Oracle® Communications Convergent Charging Controller

BRM Charging Driver Technical Guide

Release 6.0.1

April 2017

ORACLE®

Copyright

Contents

About This Document

Scope

This document presents an overview of the integration of Oracle Communications Convergent Charging Controller and Oracle Communications Billing and Revenue Management (BRM) and describes the processes of installing, configuring and administering the Convergent Charging Controller BRM Charging Driver. It also presents several message flow scenarios that describe in detail the interactions that occur for various call charging scenarios. In some cases, this document refers you to existing Convergent Charging Controller and BRM documentation to perform specific steps that have already been described in those documentation sets.

Audience

This guide was written primarily for system administrators and persons installing, configuring and administering the BRM Charging Driver. However, sections of the document may be useful to anyone requiring an introduction to the application.

Prerequisites

This document assumes that you are familiar with both the Convergent Charging Controller system and the BRM application. This document focuses on the implementation and configuration tasks that are required to integrate the two products.

A solid understanding of UNIX and familiarity with IN concepts and with BRM and its system administration are essential prerequisites for safely using the information contained in this technical guide. Attempting to install, remove, configure or otherwise alter the described systems without the appropriate background skills, could cause damage to the system; including temporary or permanent system malfunctions, loss of service, and inability to recover your system.

Although it is not a prerequisite to using this guide, familiarity with the target platform would be an advantage.

This manual describes system tasks that should only be carried out by suitably trained operators.

Related Documents

The following documents are related to this document:

- *Installation Guide*
- *Service Management System User's Guide*
- *Service Management System Technical Guide*
- *Service Logic Execution Environment Technical Guide*
- *Charging Control Services User's Guide*
- *Billing and Revenue Management Concepts*
- *Billing and Revenue Management Developer's Guide*
- *Billing and Revenue Management System Administrator's Guide*
- *Billing and Revenue Management Setting Up Pricing and Rating*
- *Billing and Revenue Management Telco Integration*

Document Conventions

Typographical Conventions

The following terms and typographical conventions are used in the Oracle Communications Convergent Charging Controller documentation.

Formatting Convention	Type of Information
Special Bold	Items you must select, such as names of tabs. Names of database tables and fields.
Italics	Name of a document, chapter, topic or other publication. Emphasis within text.
Button	The name of a button to click or a key to press. **Example:** To close the window, either click **Close**, or press **Esc**.
Key+Key	Key combinations for which the user must press and hold down one key and then press another. Example: **Ctrl+P** or **Alt+F4**.
`Monospace`	Examples of code or standard output.
`Monospace Bold`	Text that you must enter.
variable	Used to indicate variables or text that should be replaced with an actual value.
menu option > menu option >	Used to indicate the cascading menu option to be selected. Example: **Operator Functions > Report Functions**
hypertext link	Used to indicate a hypertext link.

Specialized terms and acronyms are defined in the glossary at the end of this guide.

Chapter 1

About the BRM Charging Driver

Overview

Introduction

This chapter describes the architecture and the main features of the BRM Charging Driver (BCD), which integrates the Oracle Communications Convergent Charging Controller application with the Oracle Communications Billing and Revenue Management (BRM) system.

For architectural overviews and descriptions of the Convergent Charging Controller and BRM systems, see the documentation sets for each of these products. For an architectural overview of Convergent Charging Controller, see *System Administrator's Guide*. For an overview of BRM, see *BRM Concepts*.

In this chapter

This chapter contains the following topics.

Overview of the BRM Charging Driver

About BRM

Oracle Communications Billing and Revenue Management (BRM) is a revenue management system for communications and media service providers. Some of the services and capabilities that BRM provides include:

- Managing customers
- Creating price lists for calculating customer charges
- Recording billable events for chargeable interactions
- Rating usage by measuring events and calculating charges
- Creating bills
- Managing payments and accounts receivable

For a thorough introduction to BRM, Oracle strongly recommends that you first read *BRM Concepts*.

About integrating Convergent Charging Controller and BRM

The BRM Charging Driver is the interface that allows Convergent Charging Controller to integrate and communicate with BRM. In the Convergent Charging Controller configuration, Charging Control Services (CCS) and Advanced Control Services (ACS) run on the Service Logic Controller (SLC) platforms and ACS communicates with various networks through Convergent Charging Controller network interfaces.

