMP monitoring capabilities of the VPN application rely on the Service Broker application framework.

To use traps, you must enable the SNMP agent and configure the trap destination in the configuration MBean:

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.snmp, version=1.0.0.1,name0=SnmpConfig

For more information about setting up SNMP and JMX monitoring for Service Broker, see the discussion on monitoring Service Broker in *Oracle Communications Service Broker Administrator's Guide*.

By default, the CPS threshold is 100 calls per second. To modify the threshold, set the new threshold value in the **busyHourAverageCPSThreshold** attribute in the following MBean:

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.axia.apps.vpn.vpnconfiguration,version=1.0.0.SNAPSHOT,name0=vpnConfig,name1=statConfig,name2=busyHourAverageCPSThreshold

If triggered, the trap indicates when the log was created and the triggering CPS value. The trap OID for the CPS audit log trap is:

iso.org.dod.internet.pricate.enterprises.oracle.sdp.notificationGroup.defaultNotification genCPSAuditLog

6

Using Batch Operation Tools

This chapter describes how to use the Oracle Communications Service Broker VPN batch operation tools, command-line utilities for creating, modifying, and validating VPN application data.

About Batch Operation Tools

When you invoke a VPN Provisioning API operation to create or modify a data object in storage, you are acting upon a single data object in storage at a time. The batch operation tools, on the other hand, enable you to act upon multiple objects at once. Batch operations are especially useful when adding a subscribing organization, when many users typically must be added to the system at once.

To use the batch operation tools, you place the data to be loaded or validated in local data files. The data in the files should be in comma-separated value (CSV) format.

The batch operation tools are:

- Batch Loader: Loads data from a CSV file into storage after optionally validating the data.
- Silent Batch Loader: Loads data from multiple CSV files into system storage after optionally validating the data.
- Batch Validator: Validates data of a single type in a CSV file or in storage, optionally exporting the stored data to a CSV file.
- Silent Batch Validator: Validates data of multiple types in CSV files or in storage, optionally exporting the stored data to CSV files.

The batch operation tools work by issuing requests to the VPN Provisioning API. Therefore, the same usage requirements applicable to making individual operation calls to the API apply to using the batch operation tools as well. For instance, to create a VPN subscriber, you must first create the administrative user for that subscriber, whether you use the API or the batch operation tools.

You can run the batch operation tools from any system that can access the Service Broker managed server. The managed server must be running with its HTTP listening port enabled. In a clustered environment, you only run the batch command against one of the managed servers, because the server will propagate the changes to the shared, cluster-wide data storage.

Versions of the batch operation tools are available for Linux and Windows operating systems.

Securing Batch Operation Tool Access

The batch operation tools are subject to the same security settings that apply to other requests to the VPN Provisioning API. These requirements may include SSL security or the API's custom authentication mechanism.

The following sections describe tool configuration and usage requirements for secured domains.

Using Batch Operation Tools with SSL Security

If the Service Broker domain uses SSL security, you must configure the batch operation tools to work with the enabled SSL security features. These features may include Secure HTTP (HTTPS) and client certificate authentication.

> **Note:** See *Oracle Communications Service Broker System Administrator's Guide* for information about Service Broker domain security.

The security settings for the batch operation tools are located in the following files:

- Linux:

  batch_home/**bulkloader/common.sh**

- Windows:

  batch_home/**bulkloader/common.cmd**

Where *batch_home* is the directory where the batch operation tools are located. The default directory is:

Oracle_home/**ocsb60/admin_console/utils/bulkloader/**

Configuring Security for HTTPS Access

If HTTPS access is enabled for the Service Broker domain, you must configure the following settings in **common.sh**. The settings describe the trusted keystore on the client where the certificates of the managed servers are located. The batch operation tools use these settings to verify the host identity of the managed server.

To configure the settings, remove the comment character ("#") from the start of each line, and replace the default values with the values appropriate for your system. The settings are:

```
# BK_JAVA_OPTS="${BK_JAVA_OPTS} -Djavax.net.ssl.trustStore=clienttruststore"
# BK_JAVA_OPTS="${BK_JAVA_OPTS} -Djavax.net.ssl.trustStoreType=type
# BK_JAVA_OPTS="${BK_JAVA_OPTS} -Djavax.net.ssl.trustStorePassword=password
```

Where:

- *clienttruststore* is the name of the trusted keystore on the client computer.
- *type* is the type of the trusted keystore. Generally, this is **jks**.
- *password* is the trusted keystore's passphrase.

Configuring Security for Client Certificate Authentication

If the managed server authenticates clients based on SSL client certificates, you must specify where the client keystore is located on the client system. To do so, configure

the following settings in **common.sh**. The settings identify the certificate in the keystore that the batch operation tools submit to the managed server.

To configure the settings, remove the comment character ("#") from the start of each line, and replace the default values with the values appropriate for your system. The settings are:

```
# BK_JAVA_OPTS="${BK_JAVA_OPTS} -Djavax.net.ssl.keyStore=clientkeystore
# BK_JAVA_OPTS="${BK_JAVA_OPTS} -Djavax.net.ssl.keyStoreType=type
# BK_JAVA_OPTS="${BK_JAVA_OPTS} -Djavax.net.ssl.keyStorePassword=password
```

Where:

- *clientkeystore* is the name of the client certificate keystore on the client computer.
- *type* is the type of the client key, such as **pkcs12**.
- *password* is the client keystore's passphrase.

Using Batch Operation Tools with Authentication and Authorization

If authentication and authorization is enabled for the VPN Provisioning API, you must supply valid administrator user name and password credentials when invoking the batch operation tools.

The built-in **admin** user object includes an attribute that specifies whether the username and password are validated. This attribute is named **isSecurity** for the VPN admin user and **isForceSecurity** for the Social Voice Communicator (SVC) admin user.

The value of the attribute can be:

- **true**: The username and password are validated.
- **false**: The username and password are not validated. Note, however, that they still must be supplied in the command.

Location of the Batch Operation Tools

After installing the Service Broker, you can find the batch operation tools and resources in the following directory:

oracle_home/**ocsb60/admin_console/utils/bulkloader**

The directory contains the **.sh** and **.bat** versions of these tools:

- **batch_load**: The common Batch Loader tool.
- **silent_ons_load**: Silent Batch Loader tool for the SVC one number service.
- **silent_vm_load**: Silent Batch Loader tool for the SVC voice mail service.
- **silent_vpn_load**: Silent Batch Loader tool for the VPN service.
- **batch_check**: The common Batch Validator tool.
- **silent_vpn_check_stores**: Silent Batch Validator for the VPN service.
- **common**: Script invoked by the other tools. You should not attempt to run this script directly.

In addition to the tools listed, the directory contains these subdirectories:

- **lib**: Runtime files for the batch operation tools.
- **onsdata**: Example data files for one number service. By default, the Silent Batch Loader for one number services loads the data in this directory.

- **vmdata**: Example data files for voice mail service. By default, the Silent Batch Loader for the voice mail services loads the data in this directory.
- **vpndata**: Example data files for VPN service. By default, the Silent Batch Loader for VPN services loads the data in this directory.

You can use the tools from a remote system by copying the entire contents of the **bulkloader** directory to the remote system. The remote system must have network access to the managed server.

Working with Batch Operation Data

The batch operation tools load and validate data in local data files you specify. The **batch_load** tool takes a data file as a command argument. You can create your own data file to pass to the **batch_load** tool or use a sample file included in the Administration Console distribution as a starting point.

The silent batch loader tools operate on all data files contained in the following directory:

Oracle_home`/ocsb60/admin_console/utils/bulkloader/vpndata/`

By default, this directory contains sample data files that you can use as a starting point for the data you want to load or validate. As illustrated by the sample files, you must use a separate data file for each type of VPN object you want to load or validate: VPNUser, VPNSubscriber, and so on.

The default files are:

- **1.vpnadmin.csv**
- **2.sp.csv**
- **3.vpnsubscriber.csv**
- **4.workingtime.csv**
- **5.vpnuser.csv**
- **6.profile.csv**
- **7.vpngroup.csv**
- **8.person.csv**
- **9.virtualuser.csv**

To use the silent batch operation tools, either replace the contents of the sample CSV data files with the application data to be loaded or modify the file references in the Silent Batch Loader script to refer to files you have created. In general, Oracle recommends that you create a copy of the batch tools directory, and modify the contents of the sample files in the copied directory.

To modify the data, open and edit each file. If using a silent batch operation tool, be sure to remove the contents of any file that is not applicable to your update.

The format of the data in the file must match the format required for its object type. This format is illustrated in the first lines of each sample data file. The schema information shown in the first lines serves as documentation only, and does not affect how the batch operation tools process the data. It can be excluded from your own data files.

See "VPN Data File Format Reference" for reference information on the VPN data types. Also see "Provisioning VPN Services" for more information about the types of objects in the VPN data model.

Running the Batch Loader

Before you run the Batch Loader, you must create and populate the CSV file that contains the data to be loaded. See "Working with Batch Operation Data" for more information on specifying the data to be loaded.

The Batch Loader script is named **batch_load**. There is a version of the script for Linux and Windows operating systems. The Batch Loader syntax is:

```
batch_load.bat -n username -p password -url server_url -r replace_if_exists -t
data_type -d data_file
```

Where:

- *username* is the user name of a provider administrator.
- *password* is the password for the provider administrator.
- *server_url* is the IP address or host name and HTTP port number of the managed server.
- *replace_if_exist* is a flag which, if **true**, causes existing data store objects to be entirely replaced by the contents of the data file. If **false**, the contents of the data file supplement or modify the existing data store objects, preserving existing data.
- *data_type* is the type of data you are loading, such as vpnadmin for administrative users or vpnuser for VPN users. See "Working with Batch Operation Data" for more information.
- *data_file* is the location and name of the CSV file that contains the data to be loaded.

For example:

```
./batch_load.sh -n admin -p password -url http://localhost:9001 -r true -t
vpnsubscriber -d /home/utils/bulkloader/vpndata/1.vpnsubscriber.csv
```

This example loads data in a file named **1.vpnsubscriber.csv**. If you do not include a required parameter in the command line, such as a password, the tool prompts you for a value.

The Batch Loader writes its results to a text-formatted log file in the directory from which it is run. The log file is named **bulkloader.log**. You can use the file to troubleshoot the operation of the tool or identify data format errors.

Running the Silent Batch Loader

The Silent Batch Loader loads data from multiple data files into storage at a time. Its runtime parameters, including which CSV files to load, are specified within the Silent Batch Loader script.

A version of the Silent Batch Loader exists for each type of service: VPN service, one number service, and voice mail service. By default, the tools load the data from the default directory appropriate for the service. For example, the Silent Batch Loader for the VPN services loads the data files from the **vpndata** directory. Similarly, the Silent Batch Loader for voice mail services loads the data files from **vmdata**.

The Silent Batch Loader script for the VPN application is named **silent_vpn_load**.

After you have populated the CSV files with your application data (as described in "Working with Batch Operation Data"), use the Silent Batch Loader as follows:

1. Navigate to the batch operation tools directory and open the version of the **silent_vpn_load** file appropriate for your operating system, that is, the **.bat** version for Windows or **.sh** for Linux.
2. In the script, modify these settings:
   - **ADMINNAME**: The user name of a provider administrator.
   - **PASSWORD**: The password for the provider administrator.
   - **URL**: The IP address or host name and HTTP port number of the managed server. The default is **http://localhost:9001**.
   - **batch_path**: The directory path to the batch operation tools and resources.
   - **output_path**: The directory path to which the tools write error output.
   - **data_file_path**: The directory path where the data files to be loaded are located.
   - **replaceIfExist**: If **true**, existing data store objects are entirely replaced by the contents of the data file. If **false** (default), the contents of the data file supplement or modify the existing data store objects, preserving existing data.
   - **checkOnLoad**: If **true**, the tool validates the data before loading it into storage. The default is **false**. Note that the **isBulk** setting in the VPN **admin** user also specifies whether data should be validated. If the **isBulk** setting is **false**, meaning data should be validated, but **checkOnLoad** value is **false**, the **checkOnLoad** setting prevails, meaning, and data is not validated.

     Loading data without validating it can improve performance, especially when loading a large amount of data. However, data validation should be used in most cases.
3. If needed, modify the names of the files to be loaded. The file names appear in the numbered steps section at the end of the script file.
4. Run the version of the Silent Batch Loader script appropriate for your operating system. For example, on a Linux system, enter the following command:

   ```
   sh silent_vpn_load.sh
   ```

The script loads the data, and prints results to the screen. If successful, `Done` appears on the screen. Otherwise, error messages appear that describe errors encountered while validating or loading the data.

