Oracle® Communications Service Broker

Subscriber Store User's Guide

Release 6.0

E23529-02

March 2012

ORACLE®

Oracle Communications Service Broker Subscriber Store User's Guide, Release 6.0

E23529-02

Contents

6 Subscriber Provisioning API Reference

Preface

This document describes how to set up and administer the Subscriber Store feature of the Oracle Communications Service Broker.

Audience

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

This documentation is based on the assumption that you are familiar with:

- The operating system on which your system is installed
- Database management systems
- Java
- SOAP Web services
- Web servers and protocols

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*

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

About Subscriber Provisioning

This chapter introduces Oracle Communications Service Broker subscriber storage and management features.

About the Subscriber Store

Oracle Communications Service Broker applications enable telecommunication network operators to deliver user-targeted services. For example, Policy Controller applications can apply Quality of Service (QoS) parameters for network access that are customized for a particular user. Similarly, Online Mediation Controller applications can expose the services of back-end charging systems to network end users. The Online Mediation Controller services include balance inquiry and account top-up services.

To deliver user-targeted services, Service Broker maintains its own representation of end user information in a repository called the Subscriber Store. The data in the Subscriber Store specifies the service access preferences and parameters for each network end user.

The information in the Subscriber Store is intended to supplement the information in an operator's existing subscriber repository or billing system. While existing subscriber repositories contain general account information, the Subscriber Store contains information specifically relating to the delivery of user-targeted services over diverse networks.

To implement user-targeted Service Broker features, you must create, configure, and populate the Subscriber Store repository.

The Service Broker controller types that use the Subscriber Store are:

- Policy Controller
- Online Charging Mediation Controller
- Co-deployed Policy Controller and Online Charging Mediation Controller

The Subscriber Store is not used by the Service Broker VPN or Social Voice Communicator (SVC) features. Subscriber profiles for those features are stored and managed separately. Also, the Service Controller does not use the Subscriber Store.

See *Oracle Communications Service Broker Concepts Guide* for more information about Service Broker products. Also see *Oracle Communications Service Broker VPN Implementation Guide* and *Oracle Communications Service Broker SVC Implementation Guide* for information on managing user information for those products.

About Subscriber Profiles

In the Subscriber Store, each subscriber is represented by a profile. A subscriber profile is associated with a particular user on the network through the network ID for that user. The network ID can be an International Mobile Subscriber Identity (IMSI) number, SIP address, E.164 number, or other type of network ID. The exact form of the ID differs by network.

In addition to external network IDs, each subscriber has an internal user ID allocated by the Subscriber Store. The user ID uniquely identifies an end user in the Subscriber Store, associating a user to potentially multiple external networks IDs.

In addition to network identities, a subscriber profile may contain the following information:

- Service policy information
- iFC data, which defines a customer specific service chain
- General user information such as account state and language preference

Subscriber profiles may also contain elements called profile data extensions. These are application-specific data elements in the form of name-value pairs.

See "Subscriber Profile Data Model" for more information on the subscriber profile data model.

Configuration Overview

The general steps for setting up the Subscriber Store are:

1. Configure Subscriber Store operating settings.
2. Configure the persistence mechanism for the Subscriber Store.
3. Populate and maintain the Subscriber Store with user information.

The Subscriber Store settings consist primarily of access settings for the Subscriber Provisioning API. The API provides operations for adding and managing subscriber information in the store. As a SOAP-based API, the Subscriber Provisioning API allows the Subscriber Store to be integrated with existing subscriber repositories and management applications.

2

Configuring Subscriber Data Storage

This chapter describes how to set up data storage for Oracle Communications Service Broker applications.

About Subscriber Data Persistence

The Subscriber Store can use one of the following types of persistence mechanisms:

- Oracle Berkeley DB
- Oracle Database 11g

The persistence mechanism that the Subscriber Store uses depends on which persistence package is installed in the Service Broker domain. By default, the Berkeley DB persistence package is installed. Therefore, to complete the Subscriber Store persistence implementation with Berkeley DB, you only need to configure the file storage locations and settings for the managed servers in your deployment.

Alternatively, to switch to Oracle Database storage, you must install the package and then set up the database connections for each managed server.

The following steps provide an overview of how to configure persistence for the Service Broker Subscriber Store. See configuring data storage information in *Oracle Communications Service Broker Installation Guide* for more details.

Configuring Oracle Berkeley DB Persistence

Oracle Berkeley DB is a file-based persistence mechanism that stores data in the local file system.

To configure Oracle Berkeley DB file-based persistence for the Subscriber Store:

1. Verify that the Berkeley DB package is installed in the domain as follows:
   a. In the Administration Console, expand the **Domain Management** node.
   b. Click the **Packages** node.
   c. Verify that the package list includes the following package:

      oracle.ocsb.app.rcc.service.subscriber_store.providers.store.config_bdb
2. If the managed servers in the domain are not already configured to use Berkeley DB persistence, configure the settings for the domain. For more information on configuring Berkeley DB settings for managed servers, see *Oracle Communications Service Broker Installation Guide*.