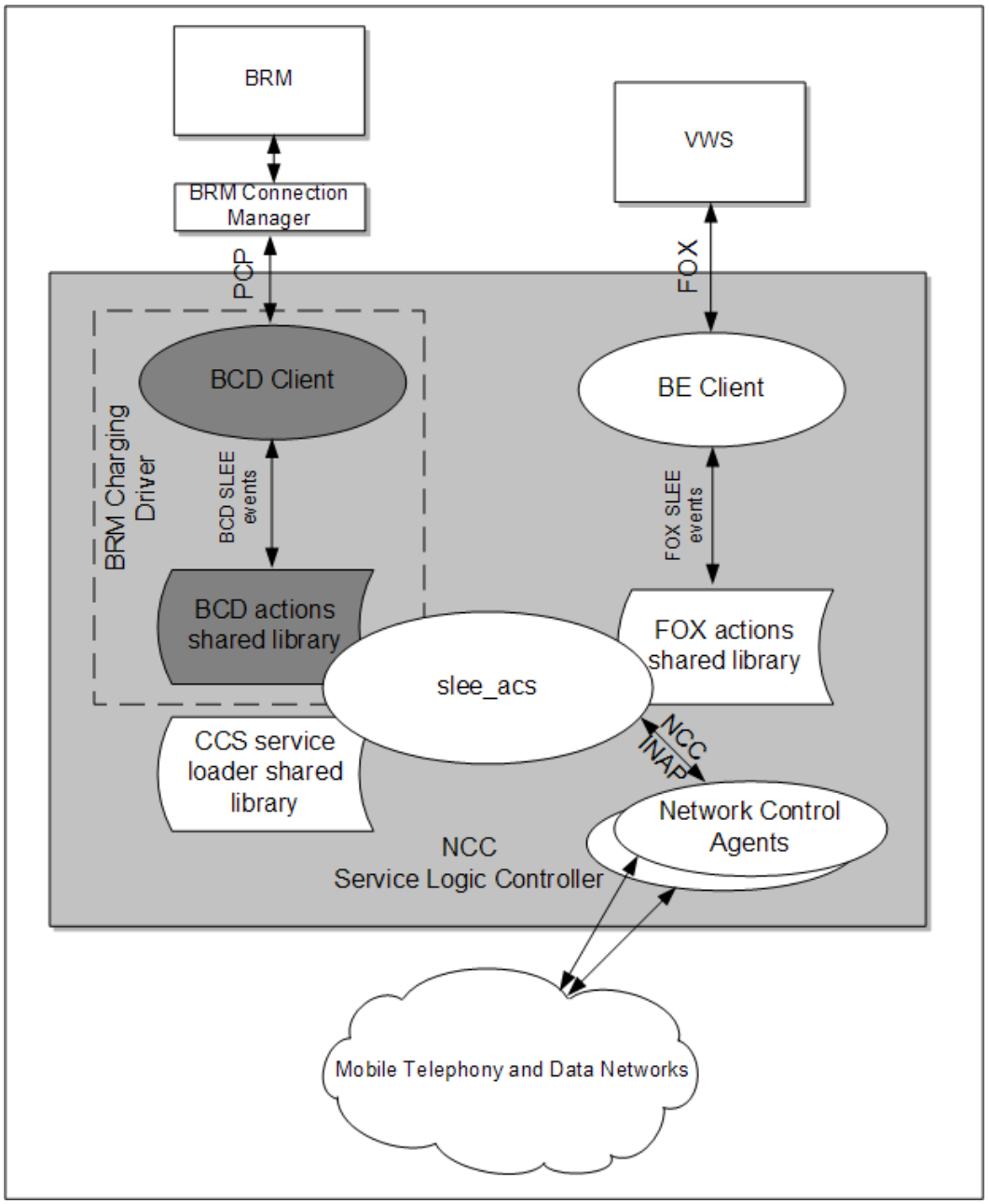
BRM stores the wallet and subscriber data and the vouchers are stored on the Convergent Charging Controller Voucher and Wallet Server (VWS). The CCS software communicates with BRM through the Portal Communications Module (PCM) API.