Running the Batch Validator

The Batch Validator tool can check the validity of the data in an input file or the data that is in storage. When using an input file, the data can be validated without loading it into storage. Validating data in storage allows you to discover data conflicts that have arisen from loading data when validation was disabled or from external actions, such as direct changes to the database.

The Batch Validator identifies broken dependencies between data objects, such as a reference from a group to a profile that does not exist.

When running the Batch Validator on stored data, you can have the tool export the data from the database to a file. This enables you to back up application data from storage to an external format that can be imported into another system.

The Batch Validator command syntax is:

```
batch_check.bat -n username -p password -url server_url {(-t data_type -d data_
file) | (-s store_name -e export_file)} -o output_file
```

Where:

- *username* is the user name of a provider administrator.
- *password* is the password for the provider administrator.
- *server_url* is the IP address or host name and HTTP port number of the managed server.
- *data_type* is the type of data to be validated in a file, such as **vpnadmin** for administrative users or **vpnuser** for VPN users. Use only if validating data in a file, in which case you must also supply the *data_file* parameter.
- *data_file* is the location and name of the CSV file that contains the data to be validated. See "Working with Batch Operation Data" for more information on how to make data available to the batch operation tools.
- *store_name* is the type of data in storage to be validated, such as vpnadmin for administrative users or vpnuser for VPN users. Use only if validating data in storage, in which case you may also supply the *export_file* parameter.
- *export_file* is the location and name of the file to which the tool writes data it exports from the store.
- *output_file* is the location and name of the file to which the tool writes error output.

For example:

```
./batch_check.sh -n admin -p password -url http://localhost:9001 -t vpnsubscriber
-d /oraHome_1/ocsb60/admin_console/utils/bulkloader/vpndatas/1.vpnsubscriber.csv
```

This example validates data in a file named **1.vpnsubscriber.csv**. If you do not provide a required value in the command line, such as a password for the administrator, you are prompted for a value.

The Batch Validator writes its results to a text-formatted log file named **bulkloader.log** in the directory from which it is run. You can use the file to help troubleshoot operation or data format errors.

Using the Silent Batch Validator

The Silent Batch Validator can validate data in multiple data files at a time or validate stored data of multiple types. Its runtime parameters are specified within the Silent Batch Validator script. By default, the Silent Batch Validator validates data in storage.

To validate application data by using the Silent Batch Validator:

1. In the batch operation tools directory, open the version of the **silent_vpn_check** file appropriate for your operating system: the **.bat** version for Windows or **.sh** for Linux.
2. In the script, modify these settings:
   - **ADMINNAME**: The user name of a provider administrator in your system.
   - **PASSWORD**: The password for the administrator.

- **URL**: The IP address or host name of the computer on which the managed server is running and port at which it is listening for API requests. The default is **http://localhost:9001**.
- **batch_path**: The directory path to the batch operation tools and resources.
- **output_path**: The directory path to which you want to write output files.

3. If necessary, modify the commands in the file. By default, the commands direct the tool to validate existing storage and export the stored data to a file. Alternatively, you can have the tool validate data in a CSV file, suppress data export, or use any other option indicated in the **batch_check** help listing.
4. Run the version of the tool appropriate for your operating system.

   For example, on a Linux system, enter the following command to validate VPN service data:

   ```
   sh silent_vpn_check.sh
   ```

Output from the script is written to the screen. In addition, the Silent Batch Validator writes detailed results to a file in the output directory named for the respective data type and suffixed with **_check_result**. The tool exports data to CSV files in the output directory that are named with **_data** as their suffix.

VPN Data File Format Reference

The following sections describe the data file format for each type of object in the VPN data model, such as VPNUsers and VPNSubscribers. The sections indicate the data type name and format for each object type, with example data for each. In this context, the data type name corresponds to the value of the data type parameter passed to the Batch Loader and Silent Batch Loader tools (passed with the **-t** switch).

See "Working with Batch Operation Data" for information on how to make the data accessible to the batch operation tools.

All parameters are required unless indicated otherwise. See the information for the corresponding create or modify operation in "VPN Provisioning API Reference" for detailed descriptions of the parameters of each data type.

Administrator User

Administrator user data represents VPN subscriber administrator accounts. Provider administrators cannot be created with the Batch Loader.

Data Type

vpnadmin

File Format

adminName,*vpnID*,*adminPassword*,*role*

Example

```
bob,oracle_vpn,password,4
```

Note

- *role* must be **4**. Provider (role 1) administrators cannot be added using the Batch Loader.
- See "Create Administrator" for parameter details.

VPN Service Provider

Service provider data represents the VPN service provider account. The Service provider is a pre-defined data object that contains global VPN settings. Because the system can have only one VPN service provider object, the CSV file cannot contain multiple data objects. It only contains values to be used to update the existing service provider object.

Data Type

sp

File Format

name,description,address,privateCallMarker,pbxPrefix,localCountryCode,localAreaCode, mtcSuppressionPrefix,callBarringBlackList,numberRange

Example

```
"Mobile Telco","Mobile telephone services operator","111 Elm St. 94114"
,7,,86,10,22,,"86139*;86138*"
```

Note

- *callBarringBlackList* and *numberRange* contain values separated by semicolon.
- The *name* and *localCountryCode* parameters are required. All other parameters are optional.
- See "Modify VPN Service Provider" for parameter details.

VPN Subscriber

VPN subscriber data represents organizations that subscribe to hosted VPN services.

Data Type

vpnsubscriber

File Format

vpnID,name,escapeCode,privateCallEnable,privateNumberCLI,maxExtensionNumberLength, forcedOnNet,partnerVPNSwitch,timeZone,worktimeID,exclusionList,partnerVPN, callBarringBlackList,callScreeningBlackList

Example

```
"vpn001","Oracle",,"TRUE","TRUE",4,"TRUE","TRUE",8,,,"vpn003",,
"vpn002","BEA",0,"TRUE","FALSE",4,"TRUE","TRUE",8,,,,"4434*;1555168210",
"vpn003","Sun",9,"TRUE","FALSE",4,"TRUE","TRUE",8,,,,"535*;536*;537*",
```

Note

- *exclusionList*, *partnerVPN*, *callBarringBlackList*, and *callScreeningBlackList* contain data separated by semicolon.
- The *escapeCode*, *worktimeID*, *exclusionList*, *partnerVPN*, *callBarringBlackList*, and *callScreeningBlackList* parameters are optional. All other parameters are required.
- See "Create VPN Subscriber" for parameter details.

VPN User

VPN user data represents a PBX-based or mobile on-net device.

Data Type

vpnuser

File Format

publicPhoneNumber,vpnID,phoneExtension,memberType,closedUserGroupID,homeProfileID, roamingProfileID,personID,privateCallEnable

Example

```
861395551001,"vpn001",1001,0,"bjEmp,"alicehomeProfile","aliceroamProfile",,"TRUE"
861395551002,"vpn001",1002,0,"bjEmp",,,,"FALSE"
861395551003,"vpn001",1003,0,"bjManager",,,,"FALSE"
861395551110,"vpn001",1110,0,,,,,"TRUE"
861395551005,"vpn001",1005,0,"bjEmp","johnhomeProfile","aliceroamProfile",,"TRUE"
"86105550111*","vpn001","111*",1,,,,,"FALSE"
```

Note

- The *closedUserGroupID*, *homeProfileID*, *roamingProfileID*, and *personID* parameters are optional. All other parameters are required.
- See "Create VPN User" for parameter details.

User Group

User group data represents a set of VPN users in a subscriber organization.

Data Type

vpngroup

File Format

groupID,vpnID,parentCugID,homeProfileID,roamingProfileID,description,privateCallEnable

Example

```
"hqgroup","vpn001",,"hqGroupProfile",,"Parent group for headquarter-based managers
and non-managers; branch managers","TRUE"
"hqEmp","vpn001","hqgroup","hqEmpProfile",,"non-manager group","TRUE"
"hqManager","vpn001","hqgroup","hqManagerProfile",,,"TRUE"
```

Note

- The *parentCugID*, *homeProfileID*, *roamingProfileID*, and *description* parameters are optional. All other parameters are required.
- See "Create User Group" for parameter details.

Person

Person data represents end users of the VPN.

Data Type

person

File Format

personID,vpnID,name

Example

```
"alice","vpn001","Alice Smith"
"bob","vpn001","Bob D. Green"
```

Note

See "Create Person" for parameter details.

Working Time

Working time data represents the work hours for a VPN subscriber organization or user group.

Data Type
workingtime

File Format
workingtimeID,vpnID,StartingHours,StartingMinutes,EndingHours,EndingMinutes

Example
```
"workingtime1","vpn001","0;9;9;9;9;9;0","0;0;0;0;0;0;0","0;18;18;18;18;18;0","0;0;0;0;0;0;0"
"workingtime2","vpn001","0;19;19;19;19;19;0","0;0;0;0;0;0;0","0;23;23;23;23;23;0","0;0;0;0;0;0;0"
```

Note
- *StartingHours*, *StartingMinutes*, *EndingHours*, and *EndingMinutes* contain values separated by semicolon. Each value in the array represents the starting or ending time on a day of the week, beginning with Sunday.
- See "Create Working Time" for parameter details.

VPN Profile

VPN profile data contains a usage policy that can be applied to the VPN.

Data Type
profile

File Format
profileID,vpnID,worktimeID,description,timeZone,callScreeningList,callBarringList,durationControlList

Example
```
"hqManagerProfile","vpn001",,"HQ manager's profile",8,,,"99999;99999;3600;60"
"aliceProfile","vpn001","workingtime1","Alice's profile when in home network",8,"|5||false","|9|1|true","99999;99999;99999;99999"
"aliceRoamProfile","vpn001","workingtime1","Alice's profile when roaming",8,,"|9||false","99999;99999;99999;99999"
```

Note
- *durationControlList* contains values separated by semicolon.
- *callBarringList* and *callscreeningList* contain access control items, which have the following format:

  "*range*|*type*|*timespan*|*permit*;*range*|*type*|*timespan*|*permit*"

  As shown, vertical bar characters (|) separate individual fields in the access control rule, and semicolons separate multiple access control rules.
- The *worktimeID*, *description*, *callScreeningList*, and *callBarringList* parameters are optional. All other parameters are required.
- See "Create Profile" for parameter details.

Virtual User

Virtual user data represents an entity that is external to the organization but is assigned an extension on the VPN.

Data Type

virtualuser

File Format

publicPhoneNumber,vpnID,name,phoneExtension

Example

```
861015551999,vpn001,virtualuser01,1999
861015559999,vpn001,virtualuser02,9999
862125552999,vpn002,virtualuser03,2999
```

Note

- The *phoneExtension* number allocated by the Batch Loader is subject to the restrictions defined for the subscriber, such as maximum length and allocation exclusion lists.
- See "Create Virtual User" for parameter details.

7

Using the VPN Provisioning API

This chapter provides an overview of the VPN Provisioning Application Programming Interface (API), a REST-style API for configuring the services of the Oracle Communications Service Broker VPN application.

About the VPN Provisioning API

The VPN application exposes a RESTful API that clients use to access, create, and modify application data for Service Broker VPN.

The API provides operations for adding subscribers, configuring access rules, and adding users. In general, any configuration or provisioning task related to how the VPN application delivers services is performed with the VPN Provisioning API.

For most deployments, the primary clients for the API will be VPN self-management web interface. Administrators in the subscribing organization use the interface to perform the day-to-day provisioning and configuration tasks for that organization's VPN service instance.

The interface may be a dedicated web application that you create or GUI controls that you add to an existing customer portal.

The batch operation tools are also clients of the VPN Provisioning API. You can use the batch operation tools to populate application data for VPN services in bulk, based on contents of a CSV file. See "Using Batch Operation Tools" for information about the batch operation tools.

Client Application Design

The specific functionality you choose to expose in the VPN self-management application depends on your specific requirements and those of your subscribers. Use cases for the subscriber application could include:

- View and modify the general subscriber settings, including numbering plan attributes.
- Enable or disable particular features, such as private calling.
- Configure general roaming and home network policies for users and groups.
- Manage access control policies for the subscriber, such as duration controls.
- Add, modify, or remove closed user groups.
- Manage on-net mobile user devices.
- Manage on-net PBX users.

- Register end users.
- Configure VPN partner organizations.

You may also choose to create an application that provider administrators use to perform global management operations. Possible use cases for this application would be:

- Configure the provider object attributes, such as the name, description, and address of the provider organization.
- Configure global policies, such as call barring blacklists, and numbering properties.
- Create, modify, and remove subscriber administrators.
- Create and provide the initial configuration for the subscriber instances.

About REST

The VPN Provisioning API follows the style of a REST (Representational State Transfer) interface. Like any RESTful API, the VPN Provisioning API makes its services available to client applications through simple HTTP requests.