Configuring Oracle Database 11g Persistence

To configure the Subscriber Store to use Oracle Database 11g for data persistence:

1. Remove the existing Subscriber Store persistence package from the domain. By default, it is:

   oracle.ocsb.app.rcc.service.subscriber_store.providers.store.config_bdb

2. Install the Oracle Database persistence package:

   oracle.ocsb.app.rcc.service.subscriber_store.providers.store.config_db

3. Make sure the run level for the package matches that of the following package:

   oracle.ocsb.app.rcc.service.subscriber_store.core

4. Configure the database schema for the Subscriber Store using the following SQL script:

   *Oracle_home***/ocsb/admin_console/scripts/database/subscriber_store.sql**

   To use the script to configure your database schema, run the script using the SQL client tool or interface provided by your database management system.

5. If the managed servers in the domain are not already configured to connect to the database, configure the connections in the **Data Store** node under **Domain Management**.

   For the Subscriber Store database, the connection pool name value in the JDBC configuration should be **oracle_driver**.

   See *Oracle Communications Service Broker Installation Guide* for more information about configuring data storage.

3

Enabling Subscriber Provisioning Service Connectivity

This chapter describes how to configure the SOAP endpoint settings that control how clients access the Subscriber Provisioning API served by Oracle Communications Service Broker.

About Subscriber Provisioning Service Connections

By default, Service Broker is not configured to serve Subscriber Provisioning API requests. You must configure SOAP endpoint settings to enable the service.

The general configuration tasks for enabling the Subscriber Provisioning API service are as follows:

1. Enable HTTP network access by opening an HTTP listening port.
2. Configure common SOAP connection settings.
3. Configure specific end point settings for SOAP and the Subscriber Provisioning service end point.
4. Add an incoming routing rule that directs requests to the Subscriber Provisioning service application.
5. Test the connection.

The following steps provide more information about performing the configuration.

General HTTP connection and Web service settings are also used by other Service Broker features. In some cases, parts of the following configuration may already exist in your Processing Domain, depending on the features that have been implemented.

About Connection Security

Client connections to the Subscriber Provisioning API are subject to the security requirements applicable to the underlying HTTP connection. That is, if the HTTP port on which Service Broker serves the Subscriber Provisioning API requires HTTP Basic Authentication credentials, SOAP requests to the service must meet the same requirement.

In addition, you can apply Web service-specific requirements for the port. For example, you can configure SOAP-specific timeout settings or authentication requirements. Service Broker supports SOAP authentication in the form of WSSE UsernameToken authentication.

Web service connection settings are configurable by Service Broker application. For example, you can use UsernameToken authentication for the Subscriber Provisioning service API while using Basic Authentication for the Top-Up service API.

Service Broker SOAP Web Services supports UsernameToken authentication as defined by the Basic Security Profile standard:

http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html#UsernameToken

When UsernameToken authentication is enabled, each SOAP request must contain a valid UsernameToken security element. The element must include a password that matches the password of the specified credential in the Service Broker credential store.

The following sections include procedural information for configuring security for both the HTTP connection and for the Subscriber Provisioning service end point. For general information about securing the Service Broker implementation, see *Oracle Communications Service Broker System Administrator's Guide*.

Enabling HTTP Network Access

Service Broker serves the Subscriber Provisioning API over an HTTP listen port you open in the Signaling Tier domain.

To configure HTTP connectivity for the Subscriber Provisioning API:

1. In the navigation tree, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Expand the **SSU Web Services** node.
4. Click the **General** item.
5. Select the **HTTP** tab.
6. In the Server subtab, select the **Network Access** subtab.
7. Click **New**.
8. In the dialog box, set the properties for the HTTP listener as follows:
   - **Server Address**: The local IP address or host name to which the port is bound. This should be the address of the Signaling Tier server or server cluster in your deployment.
   - **Server Port**: An available listening port on Service Broker serves API client requests, such as **8989**. This is the port number on which the Signaling Tier servers will listen for incoming HTTP requests to the Subscriber Provisioning API.
   - **Protocol**: The protocol used by the service. Choose **HTTPS** for secure HTTP or **HTTP** for unsecured HTTP. Oracle recommends using HTTPS for production deployments.
   - **SSL Client Auth**: Whether SSL client certificate authentication is required for the connection. Enter **false** to disable SSL client certificate authentication, or **true** to require it. Enter **true** only if using HTTPS for the Protocol, in which case you will also need to set the key store and trust store identifiers.
   - **Keystore Id**: The key you used when loading the keystore in the Credential Store. Use only with HTTPS. If you are using HTTP, this field can be left blank. If you have not already, load the keystore associated with the ID into the credential store. See *Oracle Communications Service Broker System Administrator's Guide* for more information about the credential store.

- **Truststore Id**: The key you used when loading the trust store into the Credential Store. Use only with HTTPS. If using HTTP, this field can be left blank. See *Oracle Communications Service Broker System Administrator's Guide* for more information about the credential store.
- **Target**: The managed server to which this configuration applies. Leave blank to apply the configuration to all managed server in the deployment. Specify a managed server name only if you want custom settings for individual managed servers.

9. Click **OK**.

For more information, see the discussion of routing rules for the Web Services Signaling Server Unit (SSU) in *Oracle Communications Service Broker Signaling Domain Configuration Guide*.

Configuring the Subscriber Provisioning Service End Point

The Subscriber Provisioning service end point settings specify the URI path at which Service Broker serves API requests. It also defines specific connection settings for the path.

The Signaling Tier domain contains a pre-configured Subscriber Provisioning end point definition, which you should verify and modify if needed.