Some subscriber data is held on Convergent Charging Controller and some on BRM. The Convergent Charging Controller prepaid charging platform does not access subscriber data on BRM other than essential account information, and subscriber data is not replicated from Convergent Charging Controller to BRM. In addition, the integration of BRM with Convergent Charging Controller has the following features:

- Convergent Charging Controller does not perform any accounts receivable operations.
- Balances are held only on BRM.
- BRM is solely responsible for credit limits for prepaid accounts.
- You can top up BRM balances through interaction with Convergent Charging Controller.
- BRM is responsible for any invoices and statements for prepaid accounts.
- BRM manages the pricing catalog and price elements
- BRM defines and applies any recurring charges. All pricing is configured and applied on BRM.
- BRM is responsible for any re-rating.

BRM and Convergent Charging Controller components

The following diagram illustrates the main components of Convergent Charging Controller system that is integrated with BRM. The BRM Charging Driver components that are required to integrate with BRM are the BCD Client and the BCD actions shared library, which are shown in dark shading. The other components are existing components of Convergent Charging Controller and BRM.



The following sections describe the main components of Convergent Charging Controller system that is integrated with BRM:

About BRM Connection Manager

Convergent Charging Controller connects to BRM through the BRM Connection Manager, which runs as a daemon on a dedicated BRM Connection Manager machine. When Convergent Charging Controller requests a connection, a parent Connection Manager process spawns a child process to handle the connection. After that, all communication flows from Convergent Charging Controller to the child Connection Manager.

The Connection Manager uses a connection pool, which is a set of connections that it maintains with an application. When Convergent Charging Controller requests an operation of BRM, the Connection Manager assigns the request to a connection from the pool and uses it to perform the operation. When the operation completes, the connection is returned to the pool.

If an incoming request cannot be assigned a connection immediately, the Connection Manager queues the request to wait for a configurable period of time for a connection to become available. If a connection does not become available during that time, the Connection Manager throws an exception to indicate that the request timed out.

For more information about the BRM Connection Manager, see the section on system architecture in *BRM Concepts*. For information on configuring Convergent Charging Controller to utilize the BRM Connection Manager, see Configuring bcdActionHandler .

About Portal Communications Module API

All Convergent Charging Controller access to BRM and BRM data is done through the BRM Portal Communications Module (PCM) API. Calls to the PCM API are made through a macro interface rather than directly to API functions. The PCM macros pass operations, called *opcodes,* to BRM to perform various operations. An opcode has an identifying name and number that are associated with a particular function that performs the specified operation.

A PCM opcode receives input data and sends output data in the form of field lists (*flists*), which are lists of field name and value pairs. Each opcode requires its input flist to contain the fields required to perform the operation. For example, to request that BRM debit an account balance, Convergent Charging Controller sends the PCM_OP_BILL_DEBIT opcode, including all information in the input flist that is required to debit the account balance.

BRM responds to Convergent Charging Controller operation requests by returning an output flist.

About the session ID

The BRM Charging Driver uses the PIN_FLD_SESSION_ID field in the input flist to specify a sequence number for a PCM opcode request. This enables BRM to detect whether duplicate opcodes are sent within a session.

About PIN_FLD_DIRECTION

The BRM Charging Driver uses the PIN_FLD_DIRECTION field in the input flist to indicate either an originating or terminating trigger for voice calls and SMS messages.

For voice calls, the BRM Charging Driver sets PIN_FLD_DIRECTION in the input flist based on the received `eventTypeBCSM` in the CAMEL `InitialDP` message as follows:

- 0 = Originated (MOC)
- 1 = Terminated (MTC)

For SMS, you can set PIN_FLD_DIRECTION in the input flist through the control plan as follows:

- 0 = Originated (MO SMS)
- 1 = Terminated (MT SMS)

For examples, see the messages in *Usage scenarios* (on page 51).

About BCD Client

The BCD Client is a non-blocking SLEE interface process. You can use multiple BCD Client processes to share the load of BRM Charging Driver operations.

The BCD Client uses the Portal Communications Module (PCM) API to set up connections to the BRM Connection Managers and to send and receive messages over the BRM Portal Communications Protocol (PCP). The BCD Client uses a connection pooling mechanism with load balancing and fail-over to communicate with multiple BRM nodes. Once a connection is established, it is used for multiple operations.

At start-up, the BCD Client reads its configuration information from the `BCD` section of the **eserv.config** file and also reads the user names and passwords from the SCP database on the SLC. It will re-read the configuration on receipt of a SIGHUP signal (hang up signal) and on receipt of a REREAD_CONFIG SLEE management event.

The BCD Client processes events that are passed to it over the SLEE by the slee_acs process as a result of running a control plan. The BCD Client simply takes BRM Charging Driver events and turns them into calls to the PCM API.