In a RESTful API, functions are distinguished by the combination of a particular URI and the HTTP method used to access it. In general, the URI identifies the resource on which to act, and the HTTP method identifies the type of action to perform.

The resource can be a generic resource or an instance-specific resource. A generic resource is equivalent to a factory creation function in a traditional API. The client application would use it, for example, to create a VPN instance or a new user in the VPN. After the resource is created, it can be modified or inspected at its unique URI.

For example, to create an object that represents a user in the VPN, the client application would send the following request to the API:

`POST /vpn/subscriber/`*`vpn_id`*`/users`

After the user object is created, the client application can inspect, modify, or delete the object by sending a GET, PUT, or DELETE request to the address associated with that object instance. For a user with the user name of alice, for example, the URI to access the user's object would be:

`vpn/subscriber/`*`vpn_id`*`/user/alice`

The methods in the HTTP protocol used in the VPN Provisioning API (POST, GET, PUT, and DELETE) correspond to the programming operations commonly known as CRUD operations. CRUD, which stands for create, read, update, and delete, represent the common operations applicable in data-oriented APIs. The equivalent function calls in a traditional API may be similar to `createUser()`, `getUser()`, `setUser()`, and `deleteUser()`. In this case, the instance on which the function operates is typically identified through an input parameter.

Accessing the VPN Provisioning API

In a Service Broker deployment, the VPN Provisioning API is exposed by the managed server. Therefore, at least one managed server must be deployed and running in your environment before you can use the API to configure and provision VPN services.

Several types of clients connect to the managed server instance, including the Administration Console client process and JMX clients. The managed server listens for

VPN Provisioning API at the port configured for client HTTP access. Before using the API, ensure that the managed server is configured to accept web requests. See "Opening an HTTP Listening Port for the VPN Provisioning Service" for information about configuring the managed server.

To test basic client connectivity to the API, you can try sending a GET request for one of several built-in resources in the system. For instance, you can send a GET request for the built-in **admin** administrator account, which is available at the following URI:

https://*host***:***port***/vpn/admin/admin**

Replace *host* with the host name or IP address you specified for the HTTP listener in the Web Services SSU of the Signaling Tier domain, and replace *port* with the port number.

In a managed server cluster, all servers access the same data store. Therefore, it is not important which managed server the API client addresses.

If the request is successful, the properties associated with the **admin** user appear in the body of the response message.

Request URI Format

Clients access the services of the VPN Provisioning API by sending HTTP requests to the managed server instance in the Service Broker managed domain. To use the API, the managed server must be configured and running.

The format of the URL to invoke operations in the Provisioning API is:

https://*host***:***port***/vpn/***component***/***resourceId*

Where:

- *host*:*port* are the host name and port number of an HTTP listener you have configured in the Web Service SSU of the Service Broker Signaling Tier.
- *component* is the data system component that you want to modify, create or query.
- *resourceId* is the identifier of the instance on which the application operates.

JSON Body Parameters

VPN Provisioning API operations that are invoked with the GET or DELETE HTTP methods do not require input values other than what is provided in the URI and headers of the client request. That is, they do not require HTTP body content to be supplied in the invocation request.

However, VPN Provisioning API operations invoked with the POST and PUT methods require additional input data. The Provisioning API takes input parameters in the form of JSON (JavaScript Object Notation) data in the body of the request.

JSON is a data exchange format based on JavaScript that is commonly used to pass information between web clients and servers over HTTP. In the body of the request, JSON data appears as one or more name-value pairs. For example:

```
{adminUsername:"alice",password:"password"}
```

The example shows two name-value pairs for an object in the system that represents a person. The type of both parameters are String.

JSON defines the following data types:

- number

- string
- boolean
- object

Parameters can be composed of an array. The **numberRange** attribute in the service provider object is an example. The individual values of a JSON array are enclosed in brackets and separated by commas, as in the following example:

```
{ "numberRange":["5982*","5981*"] }
```

In addition to simple types, a parameter can be a complex type. A complex type is a parameter that contains a nested set of subparameters. The **callScreeningList** parameter is an example. It contains an array of access control rules, as shown in the following example.

```
"callScreeningList": [
   {
      "range":"13522928011",
      "timeSpan":1,
      "permit":true,
      "type":1
   },

   {
      "range":"136*",
      "timeSpan":0,
      "permit":true,
      "type":0
   }
]
```

In addition to validating the basic data type of a submitted parameter, the VPN application may validate the content of the parameter. For example, it may ensure that a number falls within an expected range or that dates meet the required format.

The values of certain parameters are limited to what are called *word characters*. Expressed in regular expression notation, the word character set is: [A-Za-z0-9_]. In other words, a value may contain upper-case or lower-case letters, numbers, and underscore characters.

Numeric values are expressed as digits. In some cases, a parameter may contain a range of numbers, such as the extension value for a PBX user. To specify a range of numbers, use the common numeric prefix for the number range followed by a wildcard. An example is **898***, which applies to all numbers that begin with the numbers 898, such as 89823 or 8981.

"VPN Provisioning API Reference" lists the data format requirements for each parameter.

Creating and Updating Objects

To add users, devices, and subscribers to the VPN, you use the VPN Provisioning API to create the data objects that represent the entities you want to add.

The body of the POST request should contain the JSON-formatted data that describes the entity, such as the end user's name, phone number, extension, and so on.

Each type of object can have mandatory parameters and optional parameters. A mandatary parameter must be included in the create object request with a value that meets the content requirements for that parameter.

The value of a parameter may be a reference to another object. The VPN application checks object dependencies when it creates the object, ensuring that the objects upon which the new object depends, such as the profile or group definitions, exist.

When updating the object, you supply a new value for a modified setting in the body of a PUT request. In the body, you only supply the values for those parameters you want to modify, not for the entire object.

For example, to update a Subscriber object that currently has the following attribute values:

```
{
  "callScreeningBlackList" : [ ],
  "exclusionList" : [ ],
  "maxExtensionNumberLength" : 4,
  "callBarringBlackList" : [ "2894*", "499*" ],
  "escapeCode" : "0",
  "timeZone" : 8,
  "vpnID" : "vpn002",
  "privateNumberCLI" : false,
  "forcedOnNet" : true,
  "partnerVPNSwitch" : true,
  "name" : "company002",
  "partnerVPN" : [ ],
  "privateCallEnable" : true,
  "worktimeID" : ""
}
```

You would send an HTTP PUT request that contains only the new value for the field you want to change. Therefore, to add an item to the call barring black list and disable private calling, you would send a PUT request with the following body:

```
{"callBarringBlackList":["2894*", "499*", "498*"],"privateCallEnable" : false}
```

The operation would result in a Subscriber object of:

```
{
  "callScreeningBlackList" : [ ],
  "exclusionList" : [ ],
  "maxExtensionNumberLength" : 4,
  "callBarringBlackList" : [ "2894*", "499*", "498*" ],
  "escapeCode" : "0",
  "timeZone" : 8,
  "vpnID" : "vpn002",
  "privateNumberCLI" : false,
  "forcedOnNet" : true,
  "partnerVPNSwitch" : true,
  "name" : "company002",
  "partnerVPN" : [ ],
  "privateCallEnable" : false,
  "worktimeID" : ""
}
```

Authentication and Authorization

By default, security is disabled for the VPN Provisioning API. That is, clients can submit requests to the API without providing user credentials. While this may be suitable for test and evaluation systems, for production-level systems, access to the VPN Provisioning API should be secured.

The **isSecurity** attribute of the built-in **admin** user controls security setting for the API. If **isSecurity** is **true**, the API authenticates user credentials supplied in the request.

If the credentials are valid, the API further ensures that the user is authorized to access the requested resource, as indicated by the role of the user. There are two types of administrative users:

- **Provider administrators** have full access to the functions of the API. These users can modify data objects associated with any VPN subscriber.
- **Subscriber administrators** can only access subscriber-specific operations, and only in the context of the VPN with which they are associated.

The VPN application includes a built-in provider administrator account named **admin**. You can use this account to create additional provider and subscriber administrator accounts.

> **Note:** The initial password for the built-in **admin** user is set during initial configuration of the database used to store VPN application data. See "Configuring Data Persistence" for more information.

If security is enabled, client requests to the API must include the credentials of an administrator. The credentials should be supplied in the form of HTTP headers named **name** and **password**.

For example, a GET request submitted with user credentials would appear as follows:

```
GET /vpn/subscriber/vpn001/virtualusers HTTP/1.1
name: admin001
password: password
Host: 10.148.126.226
Accept: */*
```

A request that contains valid credentials but for a resource for which the administrator does not have privileges returns a **403 Forbidden** error, as in the following example:

```
HTTP/1.1 403 Forbidden
Content-Length: 0
Server: Jetty(8.0.1.0)
```

Note that the headers appear in HTTP requests in clear text. This may be acceptable in a test or evaluation system. However, in a production system, the connection should use Secure HTTP (HTTPS).

See "Opening an HTTP Listening Port for the VPN Provisioning Service" for information about using HTTPS to secure connections to the VPN Provisioning API.

Errors and Exceptions

When an operation succeeds, the VPN Provisioning API returns a response with HTTP status code 200.

An operation can fail for various reasons. For instance, an object you are creating may reference an object that does not yet exist; the JSON body of a request may be malformed; or field data may not match the expected form.

In general, if a request to the API fails, the VPN application returns an HTTP response with an error code that reflects the nature of the error. The error codes include:

- 400: The request body value is not correct. The response body contains detailed information about the error.
- 401: The user name or password is not correct.
- 403: The administrator does not have permission to access this object. (Such as an administrator for one VPN subscriber attempting to access settings for another.)
- 404: The URL style is unsupported or the resource does not exist in the system.
- 405: The method is not allowed.
- 406: The request method is not supported for this URL.
- 500: An internal system error occurred.
- 503: The service is unavailable.

Using Access Control Rules

An access control rule (**AccessControlItem**) is a common object in the VPN data model. It can be used by multiple types of other objects in the VPN data model.

An access control rule defines a particular type of restriction on the use of the VPN service. In general, you use it to match a telephone number to a time-based or call-type restriction. For example, an access control rule can specify call blocking to a number at a particular time or for a particular call type.

Each access control rule consists of these parameters:

- **type**
- **range**
- **timeSpan**
- **permit**

The **type** indicates the type of call to which the rule applies. Its value is an integer from 0 through 9, as follows:

- **0**: A telephone number range, which is specified in the **range** value by using a wildcard (*) character
- **1**: A single telephone number
- **2**: Inside group
- **3**: Outside group
- **4**: Inside VPN
- **5**: Outside VPN
- **6**: Inside service provider
- **7**: Outside service provider
- **8**: National
- **9**: International

The **range** value applies to type **0** or type **1** rules. It indicates the phone number to which the rule applies. For type **0** rules (a range), the range can include wildcard characters (*).

For type **1** rules (single), the range should indicate the specific phone number to which the rule applies. For the rule to apply, the specified number must fully match the actual number.

For other types of rules, the **range** value can be omitted.

The **timeSpan** parameter specifies the time of day that the rule applies. It takes an integer value, from 0 through 3, as follows:

- **0**: All times
- **1**: Within working hours for the subscriber
- **2**: Outside working hours for the subscriber

The VPN application determines whether a particular time is within working hours based on the working time specified in the worktimeID parameter for the subscriber.

The **permit** value determines the effect of a rule match on the call. If **permit** is **true**, the call is allowed, if **false**, the call is blocked.

Access Control Item Examples

The following access control item permits access for the phone number "13522928011" during working hours:

```
{
   "range":"13522928011",
   "timeSpan":1,
   "permit":true,
   "type":1
}
```

The following example permits calls for numbers starting with 136 at any time:

```
{
   "range":"136",
   "timeSpan":0,
   "permit":true,
   "type":0
}
```

The following example forbids international calls outside of working hours:

```
{
   "range":"",
   "timeSpan":2,
   "permit":false,
   "type":9
}
```

In the context of an example profile, the access control items are defined as follows:

```
{
   "description":"home profile",
   "worktimeID":"busy",
   "timeZone":8,
   "callScreeningList":[
   {
      "range":"13522928011",
```

```
      "timeSpan":1,
      "permit":true,
      "type":1
   },
   {
      "range":"136*",
      "timeSpan":0,
      "permit":true,
      "type":0
   }],
   "callBarringList":[
   {
      "range":"",
      "timeSpan":2,
      "permit":false,
      "type":9
   }],
   "durationControlList":[99999;99999;99999;99999]
}
```

8

VPN Provisioning API Reference

This chapter describes the VPN Provisioning API, a RESTful application programming interface for configuring and administering an Oracle Communications Service Broker VPN implementation.

For each operation in the API, this chapter lists the parameters accepted and returned by the operation and provides examples of HTTP requests and responses.