To verify and modify the pre-configured end point settings for the Subscriber Provisioning service:

1. In the Administration Console interface navigation tree, expand the **OCSB** node.
2. Expand the **Signaling Tier** node.
3. Expand **SSU Web Services** node.
4. Click the **Subscriber Provisioning** item.
5. In the **End Point** tab, select the existing Subscriber Provisioning configuration.
6. Click the **Update** button.
7. In the dialog box, configure the following properties:
   - **URI**: The path of the Subscriber Provisioning API service relative to the root SOAP path. By default, the path is:

     /SubscriberProvisioning
   - **Implementation Class**: The class that implements the Subscriber Provisioning API service. Unless you have implemented a custom service class, the value should be the following default value:

     oracle.ocsb.app.rcc.service.subscriber_store.ws.SubscriberProvisioningService
   - **Authentication Method**: Whether Service Broker authenticates incoming SOAP requests. Set to either:
     - **NONE**: Specifies no authentication by the endpoint. Choose this option to rely on the authentication mechanism of the underlying HTTP connection (such as Basic Authentication or client certificate authentication) or to bypass authentication requirements.
     - **USERNAME_TOKEN** for WSSE UsernameToken password authentication.

- **credentialKey**: If using UsernameToken password authentication, the credential key for the password. The credential key is the identifier for this credential in the credential store.

8. If you have not already done so, store the password for the UsernameToken credential in the credential store.

   The UsernameToken password should be stored in the Credential Store as a one-way credential. It is used to validate incoming credentials and not added to outgoing requests. See *Oracle Communications Service Broker Installation Guide* for more information about the Credential Store.

9. Click **OK**.

The new settings appear in the end point definition.

Creating the Incoming Routing Rule

An incoming routing rule controls how Service Broker Signaling Tier route incoming messages to the Service Broker application or interworking module.

To create the incoming routing rule for the Subscriber Provisioning service:

1. In the navigation tree, expand **OCSB**.
2. Expand the **Signaling Tier** node.
3. Expand the **SSU Web Services** node.
4. Click the **General** item.
5. Click the **SSU WS** tab.
6. Select the **Incoming Routing Rules** subtab.
7. Click the **New** button.
8. In the dialog box, set the incoming routing rules as follows:
   - **Name**: A unique name for the incoming routing rule. You can choose any descriptive name.
   - **Service Name**: Set this value to **SubscriberService**, the internal service identifier for the Subscriber Provisioning service.
   - **Alias**: Set this value to **ssu:ocsb/provisioning**, the internal application address for the Subscriber Provisioning service application.
9. Click **OK**.

After committing the changes to the configuration, restart the managed servers to have the new settings take effect. You can then access the WSDL for the Subscriber Provisioning service (as described in "Testing Web Service Access").

For more information on the Web Services SSU settings, see *Oracle Communications Service Broker Signaling Domain Configuration Guide*.

Testing Web Service Access

You can test your HTTP and Web service connectivity configuration from a Web browser by navigating to the Subscriber Provisioning WSDL. Before starting, ensure that you have committed your configuration changes to the running servers in your environment.

To navigate to the WSDL, go to the following address:

http://*host***:***port***/soap/SubscriberProvisioning?wsdl**

Where *host* is the host name or IP address and *port* is the server port number you specified as the server address (as described in "Enabling HTTP Network Access").

If the WSDL is accessible, you can start developing your API client applications. Many integrated development environments (IDEs) can generate client code you can use as a starting point for your application by importing the WSDL into the IDE.

If the WSDL is not accessible, you can test the HTTP port by navigating to the service index page, at the following address:

http://*host***:***port***/soap**

The page lists the SOAP interfaces exposed in the domain. If unavailable, double-check your Web Services SSU configuration.

In addition, you can view the **server.log** file at the following location to identify run-time configuration errors, such as listening port conflicts.

*Oracle_home***/ocsb/managed_server**

4

Using Subscriber Account Lifecycles

This chapter describes how to customize subscriber account lifecycle states, which represent the status of subscriber accounts in the Oracle Communications Service Broker Subscriber Store.

About Subscriber Account Lifecycle States

The lifecycle state value in the subscriber profile represents the state of the account associated with a particular subscriber. External systems can set and read the lifecycle state of a subscriber through the Subscriber Provisioning API.

The account lifecycle state is an element in the subscriber profile data model. Its value appears in the **accountState** element, which is a child element of **globalProfileData**. See "Subscriber Profile Data Model" for more information about profile data.

The Service Broker processing domain configuration includes predefined lifecycle states. The default states correspond to typical account states in a network subscriber repository. However, you should modify these values to match those used by the systems in your environment that will interact with Service Broker, such as your existing OCS.

The default lifecycle states are:

- Pre-Active
- Active
- Suspended
- Locked
- Deactivated

The Subscriber Provisioning API service validates the state values submitted in subscriber create or update requests. However, while the Subscriber Provisioning service validates and maintains state values in the Subscriber Store, Service Broker does not otherwise interpret the meaning of a particular state value. It is the role of the back end system to associate a particular state with service provisioning logic.

Adding a Lifecycle State

To add a lifecycle state:

1. In the Domain Navigation pane, expand the **OCSB** node.
2. Expand the **Processing Tier** node.
3. Expand the **Subscriber Store** node.

4. Click the **Data Validation** item.
5. In the **Account Lifecycle** tab, click the **New** button.
6. Enter the account state value in the **Lifecycle State** field.
7. Click **OK**.

The new state appears in the list of lifecycle states. After you commit the changes to the running configuration, the Subscriber Provisioning Service will accept the account state value you configured.