About BCD actions shared library

The BCD actions shared library resides on the SLC server and translates requests from CCS feature nodes into PCM operations for BRM and then handles the responses.

CCS feature nodes initiate charging actions by calling methods in the Convergent Charging Controller acsActions API. The API directs these requests to the shared library for the appropriate protocol, based on the control plan's current domain. For the BRM domain, the BRM Charging Driver implements the BCD actions shared library for the PCM API. The BCD actions shared library communicates with the BCD Client by sending SLEE events that contain PCM operations.

BRM Charging Driver reports

The BRM Charging Driver reports on events either initiated or observed by the BCD Client. Reporting is done through the standard Convergent Charging Controller Service Management System (SMS) reporting mechanism.

For information on generating BRM Charging Driver statistics and reports, see *Generating Statistics and Reports* (on page 49).

Other Convergent Charging Controller components

The following Convergent Charging Controller components are also relevant in the integration of Convergent Charging Controller and BRM:

- SLEE and slee_acs

  The Service Logic Execution Environment (SLEE) manages a group of applications that communicate with each other and share resources efficiently. The slee_acs process is the main process of the Advanced Control Services (ACS) software component and it runs the service logic. It is the process that executes control plans and gives instructions to interfaces that communicate with the network, with billing engines such as BRM, or with other entities. It communicates with these interfaces by using the Convergent Charging Controller SLEE API to send and receive SLEE events; thus, the name slee_acs.

  For more information on the slee_acs process, see *Advanced Control Services Technical Guide*.
- The Billing Engine (BE) Client

  The BE Client provides the interface that processes requests received from the slee_acs process and sent to the Convergent Charging Controller Voucher and Wallet Server.
- Voucher and Wallet Server (VWS)

Voucher and Wallet Server (VWS) is the Convergent Charging Controller real-time charging and subscriber account management component. The VWS solution maintains voucher, wallet and reservation details in a database on the Voucher and Wallet Server. The role of VWS is to manage all the billing and charging information associated with call processing. BRM has more advanced charging features that replace the charging features of the (VWS).

- FOX shared library

  The FOX shared library is linked to and run by the slee_acs process, which implements charging related actions by sending SLEE FOX events to the Billing Engine (BE) client and receiving SLEE FOX events in return.

  The BE client and the FOX shared library perform the same functions as the BCD Client and the BCD actions shared library, except that SLEE FOX events replace SLEE BCD events, FOX messages replace PCM operations, and VWS replaces BRM.

- CCS service loader shared library

  The CCS service loader shared library is primarily responsible for loading both the control plan to be run and the subscriber and service provider profiles from the database. The main piece of information it uses to do this is the calling party's number.

- Network control agents

  Network control agents communicate with the Advanced Control Services component using the internal Convergent Charging Controller protocol. In doing so, the control agents translate network protocols such as SIP, MAP, or Diameter into INAP so that a common Convergent Charging Controller service logic is possible, independent of the network protocol.

BRM Charging Driver features

The BRM Charging Driver makes available the following features, or capabilities, when it integrates BRM with Convergent Charging Controller.

Feature	Description
Session Charging	The ability to charge for voice and data sessions using the Convergent Charging Controller Universal Attempt Termination with Billing (UATB) feature node and BRM
Time Units	The ability to charge for sessions using the time unit type
Data Units	The ability to charge for sessions using the data unit type
Funds Exhaustion	Rejection of further quota requests (duration or volume) when BRM indicates that funds are exhausted
Network Ends Session	The UATB feature node will report used units (data or time) to BRM when the network indicates that the session terminated
Cost of Session / Remaining Balance	The ability to report remaining balance and cost of session at the end of a session
Voice Call Cost	The ability to use the Voice Call Cost feature node to request play a message that states the cost of a session
SMS Call Info	The ability to use the SMS Call Info feature node to request BRM to send the cost of a session
Cumulative Balances	The ability to use the Cumulative Balances feature node to play a message that states the balances of an account on BRM
SMS Account Balances	The ability to use the SMS Account Balances feature node to send the balances of an account that resides on BRM
Account State Branch	The ability to use the Account State Branch feature node to branch on the state of an account that resides on BRM
Account Status	The ability in the control plan to play the status and balance of an account on BRM