All JSON body parameters for POST (create) operations are required unless indicated as optional in the parameter description. For PUT (modify) operations, the request only needs to contain the attributes to be changed.

Note: In the request and response message examples in this chapter, line breaks and spaces have been added to the JSON data in the body of the message to improve readability.

Create Administrator

Creates an administrative user account in the system. The administrative user uses the VPN Provisioning API to configure VPN services.

There are two types of administrative users:

- Provider administrators (role 1) have privileges to invoke all API operations, including operations applicable to VPN instances. This role is intended for administrative users in the provider organization.
- Subscriber administrators (role 4) have privileges to invoke operations only within the context of the VPN subscriber organization to which they belong. This role is usually occupied by a person, group, or application within the subscribing organization.

For each operation in the API, the role of the administrator permitted to invoke the operation is indicated in the User Authorization sections.

If security is enabled for the API, invoking an API operation requires administrative user credentials in the form of basic authentication credentials.

User Authorization

Provider administrator

HTTP Method

POST

URI

/vpn/admins

Request Body

Request body parameters are:

- **adminName**: (string) The user name for the administrative user account. Must be 64 characters or less and start with a letter followed by any combination of word characters and spaces.
- **adminPassword**: (string) The password for the administrator account. Must be 64 characters or less.
- **role**: (integer) The role of the administrator. The role determines the authorization privileges for the user. Possible values are:
  - **1**: Administrative user with system-wide privileges, including the ability to modify provider settings and create additional administrative users.
  - **4**: Administrative user for a particular VPN subscriber instance identified by the **vpnID** parameter.
- **vpnID**: (string) For a role 4 administrator, the identifier of the VPN subscriber instance for which the user is an administrator.

  This parameter is mandatory for both types of administrators. However, note that a role 1 administrator is not associated with a specific VPN. For administrators with this role, you must include the **vpnID** parameter with a hyphen character as its value (that is, **"vpnID":"-"**).

Response Body

Empty

Examples

Example 8–1 Request: Create Subscriber Administrator

```
POST /vpn/admins HTTP/1.1
Host: 10.148.126.81
Accept: */*
Content-Length:83

{"adminName":"oracle_admin",
"adminPassword":"password",
"role":4,
"vpnID":"oracle"}
```

Example 8–2 Request: Create Provider Administrator

```
POST /vpn/admins HTTP/1.1
Host: 10.148.126.81
Accept: */*
Content-Length:103

{"adminName":"admin006",
"adminPassword":"password",
"role":1,
"vpnID":"-"}
```

Example 8–3 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 140
Server: Jetty(8.0.1.0)

Add the adminUser successfully. Possible effective fields include:
1.adminName; 2.adminPassword; 3.role; 4.vpnID.  Other fields are
ignored.
```

Get All Administrators

Returns a list of administrators in the system. Each administrator is identified by user name and URI.

User Authorization

Provider administrator

HTTP Method

GET

URI

/vpn/admins

Request Body

Empty

Response Body

The response body contains:

- A list of administrators, where each administrator is represented as:

  "*adminName*":"*URI*"

  For example, an administrator named **admin001** would be identified as:

  "admin001":"http://localhost:9001/vpn/admin/admin001"

Examples

Example 8–4 Request

```
GET /vpn/admins HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–5 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 227
Server: Jetty(8.0.1.0)

[{"admin003":"http://10.148.126.22:80/vpn/admin/admin003"},
{"admin":"http://10.148.126.22:80/vpn/admin/admin"},
{"admin001":"http://10.148.126.22:80/vpn/admin/admin001"},
{"admin002":"http://10.148.126.22:80/vpn/admin/admin002"}]
```

Get Administrator

Returns information about a specific provider or subscriber administrator.

User Authorization

Provider administrator or subscriber administrator

HTTP Method

GET

URI

/vpn/admin/*admin_name*

Request Body

Empty

Response Body

Response body parameters are:

- **adminName**: (string) The user name of the administrator.
- **role**: (integer) The role of the administrator. The role determines the authorization privileges for the administrator. Possible values are:
  - **1**: Administrative user with system-wide privileges, including the ability to modify provider settings and create additional administrative users.
  - **4**: Administrative user for a particular VPN subscriber instance identified by the **vpnID** parameter.
- **vpnID**: (string) For role 4 users, the identifier of the VPN subscriber instance associated with the administrator. For role 1 users, this parameter is not included in the response.
- **isSecurity**: (boolean) An attribute of the built-in **admin** user. This switch indicates whether the API requires user credentials in requests. This setting is not available for user-created provider or subscriber administrators.
- **isBulk**: (boolean) An attribute of the built-in **admin** user. This switch indicates whether application data loaded through the batch operation tools or the API is validated prior to loading. If **true**, data is not validated. This setting is not available for user-created provider or subscriber administrators

Examples

Example 8–6 Request

```
GET /vpn/admin/prov_admin HTTP/1.1
Host: 10.148.126.81
Accept: */*
```

Example 8–7 Response: Provider Administrator

```
HTTP/1.1 200 OK
Content-Type: application/text
```

```
Content-Length: 64
Server: Jetty(8.0.1.0)

{"adminName":"prov_admin",
"role":1,
"vpnID":"-"}
```

Example 8–8 Response: Subscriber Administrator

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 50
Server: Jetty(8.0.1.0)

{"adminName":"admin001",
"role":4,
"vpnID":"vpn001"}
```

Modify Administrator

Changes settings for an administrative user.

For a provider administrator, you use this operation to change password, enable bulk operations in the API, or enable authentication for API access.

For a subscriber administrator, you use this operation to modify a password only. The user name, role, and VPN ID values for administrators cannot be modified. To change these values, you must delete the administrator and create a new one with the desired user name, role, or VPN ID.

Authorization

Provider administrator

HTTP Method

PUT

URI

/vpn/admin/*admin_name*

Request Body

Request body parameters are:

- **adminPassword**: (string) Password associated with the administrative user account. Must be 64 characters or less.
- **isSecurity**: (boolean) An attribute of the built-in **admin** user only. Indicates whether the API requires user credentials in requests. If **true**, requests to the API must include valid provider administrator credentials in Basic Auth format.

  This setting is not available for user-created provider or subscriber administrators.
- **isBulk**: (boolean) An attribute of the built-in **admin** user only. Indicates whether application data loaded through the batch operation tools or the API is validated prior to loading. If **true**, data is not validated.

  This setting is not available for user-created provider or subscriber administrators

Response Body

Empty

Examples

Example 8–9 Request

```
PUT /vpn/admin/sysadmin HTTP/1.1
Host: 10.148.126.36
Accept: */*

{"adminPassword":"password"}
```

Example 8–10 Response

```
HTTP/1.1 200 OK
```

```
Content-Type: application/text
Content-Length: 113
Server: Jetty(8.0.1.0)

Modify the adminUser successfully. Possible effective fields include:
1.adminPassword.  Other fields are ignored.
```

Delete Administrator

Removes an administrative user account from the system, disabling access to the VPN Provisioning API for the account user.

You cannot remove a subscriber administrator if it is the only administrator for the VPN subscriber. You must first delete the subscriber instance before you are permitted to remove the administrator for that subscriber.

The built-in **admin** administrator account cannot be deleted.

Authorization

Subscriber administrator

HTTP Method

DELETE

URI

/vpn/admin/*username*

Request Body

Empty

Response Body

Empty

Examples

Example 8–11 Request

```
DELETE /vpn/admin/admin004 HTTP/1.1
Host: 10.148.126.36
Accept: */*
```

Example 8–12 Response

```
HTTP/1.1 200 OK
Content-Type: application/text; charset=utf-8
Content-Length: 42

Successfully delete an AdminUser[admin004]
```

Get VPN Service Provider

Returns information about the service provider organization, including general properties for the provider and global VPN settings, such as the global call barring list.

There can be only one VPN service provider object in the system. It is created automatically and cannot be deleted.

Privileges

Provider administrator or subscriber administrator

HTTP Method

GET

URI

vpn/sp

Request Body

Empty

Response Body

Response body parameters are:

- **name**: (string) The name of the service provider organization.
- **address**: (string) The address of the service provider.
- **description**: (string) A description of the service provider organization.
- **privateCallMarker**: (string) A dialed number prefix that identifies the call as a private call. Private calls are identified in the log and circumvent calling restrictions.
- **pbxPrefix**: (string) The numeric prefix associated with PBX-originated calls. The service switching point (SSP) uses this value to trigger VPN services.
- **localCountryCode**: (string) The local country dialing code.
- **localAreaCode**: (string) The local area dialing code.
- **mtcSuppressionPrefix**: (string) The mobile-terminated call (MTC) suppression prefix is a number that the VPN application prepends to the call destination number to indicate to the SSP that the originating and terminating services have been invoked in a single IN dialog.

  The purpose of the prefix is to indicate to the SSP that it should not invoke terminating services. The SSP should strip the prefix before further processing the call.
- **callBarringBlackList**: (string array) A comma-separated list of numbers that represent the call destinations to which calls from VPN users in this provider are blocked.
- **numberRange**: (string array) The public phone numbers of the service provider. The numbers may include wildcard characters to indicate a range (for example, 5982*, 5981*). The VPN application uses this value to determine whether called or

calling numbers are within the service provider's network for purposes of call restriction. That is, an access control item can define a restriction for calls conditional on whether the parties are inside or outside the service provider's network.

Examples

Example 8–13 Request

```
GET /vpn/sp HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–14 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 270
Server: Jetty(8.0.1.0)

{"callBarringBlackList":[],
"localAreaCode":"10",
"numberRange":["86139*","86138*"],
"address":"",
"localCountryCode":"86",
"description":"Mobile Telecommunication operations, worldwide",
"name":"Mobile Telco",
"pbxPrefix":"",
"mtcSuppressionPrefix":"22",
"privateCallMarker":"7"}
```

Modify VPN Service Provider

Changes settings for the service provider. The service provider object contains general properties for the provider organization along with global VPN settings, such as a global call barring list. The global settings apply to all VPN subscribers in the system.

There can be only one VPN service provider object in the system. It is created automatically and cannot be deleted.

Authorization

Provider administrator or subscriber administrator

HTTP Method

PUT

URI

vpn/sp

Request Body

Request body parameters are:

- **name**: (string) The name of the service provider organization. Must be 64 characters or less and start with a letter followed by any combination of word characters and spaces.
- **address**: (string) The address of the service provider organization. The length of this value is limited only by the size of the table field allocated for it in data storage, which is 512 bytes by default.
- **description**: (string) A description of the service provider organization. The length of this value is limited only by the size of the table field allocated for it in data storage, which is 256 bytes by default.
- **privateCallMarker**: (string) A dialed number prefix that identifies the call as a private call. Private calls are identified in the log and circumvent calling restrictions. Must be 2 digits or less.
- **pbxPrefix**: (string) The numeric prefix associated with PBX-originated calls. The service switching point (SSP) uses this value to trigger VPN services.
- **localCountryCode**: (string) The local country dialing code.
- **localAreaCode**: (string) The local area dialing code.
- **mtcSuppressionPrefix**: (string) The mobile-terminated call (MTC) suppression prefix is a number that the VPN application prepends to the call destination number to indicate to the SSP that the originating and terminating services have been invoked in a single IN dialog.

  The purpose of the prefix is to indicate to the SSP that it should not invoke terminating services. The SSP should strip the prefix before further processing the call.
- **callBarringBlackList**: (string array) A comma-separated list of numbers that represent call destinations to which calls from VPN users in this provider are blocked. You can specify the barred caller as a phone number, wildcarded number prefix (1888*), or IP address.

- **numberRange**: (string array) Identifies the public phone numbers of the service provider. The numbers may include wildcard characters to indicate a range (for example, 5982*, 5981*). The VPN application uses this value to determine whether called or calling numbers are within the service provider's network for purposes of call restriction. That is, an access control item can define a restriction for calls conditional on whether the parties are inside or outside the service provider's network.