To use JMX MBeans to add a lifecyle state, use the **addState** operation in the **lifecycleConfigMBean**, which has the following object name:

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.service.subscriber_store.datavalidation,version=6.0.0.SNAPSHOT,name0=lifecycleConfig

Changing a Lifecycle State

To change a lifecycle state:

1. In the Domain Navigation pane, expand the **OCSB** node.
2. Expand the **Processing Tier** node.
3. Expand the **Subscriber Store** node.
4. Click the **Data Validation** item.
5. In the **Account Lifecycle** tab, select the state you want to modify in the State Configuration list.
6. Click the **Update** button.
7. Enter the new value in the **Lifecycle State** field and click **OK**.

The renamed state appears in the list of lifecycle states.

To use JMX MBeans to modify a lifecyle state, edit the **name** attribute value for the MBean instance of the state you want to change under the **lifecycleConfigMBean**, which has the following object name:

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.service.subscriber_store.datavalidation,version=6.0.0.SNAPSHOT,name0=lifecycleConfig

Deleting a Lifecycle State

To remove a lifecycle state:

1. In the Domain Navigation pane, expand the **OCSB** node.
2. Expand the **Processing Tier** node.
3. Expand the **Subscriber Store** node.
4. Click the **Data Validation** item.
5. In the **Account Lifecycle** tab, select the state you want to remove in the State Configuration list.
6. Click the **Delete** button.
7. Confirm the operation by clicking **OK**.

The removed state is removed from the list of lifecycle states.

To use JMX MBeans to remove a lifecyle state, use the **removeState** operation in the **lifecycleConfigMBean**, which has the following object name:

oracle:type=oracle.axia.cm.ConfigurationMBean,name=oracle.ocsb.app.rcc.service.subscriber_store.datavalidation,version=6.0.0.SNAPSHOT,name0=lifecycleConfig

When invoking the operation, pass the instance number that corresponds to the state to remove.

5

Using the Subscriber Provisioning API

This chapter describes the SOAP Subscriber Provisioning API, the application programming interface client applications use to manage profiles in the Oracle Communications Service Broker Subscriber Store.

About the Subscriber Provisioning API

The Subscriber Provisioning API is a Document-style SOAP API that provides operations for creating, updating, retrieving and removing profiles in the Subscriber Store.

The operations in the API correspond to what are commonly called the CRUD operations, which is an acronym for create, read, update, and delete.

In the Subscriber Provisioning API, the operations are named:

- **storeSubscriber**: Creates subscriber profiles.
- **getSubscriber**: Returns data that makes up a subscriber profile.
- **updateSubscriber**: Modifies subscriber profiles.
- **deleteSubscriber**: Removes subscriber profiles from the Subscriber Store.

The Subscriber Provisioning API provides a WSDL file that describes the operations and data types in the API. After installing Service Broker and enabling Web service connectivity, you can access the Subscriber Provisioning WSDL at the following default location:

https://*host***:***port***/soap/SubscriberProvisioning?wsdl**

where *host* is the host name or IP address and *port* is the server port number you specified as the server address (as described in "Enabling HTTP Network Access").

The protocol type (http or https), SOAP root URI (**/soap**, by default), and service name are all configurable in the Service Broker domain. Your path to the WSDL may be different.

"Subscriber Provisioning API Reference" contains reference information about each operation, along with sample requests and responses.

Subscriber Profile Data Model

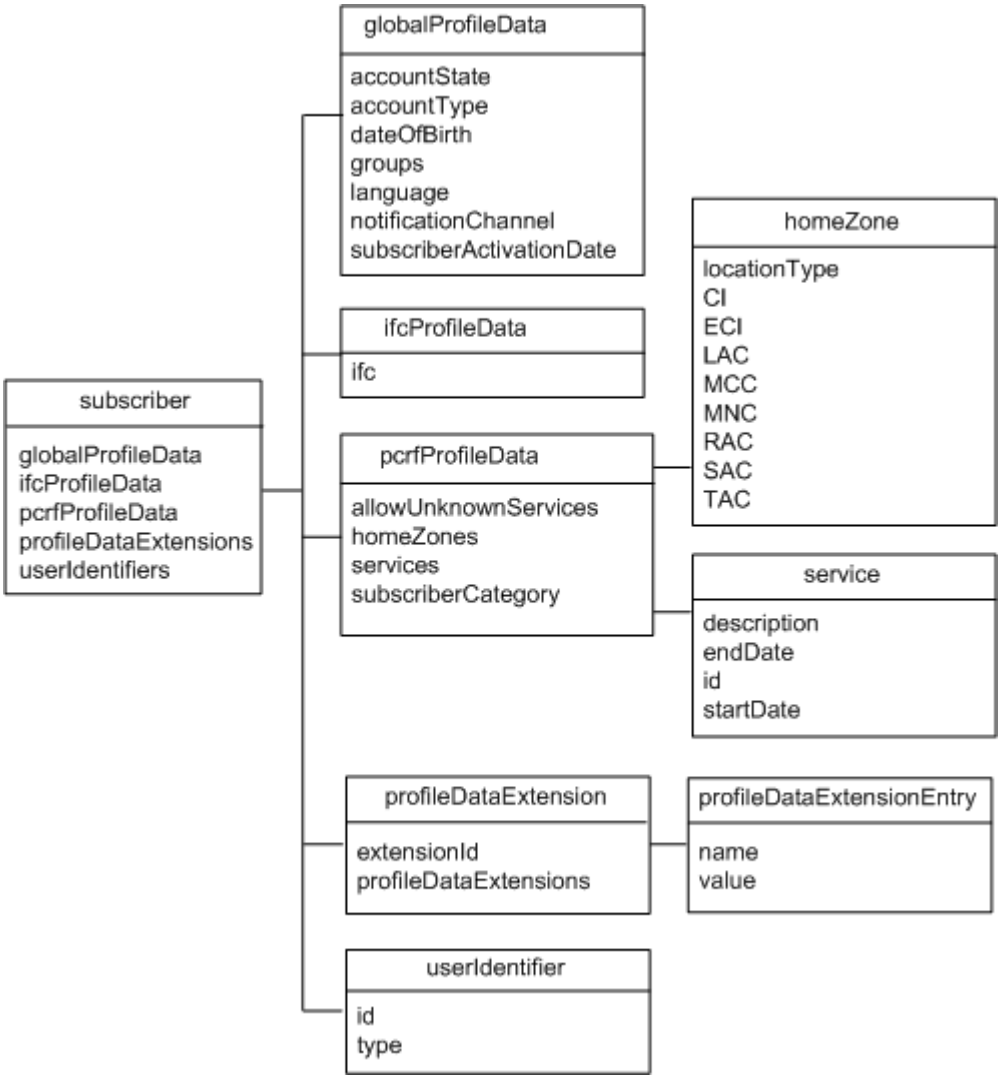
The subscriber profile data model defines the elements of a subscriber profile. The model includes general information for the subscriber along with feature-specific information, such as Policy Controller data.