Feature	Description
Billing Failure Treatment	The ability to specify in the control plan the treatment to use when the UATB feature node cannot contact BRM
Billing Failure Treatment EDRs	When billing failure treatment conditions occur, event detail records (EDRs) are produced and marked with a special tag so you can identify them for post processing
Direct Named Event	The ability to use the Named Event feature node to create a Direct Named Event against BRM
IVR Playing of Redeemed Accounts	Using Interactive Voice Response (IVR) as the input method, the ability to use the Play Voucher Redeem Balances feature node to play a message that tells the caller the amounts recharged to accounts on BRM
Redemption and Recharge of VWS Vouchers	The ability to use the Voucher Redeem and Voucher Recharge feature nodes to redeem and recharge VWS vouchers against BRM accounts
IVR Redemption and Recharge of VWS Vouchers	Using Interactive Voice Response (IVR) interaction as the input method, the ability to use the Voucher Redeem and Voucher Recharge feature nodes to recharge VWS vouchers against BRM accounts
IVR Playing of Redeemed Amounts for VWS Vouchers	Using Interactive Voice Response (IVR) as the input method, the ability to use the Play Voucher Redeem Balances feature node to play a message that tells the caller the amounts recharged to accounts on BRM from VWS vouchers
SMS Recharge of VWS Vouchers	Using SMS (text message) as the input method, the ability to use the Voucher Redeem and Voucher Recharge feature nodes to recharge VWS vouchers against BRM accounts. This does not include the ability to provide information through SMS on amounts recharged
USSD Recharge of VWS Vouchers	Using Unstructured Supplementary Service Data (USSD) as the input method, the ability to use the Voucher Redeem and Voucher Recharge feature nodes to recharge VWS vouchers against BRM accounts. This does not include the ability to provide information through USSD on amounts recharged.
Web Services Recharge of VWS Vouchers	Triggered through Web services, the ability to use the Voucher Redeem and Voucher Recharge feature nodes to recharge VWS vouchers against BRM accounts. This does not include the ability to provide information on the amounts recharged in the Web services response.
Voucher Redeem Failure Records	The ability to create a record against the BRM account in the BRM database every time an attempt to redeem a VWS voucher against that account fails
Reservationless Charging or Refund	The ability to use the DUCR feature node to debit based on a usage amount (duration or volume) without reservation of quota
Post Call Billing	The ability to use the DUCR feature node to charge for voice calls against BRM accounts after the calls have finished
Credit Transfer	The ability to use the Credit Wallet Transfer feature node to perform a credit transfer between two BRM accounts
Convergent Charging Controller Defined Credit Transfer	The ability to use the Credit Wallet Transfer feature node to perform a credit transfer between two BRM accounts using a Named event that is mapped to a deal in BRM and a voucher type defined in the Convergent Charging Controller system

Feature	Description
Rating Guidance	The ability through a feature node to provide guidance for a particular service that is implemented in a control plan so that BRM is able to apply a particular rate or discount based on criteria that is specified in the control plan. For example, a user is calling a Friends and Family number or is in their home zone.
Usage Tracking	The ability to track service usage for BRM accounts using Convergent Charging Controller tracker wallets
Connection Manager Details	The ability to specify details of how Convergent Charging Controller connects to BRM Connection Managers
Connection Manager Addresses	The ability to specify the IP address of each BRM Connection Manager to which Convergent Charging Controller connects
Connection Manager Maximum Connections	The ability to specify the maximum number of connections from each Convergent Charging Controller process to BRM
PCM API	The ability to use the PCM API to communicate with BRM Connection Managers
BRM EDRs	The ability to use the Set Billing Engine EDR feature node to include additional information, such as the charge for a voice or SMS call, in the BRM usage record that is associated with a usage event. The additional information is obtained from the control plan or session context.

Unavailable BRM Charging Driver features

When BRM is integrated with Convergent Charging Controller through the BRM Charging Driver, you cannot use the following Convergent Charging Controller Prepaid Charging feature nodes for subscribers who are charged using a BRM domain:

- Account Activation
- Balance Cascade Override
- Friend Dest Discount
- Periodic Charge State Branching
- Periodic Charge Subscription
- Periodic Charge Transfer
- Play Tariff Plan Announcement
- Scratch Card Recharge
- Scratch Card Recharge Alternate Subscriber
- Set Discount
- Set Tariff Plan Rule
- Select Credit Transfer
- Wallet State Update

The following Prepaid Charging features, or capabilities, are *not available* in a combined Convergent Charging Controller and BRM system:

- Periodic Service/Charge logic, which is the ability to perform logic based on periodic charge subscriptions in BRM
- Balance cascade override
- Service logic derived discounts
- Text modification of mid-call tariff change
- Set Discount

- Set Tariff Plan Rule
- Select Credit Transfer
- Wallet State Update

For information about the Prepaid Charging feature nodes, see *Feature Nodes Reference Guide*.