Response Body

Empty

Examples

Example 8–15 Request

```
PUT /vpn/sp HTTP/1.1
Host: 10.148.126.22
Accept: */*
Content-Length:144

{
"address":"500 Main St, Redwood Shores, CA",
"callBarringBlackList":["2894*","499*","18665555555"],
"numberRange":["86139*","86138*","86140*"]}
```

Example 8–16 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 257
Server: Jetty(8.0.1.0)

Modify the sp successfully. Possible effective fields include:
1.name; 2.address; 3.description; 4.privateCallMarker; 5.pbxPrefix;
6.localCountryCode; 7.localAreaCode; 8.mtcSuppressionPrefix;
9.callBarringBlackList; 10.numberRange. Other fields are ignored.
```

Create VPN Subscriber

Creates a subscriber object, which contains settings for a particular organization that subscribes to VPN services. You must create a subscriber object for each organization for which you are making VPN services available.

Before you can call this operation, the subscriber administrator account associated with the subscriber instance must exist.

Privileges

Provider administrator or subscriber administrator

HTTP Method

POST

URI

/vpn/subscribers

Request Body

Request body parameters are:

- **vpnID**: (string) A unique identifier for the VPN Subscriber object. Other objects in the data model, such as partner and administrator definitions, use this value to reference this subscriber instance. Must be 64 characters or less and start with a letter followed by any combination of word characters.
- **name**: (string) The name of the VPN subscriber organization. Must be 64 characters or less and start with a letter followed by any combination of word characters and spaces.
- **escapeCode**: (string) Optional. A numeric escape code that, when preceding a dialed number, causes the dialed number to circumvent number translation by the VPN module. Must be 2 digits or less.
- **privateCallEnable**: (boolean) Whether private calling is enabled for this subscriber. The default value is false.
- **privateNumberCLI**: (boolean) The calling line identification presentation mode for the VPN subscriber. If true, the private number is forwarded for presentation on the terminating device. If false, the public number is forwarded for presentation on the terminating device. The default value is false.
- **maxExtensionNumberLength**: (integer) The maximum number of digits that can be used for extensions that are allocated for devices for this VPN subscriber. The value must be from 2 to 10 digits.
- **forcedOnNet**: (boolean) If true, specifies that when an end user dials the public number of another on-net user instead of that user's extension, the call is treated as an on-net call. The default value is false.
- **partnerVPNSwitch**: (boolean) Whether the partner VPN feature is enabled for this subscriber. If true, the provider can apply differentiated charging rates to calls between members of this VPN and the VPN subscribers identified in the **partnerVPN** attribute. The default value is false.

- **timeZone**: (integer) The time zone for this subscriber. The time value is used for certain type of policy evaluation scenarios, such as calling restrictions based on time of day. Specify the time zone as the hour offset from UTC.

  The time zone setting can also be set at the profile level, allowing for a geographically distributed organization.
- **exclusionList**: (string array) Optional. Extension numbers or prefixes of numbers that cannot be allocated as private extensions for this subscriber. Specify a range of numbers using the wildcard symbol (*).
- **callBarringBlackList**: (string array) Optional. A comma-separated list of numbers that represent call destination to which outgoing calls from users in this subscriber are blocked. You can specify the barred caller as a phone number, wildcarded number prefix (1888*), or IP address.
- **callScreeningBlackList**: (string array) Optional. A comma-separated list of numbers that represent call sources from which incoming calls to users in this subscriber are blocked. You can specify the source as a phone number, wildcarded number prefix (1888*), or IP address.
- **partnerVPN**: (string array) Optional. A list of **vpnID** values that represent the partner organizations of this subscriber.
- **worktimeID**: (string) Optional. The identifier of the working time object that defines the working hours for this subscriber. The VPN application uses this value to evaluate rules based on working hours.

Response Body

Empty

Examples

Example 8–17 Request

```
POST /vpn/subscribers HTTP/1.1
Host: 10.148.126.22
Accept: */*
Content-Length:331

{"callScreeningBlackList":[],
"exclusionList":[],
"maxExtensionNumberLength":4,
"callBarringBlackList":["2894*","499*","18665555555"],
"escapeCode":"0",
"timeZone":8,
"vpnID":"vpn004",
"privateNumberCLI":false,
"forcedOnNet":true,
"partnerVPNSwitch":true,
"name":"company004",
"partnerVPN":["vpn002"],
"privateCallEnable":true,
"worktimeID":""}
```

Example 8–18 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 343
```

```
Server: Jetty(8.0.1.0)

Add the vpnsubscriber successfully. Possible effective fields include:
1.vpnID; 2.name; 3.escapeCode; 4.privateCallEnable; 5.privateNumberCLI;
6.maxExtensionNumberLength; 7.forcedOnNet; 8.partnerVPNSwitch; 9.timeZone;
10.exclusionList; 11.callBarringBlackList; 12.callScreeningBlackList;
13.partnerVPN; 14.worktimeID. Other fields are ignored.
```

Get All VPN Subscribers

Returns a list of subscribers in the system. Each subscriber in the list is represented by its vpnID name and URI.

Privileges

Provider administrator

HTTP Method

GET

URI

/vpn/subscribers/

Request Body

Empty

Response Body

The response body contains:

- A list of subscribers, where each subscriber is identified in the form:

  "*vpnID*":"*URI*"

  For example, a subscriber with an ID of **vpn001** would be identified as:

  "vpn001":"http://localhost:9001/vpn/subscriber/vpn001"

Examples

Example 8–19 Request

```
GET /vpn/subscribers HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–20 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 178
Server: Jetty(8.0.1.0)

[{"vpn001":"http://10.148.126.22:80/vpn/subscriber/vpn001"},
{"vpn003":"http://10.148.126.22:80/vpn/subscriber/vpn003"},
{"vpn002":"http://10.148.126.22:80/vpn/subscriber/vpn002"}
```

Get VPN Subscriber

Returns information on a particular subscriber.

Privileges

Provider administrator or subscriber administrator

HTTP Method

GET

URI

/vpn/subscriber/*vpn_id*

Request Body

Empty

Response Body

Response body parameters are:

- **name**: (string) The name of the VPN subscriber organization.
- **escapeCode**: (string) An escape code that, when preceding a dialed number, causes the dialed number to circumvent number translation by the VPN module.
- **privateCallEnable**: (boolean) Whether private calling is enabled for this subscriber.
- **privateNumberCLI**: (boolean) The calling line identification presentation mode for the VPN subscriber. If true, the private number is forwarded for presentation on the terminating device. If false, the public number is forwarded for presentation on the terminating device.
- **maxExtensionNumberLength**: (integer) The maximum number of digits that can be used for extensions that are allocated for devices for this VPN subscriber.
- **forcedOnNet**: (boolean) If true, specifies that when an end user dials the public number of another on-net user instead of that user's extension, the call is treated as an on-net call.
- **partnerVPNSwitch**: (boolean) Whether the partner VPN feature is enabled for this subscriber. If true, the provider can apply differentiated charging rates to calls between members of this VPN and the VPN subscribers identified in the **partnerVPN** attribute.
- **timeZone**: (integer) The time zone for this subscriber. The time value is used for certain type of policy evaluation scenarios, such as calling restrictions based on time of day. The time zone value is the hour offset from UTC.
- **exclusionList**: (string array) Extension numbers or prefixes of numbers that cannot be allocated as private extensions for this subscriber.
- **callBarringBlackList**: (string array) A comma-separated list of numbers that represent call destinations to which calls from VPN users in this provider are blocked.

- **callScreeningBlackList**: (string array) A comma-separated list of numbers that represent call sources from which incoming calls to users in this subscriber are blocked.
- **partnerVPN**: (string array) A list of **vpnID** values that represent the partner organizations of this subscriber.
- **worktimeID**: (string) The identifier of the working time object that defines the working hours for this subscriber. The VPN application uses this value to evaluate rules based on working hours.

Examples

Example 8–21 Request

```
GET /vpn/subscriber/vpn002 HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–22 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 323
Server: Jetty(8.0.1.0)

{"callScreeningBlackList":[],
"exclusionList":[],
"maxExtensionNumberLength":4,
"callBarringBlackList":["2894*","499*","18665555555"],
"escapeCode":"0",
"timeZone":8,
"vpnID":"vpn002",
"privateNumberCLI":false,
"forcedOnNet":true,
"partnerVPNSwitch":true,
"name":"company002",
"partnerVPN":[],
"privateCallEnable":true,
"worktimeID":""}
```

Modify VPN Subscriber

Changes settings for a subscriber.

Privileges

Provider administrator or subscriber administrator

HTTP Method

PUT

URI

/vpn/subscriber/*vpn_id*

Request Body

Request body parameters are:

- **name**: (string) The name of the VPN subscriber organization. Must be 64 characters or less and start with a letter followed by any combination of word characters and spaces.
- **escapeCode**: (string) An escape code that, when preceding a dialed number, causes the dialed number to circumvent number translation by the VPN module. Must be 2 digits or less.
- **privateCallEnable**: (boolean) Whether private calling is enabled for this subscriber. The default value is false.
- **privateNumberCLI**: (boolean) The calling line identification presentation mode for the VPN subscriber. If true, the private number is forwarded for presentation on the terminating device. If false, the public number is forwarded for presentation on the terminating device. The default value is false.
- **maxExtensionNumberLength**: (integer) The maximum number of digits that can be used for extensions that are allocated for devices for this VPN subscriber. The length can be from 2 to 10 digits.
- **forcedOnNet**: (boolean) If true, specifies that when an end user dials the public number of another on-net user instead of that user's extension, the call is treated as an on-net call. The default value is false.
- **partnerVPNSwitch**: (boolean) Whether the partner VPN feature is enabled for this subscriber. If true, the provider can apply differentiated charging rates to calls between members of this VPN and the VPN subscribers identified in the **partnerVPN** attribute. The default value is false.
- **timeZone**: (integer) The time zone for this subscriber. The time value is used for certain type of policy evaluation scenarios, such as calling restrictions based on time of day. You specify the time zone as an hour offset from UTC.
- **exclusionList**: (string array) Extension numbers or prefixes of numbers that cannot be allocated as private extensions for this subscriber. Specify a range of numbers using the wildcard symbol (*).
- **callBarringBlackList**: (string array) A comma-separated list of numbers that represent call destination to which outgoing calls from users in this subscriber are

blocked. You can specify the barred caller as a phone number, wildcarded number prefix (1888*), or IP address.

- **callScreeningBlackList**: (string array) A comma-separated list of numbers that represent call sources from which incoming calls to users in this subscriber are blocked. You can specify the source as a phone number, wildcarded number prefix (1888*), or IP address.
- **partnerVPN**: (string array) A list of **vpnID** values that represent the partner organizations of this subscriber.
- **worktimeID**: (string) The identifier of the working time object that defines the working hours for this subscriber. The VPN application uses this value to evaluate rules based on working hours.

Response Body

Empty

Examples

Example 8–23 Request

```
PUT /vpn/subscriber/vpn004 HTTP/1.1
Host: 10.148.126.22
Accept: */*
Content-Length:34

{"partnerVPN":["vpn002","vpn003"]}
```

Example 8–24 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 336
Server: Jetty(8.0.1.0)

Modify the vpnsubscriber successfully. Possible effective fields include:
1.name; 2.escapeCode; 3.privateCallEnable; 4.privateNumberCLI;
5.maxExtensionNumberLength; 6.forcedOnNet; 7.partnerVPNSwitch; 8.timeZone;
9.exclusionList; 10.callBarringBlackList; 11.callScreeningBlackList;
12.partnerVPN; 13.worktimeID. Other fields are ignored.
```

Delete VPN Subscriber

Removes a subscriber definition from the system. Removing the subscriber removes all VPN users, policies, groups, and other objects associated with the subscriber.

Privileges

Provider administrator or subscriber administrator

HTTP Method

DELETE

URI

/vpn/subscriber/*vpn_id*

Request Body

Empty

Response Body

Empty

Examples

Example 8–25 Request

```
DELETE /vpn/subscriber/vpn004 HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–26 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 42
Server: Jetty(8.0.1.0)

Successfully delete the subscriber[vpn004]
```

Create Virtual User

Creates a virtual user for a subscriber. A virtual user is a VPN user that is external to the subscribing organization. Calls to the virtual user are considered on-net, but calls from the virtual user to other types of VPN users are not.

Privileges

Provider administrator or subscriber administrator

HTTP Method

POST

URI

/vpn/subscriber/*vpn_id***/virtualusers**

Request Body

Request body parameters are:

- **name**: (string) An identifying name for the virtual user. Must be 64 characters or less and start with a letter followed by any combination of word characters and spaces.
- **phoneExtension**: (string) The private extension for the virtual user. This is the phone extension used from within the VPN to call the virtual user. Must meet the extension length requirement for the subscriber, and may contain digits or digits followed by an asterisk (*).
- **publicPhoneNumber**: (string) The public phone number of the virtual user. Must be 20 characters or less, and contain all digits or digits followed by an asterisk (*) or preceded with a plus character (+).

Response Body

Empty

Examples

Example 8–27 Request

```
POST /vpn/subscriber/vpn003/virtualusers HTTP/1.1
Host: 10.148.126.22
Accept: */*
Content-Length: 100

{
"phoneExtension":"4999",
"publicPhoneNumber":"861030004999",
"name":"travelagentcontact"
}
```

Example 8–28 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 141
```

```
Server: Jetty(8.0.1.0)

Add the virtualuser successfully. Possible effective fields include:
1.publicPhoneNumber; 2.name; 3.phoneExtension. Other fields are ignored.
```

Get All Virtual Users

Returns a list of virtual users for a subscriber. Each virtual user is identified by its public number and URL.