The subscriber profile data model is defined by an XML Schema file. You can access the schema at the following default location:

http://*host***:***port***/soap/SubscriberProvisioning?xsd=1**

where *host* is the host name or IP address and *port* is the server port number you specified as the server address (as described in "Enabling HTTP Network Access").

Figure 5–1 Subscriber Profile Data Model



As shown in Figure 5–1, the top-level data elements that compose the subscriber profile are:

- globalProfileData Element
- ifcProfileData Element
- pcrfProfileData Element
- profileDataExtension Element
- userIdentifier Element

The following sections provide information on the data elements of the subscriber profile. For more information on how specific applications use the data, see the Service Broker product implementation guide specific for your controller type.

globalProfileData Element

The **globalProfileData** element defines general properties for a subscriber. These values are used by multiple Service Broker applications. This element contains the following elements:

- **accountState**: The account lifecycle state that indicates the status of the account, such as active or inactive. See "Using Subscriber Account Lifecycles" for more information about account lifecycle states.
- **accountType**: The account type from the options prepaid, postpaid, or hybrid. Hybrid accounts can use both prepaid and postpaid charging methods.
- **dateOfBirth**: The birth date of the subscriber. The date is in *YYYYMMDD* format, **19910128**.
- **groups**: Group identifier for the subscriber.
- **language**: The language to use for the subscriber. Service Broker applications that interact with subscribers can use this attribute to select a response or adapt a message for the subscriber.
- **notificationChannel**: The notification channel used to send notifications to the user.
- **subscriberActivationDate**: The date, in *YYYYMMDD* format, when the subscriber account was first activated, such as **20111231**.

ifcProfileData Element

The **ifcProfileData** element contains the iFC logic for the subscriber. The iFC defines the service chain for a particular subscriber, which services are invoked and the order in which they are invoked.

pcrfProfileData Element

The **PcrfProfileData** element contains subscriber data used by the Policy Controller. **PcrfProfileData** includes the following elements:

- **allowUnknownServices**: Whether the subscriber is permitted to access services that are not specifically allocated to that subscriber, as indicated by the **services** field.
- **homeZone**: The home, non-roaming network zone for the subscriber. Policy Controller administrators can apply charging rates or service access decisions based on whether the subscriber is using the network from their home zone.

  The **homeZone** element is made up of a **locationType**, which identifies the protocol type in which the location information is represented, such as IEEE-802.11a or 3GPP-GERAN. Depending on the location type, it also includes elements that identify the location as specified for that type.

  Possible values for **locationType** can be: **CGI**, **SAI**, **RAI**, **TAI**, **ECGI**, **TAI_ECGI**. Depending on the value of **locationType**, the following additional elements may be present:

  – **CI**

  - **ECI**
  - **LAC**
  - **MCC**
  - **MNC**
  - **RAC**
  - **SAC**
  - **TAC**

  For example, if the location type is **CGI**, then the location-type specific parameters that should be set are: **CI**, **LAC**, **MCC**, and **MNC**.

  For more information on the location identification protocol, see the 3GPP specifications: TS 29.060, TS 29.061, and TS 29.274.

- **services** contains one or more **service** elements. The **service** defines the services this subscriber can access. Each service has the following fields:
  - **description**: A description of the service.
  - **endDate**: The date, in YYYYMMDD format, on which the service availability ends for an individual subscriber. Along with the **startDate** value, this value enables you to make services available to subscribers within a specific date range.
  - **id**: The identity of the Policy Controller application described by this **service** element. See *Oracle Communications Service Broker Policy Controller Implementation Guide* for more information.
  - **startDate**: The date, in YYYYMMDD format, on which service availability begins. Along with the **endDate** value, this value enables you to make services available to subscribers within a specific date range.
- **subscriberCategory**: An application-defined service level for a given subscriber, such as gold, silver, and bronze. The Policy Controller rules determine how a category dictates the level of service access for the subscriber.

profileDataExtension Element

Service Broker applications use the **profileDataExtension** element to store custom data elements in the profile. For example, the Policy Controller rule engine can make policy decisions based on dynamically defined subscriber attributes.