BRM Integration Summary

Integrating Convergent Charging Controller and BRM

The process of integrating Convergent Charging Controller and BRM consists of the following general steps, which subsequent chapters describe in detail:

1 Installing the BRM Charging Driver.

For information on installing the BRM Charging Driver, see *Installing the BRM Charging Driver* (on page 11).

2 Configuring Convergent Charging Controller for the BRM Charging Driver.

For information on configuring Convergent Charging Controller, see *Configuring Convergent Charging Controller for the BRM Charging Driver* (on page 13).

3 Configuring BRM for the BRM Charging Driver and the specific scenarios that you want to implement.

For information on configuring BRM, see *Configuring BRM for the BRM Charging Driver* (on page 41)

4 Creating Products and Deals.

For information on creating products and deals, see *Creating Products and Deals* (on page 47).

5 Generating statistics and reports.

For information on generating statistics and reports, see *Generating Statistics and Reports* (on page 49).

Chapter 2

Installing the BRM Charging Driver

Overview of Installing the BRM Charging Driver

Introduction

This chapter explains how to install the BRM Charging Driver components.

In this chapter

This chapter contains the following topics.

About Installing the BRM Charging Driver

Overview of Installing BRM Charging Driver

Note: Install 32-bit Portal Development Toolkit 7.5 in **/opt/portal/7.5/PortalDevKit/lib** as a prerequisite to installing Convergent Charging Controller with the BRM Charging Driver.

The Convergent Charging Controller BRM Charging Driver supports BRM versions 7.4 and 7.5, with certification for version 7.5.

The general platform requirements for the BRM Charging Driver components are the same as they are for Convergent Charging Controller. For information on the general platform requirements for Convergent Charging Controller and on installing Convergent Charging Controller, see *Installation Guide*.

Chapter 3

Configuring Convergent Charging Controller for the BRM Charging Driver

About configuring Convergent Charging Controller for the BRM Charging Driver

Introduction

This chapter explains how to configure Convergent Charging Controller for the BRM Charging Driver.

In this chapter

This chapter contains the following topics.

Summary of Convergent Charging Controller Configuration Tasks

Steps to Configure Convergent Charging Controller for the BRM Charging Driver

The following steps summarize what is required to configure Convergent Charging Controller for the BRM Charging Driver. The sections that follow describe how to complete these steps in more detail.

Step	Action
1	Create a BRM domain with a domain type of BCD. You can only create one domain of this type. Decide whether to enable both wallets and vouchers or only wallets.If you enable only wallets, Convergent Charging Controller will use Convergent Charging Controller vouchers to recharge BRM accounts.
2	Configure replication for following SMF database tables, which are used by the BRM Charging Driver: • SMF_NORMALIZATION • SMF_STDEF_BCD
3	Edit the BRM Charging Driver configuration parameters in the **eserv.config** file on the SLC server.
4	(Optional) Set up multiple BCD Client start-up scripts (**bcdBeClient.sh**) if you have a large system and you want to run more processes.

Step	Action
5	Start the SLEE.
6	Create balance types in the Convergent Charging Controller Service Management System to match BRM resource IDs and map Convergent Charging Controller balance types to BRM resource IDs, which are effectively BRM balance types. For more information, see Creating Balance Type Mappings.

Creating the BRM Domain

Procedure to create a BRM domain

Follow these steps to create a BRM domain.