Privileges

Provider administrator or subscriber administrator

HTTP Method

GET

URI

vpn/subscriber/*vpn_id***/virtualusers**

Request Body

Empty

Response Body

The response body contains:

- A list of virtual users, where each virtual user is identified in the form:

  "*publicPhoneNumber*":"*URI*"

  For example, a virtual user with a public phone number of **861030009999** would be identified as:

  ```
  "86103009999":
  "http://host:9001/vpn/subscriber/vpn3/virtualuser/86103009999"
  ```

Examples

Example 8–29 Request

```
GET /vpn/subscriber/vpn003/virtualusers HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–30 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 181
Server: Jetty(8.0.1.0)

[{"861030009999":
   "http://10.148.126.22:80/vpn/subscriber/vpn003/virtualuser/861030009999"},
{"861030003999":
   "http://10.148.126.22:80/vpn/subscriber/vpn003/virtualuser/861030003999"}]
```

Get Virtual User

Returns information on a specific virtual user.

Privileges

Provider administrator or subscriber administrator

HTTP Method

GET

URI

/vpn/subscriber/*vpn_id***/virtualuser/***public_phone_number*

Request Body

Empty

Response Body

Request body parameters are:

- **name**: (string) An identifying name for the virtual user.
- **phoneExtension**: (string) The private extension for the virtual user.
- **publicPhoneNumber**: (string) The public phone number of the virtual user.

Examples

Example 8–31 Request

```
GET /vpn/subscriber/vpn003/virtualuser/861030009999 HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–32 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 100
Server: Jetty(8.0.1.0)

{"phoneExtension":"9999",
"publicPhoneNumber":"861030009999",
"name":"travelagentcontact",
"vpnID":"vpn003"}
```

Modify Virtual User

Changes settings for a virtual user, including the virtual user's name or phone extension. To change the public number for a user, you must remove the existing virtual user object and create a new one with the desired number.

Privileges

Provider administrator or subscriber administrator

HTTP Method

PUT

URI

/vpn/subscriber/*vpn_id***/virtualuser/***publicPhoneNumber*

Request Body

Request body parameters are:

- **name**: (string) An identifying name for the virtual user. Must be 64 characters or less and start with a letter followed by any combination of word characters and spaces.
- **phoneExtension**: (string) The private extension for the virtual user. Must meet the extension length requirement for the subscriber, and may contain digits or digits followed by an asterisk (*).

Response Body

Empty

Examples

Example 8–33 Request

```
PUT /vpn/subscriber/vpn003/virtualuser/861030009999 HTTP/1.1
Host: 10.148.126.22
Accept: */*
Content-Length:25

{"phoneExtension":"8999"}
```

Example 8–34 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 123
Server: Jetty(8.0.1.0)

Modify the virtualuser successfully. Possible effective fields include:
1.name; 2.phoneExtension. Other fields are ignored.
```

Delete Virtual User

Removes a virtual user from the subscriber.

Privileges

Provider administrator or subscriber administrator

HTTP Method

DELETE

URI

/vpn/subscriber/*vpn_id***/virtualuser/***publicPhoneNumber*

Request Body

Empty

Response Body

Empty

Examples

Example 8–35 Request

```
DELETE /vpn/subscriber/vpn003/virtualuser/861030004999 HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–36 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 49
Server: Jetty(8.0.1.0)

Successfully delete the VirtualUser[861030004999]
```

Create VPN User

Creates a VPN user object, which represents an on-net mobile or PBX on-net device.

For a mobile user, the VPN user defines properties such as the user's group membership and usage policy. For a PBX user, you specify the public phone number, phone extension, and member type parameters for the user. The phone extension and public phone number usually take the form of a number range. The other VPN user parameters do not apply to a PBX VPN user.

Privileges

Provider administrator or subscriber administrator

HTTP Method

POST

URI

/vpn/subscriber/*vpn_id***/users**

Request Body

Request body parameters are:

- **publicPhoneNumber**: (string) The public phone number of the user device. For a PBX device, this can be a range, specified by a wildcard symbol. Must be 20 characters or less, and contain all digits or digits followed by an asterisk (*) or preceded with a plus character (+).
- **phoneExtension**: (string) The private extension for the user device.

  For a PBX device, this can be a range of extensions specified by a numeric prefix. The phone extension range should provide a one-to-one correspondence with the public phone number range. For example:

  "phoneExtension" : "11*"

  "publicPhoneNumber" : "17240011*"

  In this case, public phone number 1724001101 would map to extension 1101, 1724001102 to extension 1102, and so on.

  The parameter value must meet the extension length requirement for the subscriber, and may contain digits or digits followed by an asterisk (*).
- **memberType**: (integer) The user type, with these options:
  - **0**: Mobile on-net member
  - **1**: PBX on-net member or members.
- **closedUserGroupID**: (string) Optional. The identifier of an existing user group to which this mobile device should be assigned. This parameter does not apply to PBX users.
- **homeProfileID**: (string) Optional. The identifier of an existing policy profile that applies to this mobile device when operated within its home network. This parameter does not apply to PBX users.

- **roamingProfileID**: (string) Optional. The identifier of an existing policy profile that applies to this mobile device when operated outside of its home network. This parameter does not apply to PBX users.
- **personID**: (string) Optional. The identifier of the end user associated with this mobile device as defined in an existing person object. This parameter does not apply to PBX users.
- **privateCallEnable**: (boolean) Whether the mobile device can be used for private calls. This parameter does not apply to PBX users. The default value is false.

Response Body

Empty

Examples

Example 8–37 Request

```
POST /vpn/subscriber/vpn001/users HTTP/1.1
Host: 10.148.126.22
Accept: */*
Content-Length:205

{"phoneExtension":"1115",
"publicPhoneNumber":"861391001015",
"memberType":0,
"homeProfileID":"",
"closedUserGroupID":"bjManager",
"privateCallEnable":false,
"roamingProfileID":"",
"personID":""}
```

Example 8–38 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 235
Server: Jetty(8.0.1.0)

Add the vpnuser successfully. Possible effective fields include:
1.publicPhoneNumber; 2.phoneExtension; 3.memberType; 4.closedUserGroupID;
5.homeProfileID; 6.roamingProfileID; 7.personID; 8. privateCallEnable. Other
fields are ignored.
```

Get All VPN Users

Returns a list of users in the system. Each user is represented by the public number of the device and URI.

User Authorization

Provider administrator or subscriber administrator

HTTP Method

GET

URI

/vpn/subscriber/*vpn_id***/users**

Request Body

Empty

Response Body

The response body contains:

- A list of users, where each user is represented as:

  "*publicPhoneNumber*":"*URI*":

  For example, a user with a public number of **861391001001** would be identified as:

  "861391001001" :
  "http://localhost:9001/vpn/subscriber/vpn001/user/861391001001"

Examples

Example 8–39 Request

```
GET /vpn/subscriber/vpn001/users HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–40 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 576
Server: Jetty(8.0.1.0)

[{"861391001002":
  "http://10.148.126.22:80/vpn/subscriber/vpn001/user/861391001002"},
{"861391001001":
  "http://10.148.126.22:80/vpn/subscriber/vpn001/user/861391001001"},
{"861391001003":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/user/861391001003"},
{"861391001005":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/user/861391001005"},
{"861391001110":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/user/861391001110"},
{"86101000111*":
```

```
   "http://10.148.126.22:80/vpn/subscriber/vpn001/user/86101000111*"},
{"17240011*":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/user/17240011*"}]
```

Get VPN User

Returns information on a specific user by public phone number or, for a PBX user, by phone number range.

Privileges

Provider administrator or subscriber administrator

HTTP Method

GET

URI

/vpn/subscriber/*vpn_id***/user/***public_phone_number*

Request Body

Empty

Response Body

Response body parameters are:

- **publicPhoneNumber**: (string) The public phone number of the user device. For a PBX device, this can be a range, specified by a wildcard symbol.
- **phoneExtension**: (string) The private extension for the user device. For a PBX device, this can be a range of extensions specified by a numeric prefix. The range should provide a one-to-one correspondence with the public phone number range.
- **memberType**: (integer) The type of user device, with these options:
  - **0**: Mobile on-net member
  - **1**: PBX on-net member or members.
- **closedUserGroupID**: (string) The identifier of the user group to which this mobile device belongs. This parameter does not apply to type 1, PBX users.
- **homeProfileID**: (string) The identifier of the policy profile that applies to this mobile device when operated in its home network. This parameter does not apply to type 1, PBX users.
- **roamingProfileID**: (string) The identifier of the policy profile that applies to this mobile device when operated outside of its home network. This parameter does not apply to type 1, PBX users.
- **privateCallEnable**: (boolean) Whether the mobile device can be used for private calls. This parameter does not apply to type 1, PBX users.
- **vpnID**: (string) The identifier for the VPN instance to which the user device belongs.
- **personID**: (string) The name of the end user associated with this mobile device as defined by a **person** object. This parameter does not apply to type 1, PBX users.
- **privateCallEnable**: (boolean) Whether the device can be used for private calls.

Examples

Example 8–41 Request: Get Mobile User

```
GET /vpn/subscriber/vpn001/user/861391001003 HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–42 Response: Get Mobile User

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 205
Server: Jetty(8.0.1.0)

{"phoneExtension":"1003",
"publicPhoneNumber":"861391001003",
"memberType":0,
"homeProfileID":"",
"closedUserGroupID":"bjManager",
"vpnID":"vpn001",
"privateCallEnable":false,
"roamingProfileID":"",
"personID":""}
```

Example 8–43 Request: Get PBX User

```
GET /vpn/subscriber/vpn001/user/17240011* HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–44 Response: Get PBX User

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 192
Server: Jetty(8.0.1.0)

{"phoneExtension":"11*",
"publicPhoneNumber":"17240011*",
"memberType":1,
"homeProfileID":"",
"closedUserGroupID":"",
"vpnID":"vpn001",
"privateCallEnable":false,
"roamingProfileID":"",
"personID":""}
```

Modify VPN User

Changes settings for a user device.

Privileges

Provider administrator or subscriber administrator

HTTP Method

PUT

URI

/vpn/subscriber/*vpn_id***/user/***public_phone_number*

Request Body

Request body parameters are:

- **phoneExtension**: (string) The private extension for the user device. For a PBX device, this can be a range of extensions specified by a numeric prefix. The phone extension range should provide a one-to-one correspondence with the public phone number range. For example:

  "phoneExtension" : "11*"

  "publicPhoneNumber" : "17240011*"

  In this case, public phone number 1724001101 would be mapped to extension 1101, and public number 1724001102 to extension 1102, and so on.

  The parameter value must meet the extension length requirement for the subscriber, and may contain digits or digits followed by an asterisk (*).

- **memberType**: (integer) The user type, with these options:
  - **0**: Mobile on-net member
  - **1**: PBX on-net member or members.
- **closedUserGroupID**: (string) The identifier of an existing user group to which this mobile device should be assigned. This parameter does not apply to PBX users.
- **homeProfileID**: (string) The identifier of an existing policy profile that applies to this mobile device when operated in its home network. This parameter does not apply to PBX users.
- **roamingProfileID**: (string) The identifier of an existing policy profile that applies to this mobile device when operated outside of its home network. This parameter does not apply to PBX users.
- **personID**: (string) The identifier of the end user associated with this mobile device as defined in an existing person object. This parameter does not apply to PBX users.
- **privateCallEnable**: (boolean) Whether the mobile device can be used for private calls. This parameter is not relevant to type 1, PBX users. The default value is false.

Response Body

Empty

Examples

Example 8–45 Request

```
PUT /vpn/subscriber/vpn001/user/17240011* HTTP/1.1
Host: 10.148.126.22
Accept: */*
Content-Length:29

{"homeProfileID":"hqProfile"}
```

Delete VPN User

Removes a user device from the system.

Privileges

Provider administrator or subscriber administrator

HTTP Method

DELETE

URI

/vpn/subscriber/*vpn_id***/user/***public_phone_number*

Request Body

Empty

Response Body

Empty

Examples

Example 8–46 Request

```
DELETE /vpn/subscriber/vpn001/user/861391001001 HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–47 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 48
Server: Jetty(8.0.1.0)

Successfully deleting the vpn user[861391001001]
```

Create Profile

Creates a profile, which defines a particular usage policy for VPN users.