The **extensionId** value identifies the Service Broker component that uses the data extension. For the Policy Controller, for example, the **extensionId** value is **pcrf**. Each data extension entry consists of one or more name-value pairs, as specified by the Service Broker component that uses it.

userIdentifier Element

The **userIdentifier** element contains the unique identifier for an end user on a particular network. An end user can belong to multiple networks, for instance, if they own multiple devices used to access different networks. Therefore, a subscriber can have multiple **userIdentifier** elements.

Each **userIdentifier** element consists of the identifier value and the type of the identifier. The type represents the system in which the ID belongs. It may have one of the following values:

- **END_USER_E164**: The subscriber's identity in ITU-T E.164 format, as defined in recommendations E164 and CE164.
- **END_USER_IMSI**: The subscriber's identity in International Mobile Subscriber Identity format, as defined in ITU-T recommendations E212 and CE212.
- **END_USER_SIP_URI**: The subscriber's identity in a SIP network.
- **END_USER_NAI**: The subscriber's identity in a mobile IP Network Address Identifier format.
- **END_USER_PRIVATE**: The subscriber's private identity in a credit-control server.
- **END_USER_GLOBAL_UID**: A unique global user ID generated by Service Broker to identify subscribers internally. Service Broker generates this ID automatically when you create a subscriber profile.

The specific value for each identifier type varies by the network type. For example, the identifier for the URI type should be in the form of a SIP URL, such as **sip:username@example**.

Handling Errors

The Subscriber Provisioning service responds to client request errors with a **ProvisioningException** error response. You can use the information in the response to troubleshoot client application errors.

The **ProvisioningException** response includes the **faultcode** and **faultstring** elements, which indicate the type of error and provide more information about the error. For example:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <S:Fault xmlns:ns3="http://www.w3.org/2003/05/soap-envelope">
         <faultcode>S:Server</faultcode>
         <faultstring>Profile not found : 55555656|END_USER_E164</faultstring>
         <detail>
            <ns2:ProvisioningException
            xmlns:ns2="http://ws.subscriber_store.service.rcc.app.ocsb.oracle/">
               <message>Profile not found : 55555656|END_USER_E164</message>
            </ns2:ProvisioningException>
         </detail>
      </S:Fault>
   </S:Body>
</S:Envelope>
```

This example shows the error response if the subscriber ID passed in the **getSubscriber** request was not found in the Subscriber Store.

If a SOAP request does not contain a required element, the **ProvisioningException** response indicates an invalid input parameter error and identifies the missing element.

For example, a **storeSubscriber** request with an incomplete **userIdentifier** element generates the following error:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <S:Fault xmlns:ns3="http://www.w3.org/2003/05/soap-envelope">
         <faultcode>S:Server</faultcode>
         <faultstring>Invalid input parameters:
                   {userIdentifier=null}</faultstring>
         <detail>
```

```
            <ns2:ProvisioningException xmlns:ns2=
                   "http://ws.subscriber_store.service.rcc.app.ocsb.oracle/">
               <message>Invalid input parameters:{userIdentifier=null}</message>
            </ns2:ProvisioningException>
         </detail>
      </S:Fault>
   </S:Body>
</S:Envelope>
```

As indicated by the **faultstring**, the **userIdentifier** element in the request message is invalid or missing. Similarly, the content of the **ProvisioningException** response for other types of errors provides information on the nature of the error that caused the exception.

6

Subscriber Provisioning API Reference

This chapter provides reference information for the Subscriber Provisioning API, the application programming interface client applications use to manage profiles in the Oracle Communications Service Broker Subscriber Store.

About the Subscriber Provisioning API

You use the Subscriber Store SOAP API to add and manage subscriber profiles in the subscriber store. The Subscriber Provisioning API contains the following operations:

- storeSubscriber
- getSubscriber
- updateSubscriber
- deleteSubscriber

The following sections provide reference information about the operations.

Each operation acts upon a subscriber profile. See "Subscriber Profile Data Model" for information on the contents of the subscriber profile, including its data elements.

storeSubscriber

Creates one or more subscriber profiles in the Subscriber Store based on the data passed in the operation call.

When it receives the request, the Subscriber Provisioning service creates the profile and assigns it an automatically generated user identifier. You can use any of the user identifiers to a **getSubscriber** call to verify the presence of the newly created subscriber profile.

Request Parameters

The request body contains subscriber profile data. Each profile should be enclosed in an **arg0** element.

In the profile data, the **profileDataExtensions** element is optional. All other elements, including all fields of the **globalProfileData**, are required.

See "Subscriber Profile Data Model" for information about the format of a subscriber profile.

Response Parameters

The response contains an empty **storeSubscriberResponse** element.

Example

Example 6–1 storeSubscriber Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
           xmlns:ws="http://ws.subscriber_store.service.rcc.app.ocsb.oracle/">
   <soapenv:Header/>
   <soapenv:Body>
      <ws:storeSubscriber>
         <arg0>
            <globalProfileData>
               <accountState>Active</accountState>
               <accountType>MyGroup</accountType>
               <dateOfBirth>20001112</dateOfBirth>
               <groups>PkGroup</groups>
               <language>english</language>
               <notificationChannel>SMS</notificationChannel>
               <subscriberActivationDate>20110101</subscriberActivationDate>
            </globalProfileData>
            <ifcProfileData>
               <ifc>MyIfc</ifc>
            </ifcProfileData>
            <pcrfProfileData>
               <allowUnknownServices>false</allowUnknownServices>
               <homeZones>
                  <locationType>CGI</locationType>
                  <CI>99</CI>
                  <LAC>86</LAC>
                  <MCC>240</MCC>
                  <MNC>1</MNC>
               </homeZones>
               <services>
                  <description>Service 1</description>
```

```
                  <endDate>20131231</endDate>
                  <id>serviceId</id>
                  <startDate>20120101</startDate>
               </services>
               <subscriberCategory>Silver</subscriberCategory>
            </pcrfProfileData>
            <profileDataExtensions>
               <extensionId>MyExtensionId</extensionId>
               <profileDataExtensions>
                  <name>myName</name>
                  <value>01</value>
               </profileDataExtensions>
            </profileDataExtensions>
            <userIdentifiers>
               <id>15551234567</id>
               <type>END_USER_E164</type>
            </userIdentifiers>
         </arg0>
      </ws:storeSubscriber>
   </soapenv:Body>
</soapenv:Envelope>
```

Example 6–2 Creating Multiple Subscriber Profiles

```
<?xml version="1.0" ?>
   <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
      <S:Body>
         <ns2:storeSubscriber
            xmlns:ns2="http://ws.subscriber_store.service.rcc.app.ocsb.oracle/">
            <arg0>
               <globalProfileData>
                  <accountState>Active</accountState>
                  <groups>MyGroup</groups>
                  <language>english</language>
                  <notificationChannel>SMS</notificationChannel>
               </globalProfileData>
               <ifcProfileData>
                  <ifc>MyIFC</ifc>
               </ifcProfileData>
               <pcrfProfileData>
                  <allowUnknownServices>true</allowUnknownServices>
                  <homeZones>
                     <locationType>CGI</locationType>
                     <CI>99</CI>
                     <LAC>86</LAC>
                     <MCC>240</MCC>
                     <MNC>1</MNC>
                  </homeZones>
                  <services>
                     <description>MyDescription</description>
                     <endDate>MyEndDate</endDate>
                     <id>MyServiceId</id>
                  </services>
                  <subscriberCategory>silver</subscriberCategory>
               </pcrfProfileData>
               <profileDataExtensions>
                  <extensionId>MyExtensionId</extensionId>
                  <profileDataExtensions>
                     <name>MyExtensionEntryName</name>
                     <value>MyExtensionEntryValue</value>
```

```
                  </profileDataExtensions>
               </profileDataExtensions>
               <userIdentifiers>
                  <id>15551234567</id>
                  <type>END_USER_E164</type>
               </userIdentifiers>
            </arg0>
            <arg0>
               <globalProfileData>
                  <accountState>Active</accountState>
                  <groups>MyGroup</groups>
                  <language>english</language>
                  <notificationChannel>SMS</notificationChannel>
               </globalProfileData>
               <ifcProfileData>
                  <ifc>MyIFC</ifc>
               </ifcProfileData>
               <pcrfProfileData>
                  <allowUnknownServices>true</allowUnknownServices>
                  <homeZones>
                     <locationType>CGI</locationType>
                     <CI>99</CI>
                     <LAC>86</LAC>
                     <MCC>240</MCC>
                     <MNC>1</MNC>
                  </homeZones>
                  <services>
                     <description>MyDescription</description>
                     <endDate>20110302</endDate>
                     <id>MyServiceId</id>
                  </services>
                  <subscriberCategory>silver</subscriberCategory>
               </pcrfProfileData>
               <profileDataExtensions>
                  <extensionId>MyExtensionId</extensionId>
                  <profileDataExtensions>
                     <name>MyExtensionEntryName</name>
                     <value>MyExtensionEntryValue</value>
                  </profileDataExtensions>
               </profileDataExtensions>
               <userIdentifiers>
                  <id>15555678923</id>
                  <type>END_USER_E164</type>
               </userIdentifiers>
            </arg0>
         </ns2:storeSubscriber>
      </S:Body>
   </S:Envelope>
```

Example 6–3 Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
          xmlns:ws="http://ws.subscriber_store.service.rcc.app.ocsb.oracle/">
   <soapenv:Header/>
   <soapenv:Body>
      <ws:storeSubscriberResponse/>
   </soapenv:Body>
</soapenv:Envelope>
```

getSubscriber

Returns a single profile of a subscriber identified by the submitted subscriber identifier.

Request Parameters

The request body is made up of an **arg0** element that contains the following information:

- **id**: The identifier of the subscriber for which you want to retrieve a profile.
- **type**: The type of the subscriber identifier.

See "userIdentifier Element" for possible values of the **type** field.

Response Parameters

The response body contains the complete subscriber profile.

See "Subscriber Profile Data Model" for information about the format of the subscriber profile.