Privileges

Provider administrator or subscriber administrator

HTTP Method

POST

URI

vpn/subscriber/*vpnID***/profiles/**

Request Body

Request body parameters are:

- **profileID**: (string) A unique name for the profile definition. Must be 64 characters or less and start with a letter, and may contain any combination of letters, numbers, and underscore characters. Must be 64 characters or less and start with a letter followed by any combination of word characters.
- **description**: (string) Optional. A description for the profile. The length of this value is limited only by the size of the table field allocated for it in data storage, which is 256 bytes by default.
- **worktimeID**: (string) The identifier of the working time object that defines the working hours for this subscriber. The VPN application uses this value to evaluate rules based on working hours.
- **timeZone**: (integer) The time zone for this subscriber. The time value is used for certain type of policy evaluation scenarios, such as calling restrictions based on time of day. Specify the time zone as the hour offset from UTC.
- **callBarringList**: (object array) Optional. A list of access control items that define outgoing call barring rules for the VPN. See "Using Access Control Rules" for information about the object format.
- **callScreeningList**: (object array) Optional. A list of access control items that define incoming call screening rules for the VPN. See "Using Access Control Rules" for information about the object format.
- **durationControlList**: (integer array) Optional. Call duration limits for various types of calls on the VPN. The array is made up of four numbers that represent the maximum number of seconds for calls of these types, respectively: within the VPN subscriber, outside the VPN subscriber, national, and international.

  For each item, 99999 means no limit.

Response Body

Empty

Examples

Example 8–48 Request

```
POST /vpn/subscriber/vpn001/profiles HTTP/1.1
Host: 10.148.126.22
Accept: */*
Content-Length:325

{"durationControlList":[99999,99999,99999,7200],
"profileID":"hqProfile",
"description":"The headquarters group profile",
"callScreeningList":[],
"timeZone":6,
"callBarringList":[
    {"range":"17909*","timeSpan":0,"permit":false,"type":0},
    {"range":"17910*","timeSpan":0,"permit":false,"type":0}],
"worktimeID":""}
```

Example 8–49 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 207
Server: Jetty(8.0.1.0)

Add the profile successfully. Possible effective fields include: 1.profileID;
2.description; 3.worktimeID; 4.timeZone; 5.callScreeningList; 6.callBarringList;
7.durationControlList. Other fields are ignored.
```

Get All Profiles

Returns a list of profiles in the system. Each profile is represented by its profile ID and URI.

User Authorization

Provider administrator or subscriber administrator

HTTP Method

GET

URI

vpn/subscriber/*vpnID***/profiles/**

Request Body

Empty

Response Body

The response body contains:

- A list of profiles, where each profile is represented as:

  "*profileID*":"*URI*":

  For example, a profile with an ID of **hqProfile** would be identified as:

  "hqprofile":"http://localhost:9001/vpn/subscriber/vpn001/profile/hqprofile"

Examples

Example 8–50 Request

```
GET /vpn/subscriber/vpn001/profiles HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–51 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 561
Server: Jetty(8.0.1.0)

[{"bjGroupProfile":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/profile/bjGroupProfile"},
{"bjEmployeeProfile":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/profile/bjEmployeeProfile"},
{"alicehomeProfile":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/profile/alicehomeProfile"},
{"aliceroamProfile":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/profile/aliceroamProfile"},
{"bjManagerProfile":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/profile/bjManagerProfile"},
{"johnhomeProfile":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/profile/johnhomeProfile"}]
```

Get Profile

Returns information on a specific profile.

Privileges

Provider administrator or subscriber administrator

HTTP Method

GET

URI

vpn/subscriber/*vpnID***/profile/***profile_ID*

Request Body

Empty

Response Body

Response body parameters are:

- **profileID**: (string) A unique name for the profile definition.
- **vpnID**: (string) The identifier of the VPN subscriber instance to which the profile belongs.
- **description**: (string) A description for the profile.
- **worktimeID**: (string) A reference to the working time object that defines the working hours for this subscriber. The VPN application uses this value to evaluate rules based on working hours.
- **timeZone**: (integer) The time zone for this subscriber. The time value is used for certain type of policy evaluation scenarios, such as calling restrictions based on time of day. Specify the time zone as the hour offset from UTC.
- **callBarringList**: (object array) A list of access control items that define outgoing call barring rules for the VPN. See "Using Access Control Rules" for information about the object format.
- **callScreeningList**: (object array) A list of access control items that define incoming call screening rules for the VPN. See "Using Access Control Rules" for information about the object format.
- **durationControlList**: (integer array) Call duration limits for various types of calls on the VPN. The array is made up of four numbers that represent the maximum number of seconds for calls of these types, respectively: within the VPN subscriber, outside the VPN subscriber, national, and international.

  For each item, 99999 means no limit.

Examples

Example 8–52 Request

```
GET /vpn/subscriber/vpn001/profile/bjGroupProfile HTTP/1.1
Host: 10.148.126.22
```

```
Accept: */*
```

Example 8–53 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 264
Server: Jetty(8.0.1.0)

{"durationControlList":[99999,99999,99999,99999],
"profileID":"bjGroupProfile",
"description":"this is bj group profile",
"callScreeningList":[],
"timeZone":8,
"callBarringList":[
   {"range":"17909*","timeSpan":0,"permit":false,"type":0}],
"vpnID":"vpn001",
"worktimeID":""}
```

Modify Profile

Changes settings for a profile.

Privileges

Provider administrator or subscriber administrator

HTTP Method

PUT

URI

vpn/subscriber/*vpnID***/profile/***profile_ID*

Request Body

Request body parameters are:

- **description**: (string) A description for the profile. The length of this value is limited only by the size of the table field allocated for it in data storage, which is 256 bytes by default.
- **worktimeID**: (string) A reference to the working time object that defines the working hours for this subscriber. The VPN application uses this value to evaluate rules based on working hours.
- **timeZone**: (integer) The time zone for this subscriber. The time value is used for certain type of policy evaluation scenarios, such as calling restrictions based on time of day. Specify the time zone as the hour offset from UTC.
- **callBarringList**: (object array) A list of access control items that define outgoing call barring rules for the VPN. See "Using Access Control Rules" for information about the object format.
- **callScreeningList**: (object array) A list of access control items that define incoming call screening rules for the VPN. See "Using Access Control Rules" for information about the object format.
- **durationControlList**: (integer array) Call duration limits for various types of calls on the VPN. The array is made up of four numbers that represent the maximum number of seconds for calls of these types, respectively: within the VPN subscriber, outside the VPN subscriber, national, and international.

  For each item, 99999 means no limit.

Response Body

Empty

Examples

Example 8–54 Request

```
PUT /vpn/subscriber/vpn001/profile/bjGroupProfile HTTP/1.1
Host: 10.148.126.22
Accept: */*
Content-Length:133
```

```
{"callBarringList":[
   {"range":"17909*","timeSpan":0,"permit":false,"type":0},
   {"range":"17910*","timeSpan":0,"permit":false,"type":0}]}
```

Example 8–55 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 197
Server: Jetty(8.0.1.0)

Modify the profile successfully. Possible effective fields include:
1.description; 2.worktimeID; 3.timeZone; 4.callScreeningList; 5.callBarringList;
6.durationControlList. Other fields are ignored.
```

Delete Profile

Removes a profile from the system. For the operation to succeed, the profile cannot be referenced by a user or user group.

Privileges

Provider administrator or subscriber administrator

HTTP Method

DELETE

URI

vpn/subscriber/*vpnID***/profile/***profile_ID*

Request Body

Empty

Response Body

Empty

Examples

Example 8–56 Request

```
DELETE /vpn/subscriber/vpn001/profile/hqProfile HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–57 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 42
Server: Jetty(8.0.1.0)

Successfully delete the profile[hqProfile]
```

Create User Group

Creates a closed user group, which is a group of mobile device users. Groups give you a way to manage the policy for multiple user devices at a time.

After creating a user group, you add users to it by specifying the group as the **closedUserGroupID** value in the user object.

Privileges

Provider administrator or subscriber administrator

HTTP Method

POST

URI

vpn/subscriber/*vpnID***/closedgroups**

Request Body

Request body parameters are:

- **cugID**: (string) A unique name for the user group. Must be 64 characters or less and start with a letter followed by any combination of word characters.
- **description**: (string) Optional. A description for the user group. The length of this value is limited only by the size of the table field allocated for it in data storage, which is 512 bytes by default.
- **parentCugID**: (string) Optional. The parent user group for this group. If a group has a parent group, users that belong to the child group must meet the policies of both child and parent group.

  A group can have only one parent group. However, there can be multiple levels of parent-child groups. That is, the parent group itself can have a parent group.
- **homeProfileID**: (string) Optional. The identifier of an existing policy profile that applies to this group of mobile devices when operated in their home network.
- **roamingProfileID**: (string) Optional. The identifier of an existing policy profile that applies to this group of mobile devices when operated outside their home network.
- **privateCallEnable**: (boolean) Whether the mobile devices in the group can be used for private calls. The default value is false.

Response Body

Empty

Examples

Example 8–58 Request

```
POST /vpn/subscriber/vpn001/closedgroups HTTP/1.1
Host: 10.148.126.22
Accept: */*
Content-Length:190
```

```
{"publicPhoneNumber":[],
"description":"",
"parentCugID":"",
"homeProfileID":"hqGroupProfile",
"cugID":"hqgroup",
"privateCallEnable":true,
"roamingProfileID":""}
```

Example 8–59 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 188
Server: Jetty(8.0.1.0)

Add the vpngroup successfully. Possible effective fields include: 1.cugID;
2.description; 3.parentCugID; 4.privateCallEnable; 5.homeProfileID;
6.roamingProfileID. Other fields are ignored.
```

Get All User Groups

Returns a list of user groups in the system. Each group is represented by its group name (**cugID**) and URI.

User Authorization

Provider administrator or subscriber administrator

HTTP Method

GET

URI

vpn/subscriber/*vpnID***/closedgroups**

Request Body

Empty

Response Body

The response body contains:

- A list of user groups, where each group is represented as:

  "*userGroupID*":"*URI*":

  For example, a user group with an ID of **hqgroup** would be identified as:

  "hqgroup":"http://localhost:9001/vpn/subscriber/vpn01/closedgroup/hqgroup"

Examples

Example 8–60 Request

```
GET /vpn/subscriber/vpn001/closedgroups HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–61 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 331
Server: Jetty(8.0.1.0)

[{"bjgroup":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/closedgroup/bjgroup"},
{"bjEmployee":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/closedgroup/bjEmployee"},
{"bjManager":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/closedgroup/bjManager"},
{"hqgroup":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/closedgroup/hqgroup"}]
```

Get User Group

Returns information on a particular user group.

Privileges

Provider administrator or subscriber administrator

HTTP Method

GET

URI

vpn/subscriber/*vpnID***/closedgroup/***cug_ID*

Request Body

Empty

Response Body

Response body parameters are:

- **cugID**: (string) The name of the user group.
- **vpnID**: (string) The identifier of the VPN subscriber instance to which the user group belongs.
- **description**: (string) A description of the user group.
- **parentCugID**: (string) The parent user group of this group.
- **homeProfileID**: (string) The policy to be applied to users in the group when operating within their home network.
- **roamingProfileID**: (string) The policy to be applied to users in the group when operating outside of their home network.
- **privateCallEnable**: (boolean) Whether the mobile devices in the group can be used for private calls.
- **childCugIDs**: (string array) Children user groups of this group. A child user group inherits and extends the policy specified in the parent user group.

  Note that you cannot add child user groups directly. The value of this response parameter is derived from parent user group settings in the child groups.
- **publicPhoneNumber**: (string array) The telephone numbers of the users that are direct members of this group.

  Note that you do not add users by public number to a group directly. The value of this response parameter is derived from the group settings of individual user definitions.

Examples

Example 8–62 Request

```
GET /vpn/subscriber/vpn001/closedgroup/hqgroup HTTP/1.1
Host: 10.148.126.22
```

```
Accept: */*
```

Example 8–63 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 190
Server: Jetty(8.0.1.0)

{"publicPhoneNumber":[],
"description":"",
"parentCugID":"",
"homeProfileID":"hqGroupProfile",
"vpnID":"vpn001",
"cugID":"hqgroup",
"privateCallEnable":true,
"childCugIDs":[],
"roamingProfileID":""}
```

Modify User Group

Changes settings for a user group.

Privileges

Provider administrator or subscriber administrator

HTTP Method

PUT

URI

vpn/subscriber/*vpnID***/closedgroup/***cug_ID*

Request Body

Request body parameters are:

- **description**: (string) A description for the user group. The length of this value is limited only by the size of the table field allocated for it in data storage, which is 512 bytes by default.
- **parentCugID**: (string) The parent user group for this group. If a group has a parent group, users that belong to the child group must meet the policies of both child and parent group.

  A group can have only one parent group. However, there can be multiple levels of parent-child groups. That is, the parent group itself can have a parent group.
- **homeProfileID**: (string) The policy to be applied to users in the group when operating within their home network.
- **roamingProfileID**: (string) The policy to be applied to users in the group when operating outside of their home network.
- **privateCallEnable**: (boolean) Whether the mobile devices in the group can be used for private calls. The default value is false.

Response Body

Empty

Examples

Example 8–64 Request

```
PUT /vpn/subscriber/vpn001/closedgroup/hqgroup HTTP/1.1
Host: 10.148.126.22
Accept: */*
Content-Length:60

{"description":"User group for headquarter-based employees"}
```

Example 8–65 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 182
```

```
Server: Jetty(8.0.1.0)

Modify the vpngroup successfully. Possible effective fields include:
1.description; 2.parentCugID; 3.privateCallEnable; 4.homeProfileID;
5.roamingProfileID. Other fields are ignored.
```

Delete User Group

Removes a user group from the system. For the operation to succeed, the user group must not contain any users (that is, it cannot be referenced by a user definition), as indicated in the group's **publicPhoneNumber** attribute.