Example

Example 6–4 Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
          xmlns:ws="http://ws.subscriber_store.service.rcc.app.ocsb.oracle/">
   <soapenv:Header/>
   <soapenv:Body>
      <ws:getSubscriber>
         <arg0>
            <id>1</id>
            <type>END_USER_E164</type>
         </arg0>
      </ws:getSubscriber>
   </soapenv:Body>
</soapenv:Envelope>
```

Example 6–5 Response

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body>
      <ns2:getSubscriberResponse
            xmlns:ns2="http://ws.subscriber_store.service.rcc.app.ocsb.oracle/">
         <result>
            <globalProfileData>
               <accountState>Active</accountState>
               <accountType>MyGroup</accountType>
               <dateOfBirth>20001112</dateOfBirth>
               <groups>PkGroup</groups>
               <language>english</language>
               <notificationChannel>SMS</notificationChannel>
               <subscriberActivationDate>20110101</subscriberActivationDate>
            </globalProfileData>
            <ifcProfileData>
               <ifc>MyIfc</ifc>
            </ifcProfileData>
```

```
                <pcrfProfileData>
                   <allowUnknownServices>false</allowUnknownServices>
                   <homeZones>
                      <locationType>CGI</locationType>
                      <CI>99</CI>
                      <LAC>86</LAC>
                      <MCC>240</MCC>
                      <MNC>1</MNC>
                   </homeZones>
                   <services>
                      <description>Service 1</description>
                      <endDate>20131231</endDate>
                      <id>serviceId</id>
                      <startDate>20120101</startDate>
                   </services>
                   <subscriberCategory>Silver</subscriberCategory>
                </pcrfProfileData>
                <userIdentifiers>
                   <id>15551234567</id>
                   <type>END_USER_E164</type>
                </userIdentifiers>
                <userIdentifiers>
                   <id>OCSB_GUID:15551234567</id>
                   <type>END_USER_GLOBAL_UID</type>
                </userIdentifiers>
             </result>
          </ns2:getSubscriberResponse>
       </S:Body>
    </S:Envelope>
```

updateSubscriber

Modifies the data for a single subscriber profile. The request must contain the entire profile for the subscriber, including unchanged fields. The Subscriber Provisioning service replaces the contents of the existing profile with the one submitted in the request.

Request Parameters

Request body consists of one or more **arg** elements made up of subscriber profiles.

In the profile data, the **profileDataExtensions** element is optional. All other elements, including all fields of the **globalProfileData**, are required.

See "Subscriber Profile Data Model" for information about the format of a subscriber profile.

Response Parameters

If the operation was successful, the response body contains an empty **updateSubscriberResponse** element.

Example

Example 6–6 Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
           xmlns:ws="http://ws.subscriber_store.service.rcc.app.ocsb.oracle/">
   <soapenv:Header/>
   <soapenv:Body>
      <ws:updateSubscriber>
         <arg0>
            <globalProfileData>
               <accountState>Active</accountState>
               <accountType>MyGroup</accountType>
               <dateOfBirth>20001112</dateOfBirth>
               <groups>PkGroup</groups>
               <language>spanish</language>
               <notificationChannel>SMS</notificationChannel>
               <subscriberActivationDate>20110101</subscriberActivationDate>
            </globalProfileData>
            <ifcProfileData>
               <ifc>MyIfc</ifc>
            </ifcProfileData>
            <pcrfProfileData>
               <allowUnknownServices>false</allowUnknownServices>
               <homeZones>
                  <location>MyLocation</location>
                  <locationType>0</locationType>
               </homeZones>
               <services>
                  <description>Service 1</description>
                  <endDate>20131231</endDate>
                  <id>serviceId</id>
                  <startDate>20120101</startDate>
               </services>
               <subscriberCategory>Silver</subscriberCategory>
            </pcrfProfileData>
```

```
            <profileDataExtensions>
               <extensionId>MyExtensionId</extensionId>
               <profileDataExtensions>
                  <name>myName</name>
                  <value>01</value>
               </profileDataExtensions>
            </profileDataExtensions>
            <userIdentifiers>
               <id>15551234567</id>
               <type>END_USER_E164</type>
            </userIdentifiers>
            <userIdentifiers>
               <id>310150123456789</id>
               <type>END_USER_IMSI</type>
            </userIdentifiers>
         </arg0>
      </ws:updateSubscriber>
   </soapenv:Body>
</soapenv:Envelope>
```

Example 6–7 Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
          xmlns:ws="http://ws.subscriber_store.service.rcc.app.ocsb.oracle/">
   <soapenv:Header/>
   <soapenv:Body>
      <ws:updateSubscriberResponse/>
   </soapenv:Body>
</soapenv:Envelope>
```

deleteSubscriber

Removes a single subscriber profile from the subscriber store.

Request Parameters

Request body parameters are:

- **id**: The identifier of the subscriber profile to be deleted.
- **type**: The type of the subscriber identifier, such as IMSI or E.164. See "userIdentifier Element" for a list of the types.

Response Parameters

If the operation was successful, the response body contains an empty **deleteSubscriberResponse** element.

Example

Example 6–8 Request

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
           xmlns:ws="http://ws.subscriber_store.service.rcc.app.ocsb.oracle/">
   <soapenv:Header/>
   <soapenv:Body>
      <ws:deleteSubscriber>
         <arg0>
               <id>310150123456789</id>
               <type>END_USER_IMSI</type>
         </arg0>
      </ws:deleteSubscriber>
   </soapenv:Body>
</soapenv:Envelope>
```

Example 6–9 Response

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
           xmlns:ws="http://ws.subscriber_store.service.rcc.app.ocsb.oracle/">
   <soapenv:Header/>
   <soapenv:Body>
      <ws:deleteSubscriberResponse/>
   </soapenv:Body>
</soapenv:Envelope>
```