Privileges

Provider administrator or subscriber administrator

HTTP Method

DELETE

URI

vpn/subscriber/*vpnID***/closedgroup/***cug_ID*

Request Body

Empty

Response Body

Empty

Examples

Example 8–66 Request

```
DELETE /vpn/subscriber/vpn001/closedgroup/hqgroup HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–67 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 50
Server: Jetty(8.0.1.0)

Successfully deleting the ClosedUserGroup[hqgroup]
```

Create Person

Creates a person object, which identifies a particular end user in the subscriber organization. A person object is referenced by one or more user objects, which associates the end use with a device

Privileges

Provider administrator or subscriber administrator

HTTP Method

POST

URI

vpn/subscriber/*vpnID***/persons**

Request Body

Request body parameters are:

- **personID**: (string) A unique identifier for the end user. Must be 64 characters or less and start with a letter followed by any combination of word characters.
- **name**: (string) An informal name for the end user. Must be 32 characters or less and start with a letter followed by any combination of word characters and spaces.

Response Body

Empty

Examples

Example 8–68 Request

```
POST /vpn/subscriber/vpn001/persons HTTP/1.1
Host: 10.148.126.22
Accept: */*
Content-Length:70

{"name":"Alice M",
"personID":"alice"}
```

Example 8–69 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 109
Server: Jetty(8.0.1.0)

Add the person successfully. Possible effective fields include: 1.personID;
2.name. Other fields are ignored.
```

Get All Persons

Returns a list of the person objects in the system. Each person object is represented by name and URI.

User Authorization

Provider administrator or subscriber administrator

HTTP Method

GET

URI

vpn/subscriber/*vpnID***/persons**

Request Body

Empty

Response Body

The response body contains:

- A list of persons, where each person is represented as:

  "*personID*":"*URI*":

  For example, a person with an ID of **alice** would be identified as:

  "alice":"http://localhost:9001/vpn/subscriber/vpn001/person/alice"

Examples

Example 8–70 Request

```
GET /vpn/subscriber/vpn001/persons HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–71 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 143
Server: Jetty(8.0.1.0)

[{"User0":"http://10.148.126.22:80/vpn/subscriber/vpn001/person/User0"},
{"alice":"http://10.148.126.22:80/vpn/subscriber/vpn001/person/alice"}]
```

Get Person

Returns information on a particular person object.

Privileges

Provider administrator or subscriber administrator

HTTP Method

GET

URI

vpn/subscriber/*vpnID***/person/***personID*

Request Body

Empty

Response Body

Response body parameters are:

- **name**: (string) An informal name of the end user.
- **vpnID**: (string) The identifier of the VPN subscriber instance to which the person belongs. The association is created through the user object that references the person.
- **vpnUserIDs**: (string array) The list of users that reference this person object by public phone number.
- **personID**: (string) The unique identifier of the end user.

Examples

Example 8–72 Request

```
GET /vpn/subscriber/vpn001/person/alice HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–73 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 70
Server: Jetty(8.0.1.0)

{"name":"Alice M",
"vpnID":"vpn001",
"vpnUserIDs":[],
"personID":"alice"}
```

Modify Person

Changes settings for a particular person.

Privileges

Provider administrator or subscriber administrator

HTTP Method

PUT

URI

vpn/subscriber/*vpnID***/person/***personID*

Request Body

Request body parameters are:

- **name**: (string) An informal name for the end user. Must be 32 characters or less and start with a letter followed by any combination of word characters and spaces.

Response Body

Empty

Examples

Example 8–74 Request

```
PUT /vpn/subscriber/vpn001/person/alice HTTP/1.1
Host: 10.148.126.22
Accept: */*
Content-Length:18

{"name":"Alice O"}
```

Example 8–75 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 100
Server: Jetty(8.0.1.0)

Modify the person successfully. Possible effective fields include: 1.name. Other
fields are ignored.
```

Delete Person

Removes a person from the subscriber. For the operation to succeed, the person cannot be referenced by a user object.

Privileges

Provider administrator or subscriber administrator

HTTP Method

DELETE

URI

vpn/subscriber/*vpnID***/person/***personID*

Request Body

Empty

Response Body

Empty

Examples

Example 8–76 Request

```
DELETE /vpn/subscriber/vpn001/person/alice HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–77 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 39
Server: Jetty(8.0.1.0)

Successfully deleting the person[alice]
```

Create Working Time

Defines the working time for a subscriber or profile. You can create call restriction rules that control calls based on a user's working times.

A working time definition is made up of four parameters, each of which contains an array of seven integers. Each integer represents a day of the week, starting on Sunday. The four parameters represent the starting hour, starting minutes, ending hour, and ending minutes. Hours are expressed as an integer, from 0 through 23 (representing a 24-hour based clock). Minutes are expressed as integers from 0 through 59.

For example, the following working time definition specifies a work time of 9:30 AM to 6:00 PM from Monday through Friday:

```
{
   "startHour":[0,9,9,9,9,9,0],
   "startMin":[0,30,30,30,30,30,0],
   "endHour":[0,18,18,18,18,18,0],
   "endMin":[0,0,0,0,0,0,0]
}
```

Privileges

Provider administrator or subscriber administrator

HTTP Method

POST

URI

vpn/subscriber/*vpnID***/worktimes**

Request Body

Request body parameters are:

- **worktimeID**: (string) A unique identifier for the work time definition. Must be 64 characters or less and start with a letter followed by any combination of word characters. See "JSON Body Parameters" for information about word characters.
- **startHour**: (integer array) The hour when work starts each day. Specified as a seven integer array. Each item is a value from 0 to 23 that represents the starting hour. The first integer represents the Sunday start time and the last integer represents the Saturday start time.
- **startMin**: (integer array) The minute when work starts each day. Specified as a seven integer array. Each integer is a value ranging from 0 to 59 that represents the minute at which work starts. The first integer represents the Sunday start time and the last integer represents the Saturday start time.
- **endHour**: (integer array) The hour when work terminates each day. Specified as a seven integer array. Each integer is a value ranging from 0 to 23 that represents the ending hour. The first integer represents the Sunday end time and the last integer represents the Saturday end time.
- **endMin**: (integer array) The minute when work terminates ends each day. Specified as a seven integer array. Each integer is a value ranging from 0 to 59 that

represents the minute when work ends. The first integer represents the Sunday end time and the last integer represents the Saturday end time.

Response Body

Empty

Examples

Example 8–78 Request

```
POST /vpn/subscriber/vpn001/worktimes HTTP/1.1
Host: 10.148.126.22
Accept: */*
Content-Length:164

{"startMin":[0,30,30,30,30,30,0],
"startHour":[0,7,7,7,7,7,10],
"endMin":[0,0,0,0,0,0,0],
"endHour":[0,16,16,16,16,16,14],
"worktimeID":"workingtime3"}
```

Example 8–79 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 155
Server: Jetty(8.0.1.0)

Add the workingtime successfully. Possible effective fields include:
1.worktimeID; 2.startHour; 3.startMin; 4.endHour; 5.endMin.  Other fields are
ignored.
```

Get All Working Times

Returns a list of working time definitions in the system. Each working time definition is represented by name and URI.

User Authorization

Provider administrator

HTTP Method

GET

URI

vpn/subscriber/*vpnID***/worktimes**

Request Body

Empty

Response Body

The response body contains:

- A list of working time definitions, where each working time definition is represented as:

  "*workingTimeID*":"*URI*":

  For example, a working time with an ID of **wt001** would be identified as:

  "wt001":"http://localhost:9001/vpn/subscriber/vpn001/worktime/wt001"

Examples

Example 8–80 Request

```
GET /vpn/subscriber/vpn001/worktimes HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–81 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 175
Server: Jetty(8.0.1.0)

[{"workingtime2":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/worktime/workingtime2"},
{"workingtime1":
   "http://10.148.126.22:80/vpn/subscriber/vpn001/worktime/workingtime1"}]
```

Get Working Time

Returns a particular working time definition.

Privileges

Provider administrator or subscriber administrator

HTTP Method

GET

URI

vpn/subscriber/*vpnID***/worktime/***worktimeID*

Request Body

Empty

Response Body

Response body parameters are:

- **worktimeID**: (string) The unique identifier of the work time definition.
- **vpnID**: (string) The identifier of the VPN subscriber instance to which the work time belongs.
- **startHour**: (integer array) The hour when work starts each day. Specified as a seven integer array. Each integer is a value ranging from 0 to 23 that represents the starting hour. The first integer represents the Sunday start time and the last integer represents the Saturday start time.
- **startMin**: (integer array) The minute when work starts each day. Specified as a seven integer array. Each integer is a value ranging from 0 to 59 that represents the minute at which work starts. The first integer represents the Sunday start time and the last integer represents the Saturday start time.
- **endHour**: (integer array) The hour when work terminates each day. Specified as a seven integer array. Each integer is a value ranging from 0 to 23 that represents the ending hour. The first integer represents the Sunday end time and the last integer represents the Saturday end time.
- **endMin**: (integer array) The minute when work terminates ends each day. Specified as a seven integer array. Each integer is a value ranging from 0 to 59 that represents the minute when work ends. The first integer represents the Sunday end time and the last integer represents the Saturday end time.

Examples

Example 8–82 Request

```
GET /vpn/subscriber/vpn001/worktime/workingtime2 HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–83 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 162
Server: Jetty(8.0.1.0)

{"startMin":[0,0,0,0,0,0,0],
"startHour":[0,19,19,19,19,19,0],
"vpnID":"vpn001",
"endMin":[0,0,0,0,0,0,0],
"endHour":[0,23,23,23,23,23,0],
"worktimeID":"workingtime2"}
```

Modify Working Time

Changes settings for a particular working time definition.

Privileges

Provider administrator or subscriber administrator

HTTP Method

PUT

URI

vpn/subscriber/*vpnID***/worktime/***worktimeID*

Request Body

Request body parameters are:

- **startHour**: (integer array) The hour when work starts each day. Specified as a seven integer array. Each integer is a value ranging from 0 to 23 that represents the starting hour. The first integer represents the Sunday start time and the last integer represents the Saturday start time.
- **startMin**: (integer array) The minute when work starts each day. Specified as a seven integer array. Each integer is a value ranging from 0 to 59 that represents the minute at which work starts. The first integer represents the Sunday start time and the last integer represents the Saturday start time.
- **endHour**: (integer array) The hour when work terminates each day. Specified as a seven integer array. Each integer is a value ranging from 0 to 23 that represents the ending hour. The first integer represents the Sunday end time and the last integer represents the Saturday end time.
- **endMin**: (integer array) The minute when work terminates ends each day. Specified as a seven integer array. Each integer is a value ranging from 0 to 59 that represents the minute when work ends. The first integer represents the Sunday end time and the last integer represents the Saturday end time.

Response Body

Empty

Examples

Example 8–84 Request

```
PUT /vpn/subscriber/vpn001/worktime/workingtime3 HTTP/1.1
Host: 10.148.126.22
Accept: */*
Content-Length:33

{"endHour":[0,17,17,17,17,17,14]}
```

Example 8–85 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
```

```
Content-Length: 144
Server: Jetty(8.0.1.0)

Modify the workingtime successfully. Possible effective fields include:
1.startHour; 2.startMin; 3.endHour; 4.endMin.  Other fields are ignored.
```

Delete Working Time

Removes a working time definition from the system. For this operation to succeed, the working time cannot be referenced by a subscriber or profile object.

Privileges

Provider administrator or subscriber administrator

HTTP Method

DELETE

URI

vpn/subscriber/*vpnID***/worktime/***worktimeID*

Request Body

Empty

Response Body

Empty

Examples

Example 8–86 Request

```
DELETE /vpn/subscriber/vpn001/worktime/workingtime3 HTTP/1.1
Host: 10.148.126.22
Accept: */*
```

Example 8–87 Response

```
HTTP/1.1 200 OK
Content-Type: application/text
Content-Length: 52
Server: Jetty(8.0.1.0)

Successfully deleting the working time[workingtime3]
```