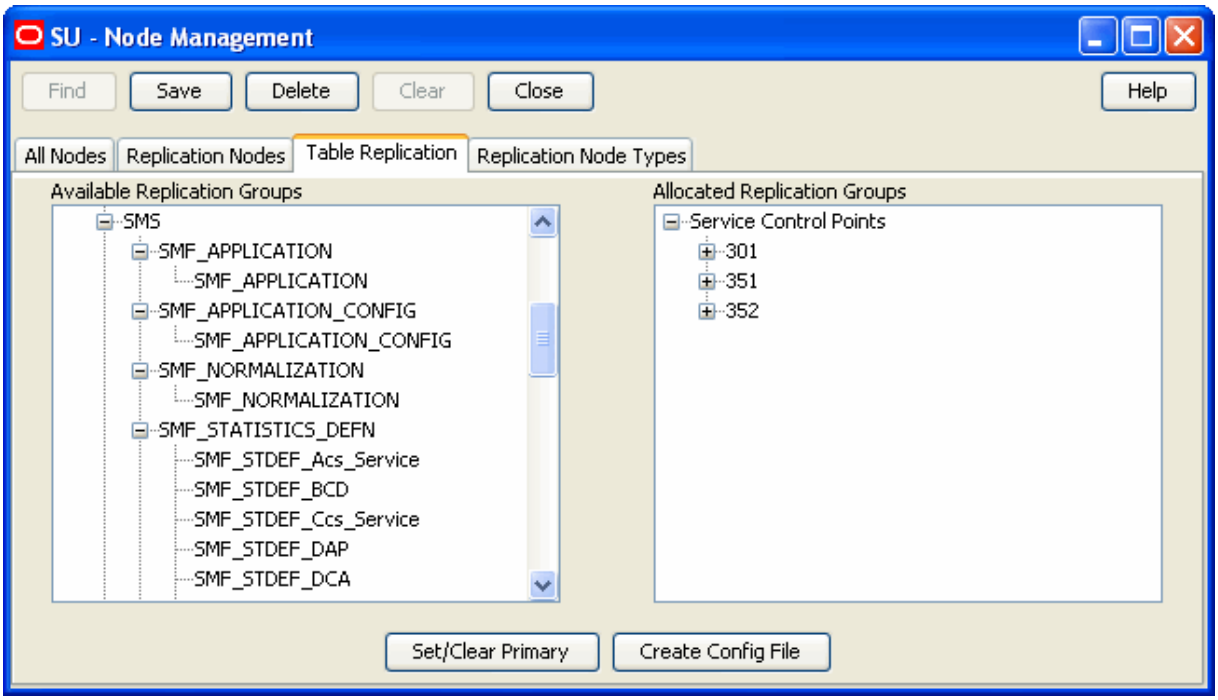
Step	Action
1	Log in to the Service Management System (SMS).
2	From the **Services** menu, select **Prepaid Charging**, then **Service Management**.
3	Select the **Domain** tab to view the current list of Convergent Charging Controller domains.
4	Click **New** to create a new domain.
5	In the **BE** section, enter the following values: In the **Name** field, enter **BRM** From the **Type** list, select **BCD** In the **Maximum Accounts** field, enter the maximum number of accounts in this domain. Select the **Update Username and Password** check box and enter values for **Username** and **Password**, which are used by the BCD Client to connect to the BRM communication managers.
6	(Optional) In the **Manages** section, select the **Charging** and **Voucher Management** options.
7	Click **Save** to save the new domain.

Configuring Replication

Replication for the BRM Charging Driver tables

Using the SMS UI, configure replication for the SMF_NORMALIZATION and SMF_STDEF_BCD replication groups, which are required for the BRM Charging Driver.

To access the replication settings, select the **Operator Functions** menu in the Convergent Charging Controller SMS UI and select **Node Management**. Then select the **Table Replication** tab as shown in the following figure.



For information on using the SMS UI to configure table replication, see *SMS User's Guide*.

About Editing eserv.config Parameters

Editing eserv.config parameters for the BRM Charging Driver

The **eserv.config** file is a shared configuration file from which many Convergent Charging Controller components read their configuration parameters and data. Each component reads those sections of the file that are relevant to its configuration. The section of the **eserv.config** file for the BRM Charging Driver is labeled BCD and the parameters are divided into two structures: `bcdActionHandler` and `bcdBillingClient`.

The **eserv.config** file resides in the **/IN/service_packages/** directory on each of the SLC servers.

The BRM Charging Driver installation process installs two versions of the **eserv.config** file, a default version and an example version. The default version is installed in the following location:

/IN/service_packages/BCD/etc/eserv.config.default

The content of the default version is added to the end of **/IN/service_packages/eserv.config** file. Its BCD section contains only those parameters that are essential to make the BRM Charging Driver functional.

The example version is located here:

/IN/service_packages/BCD/etc/eserv.config.example

The `BCD` section in the example version contains a complete set of Convergent Charging Controller configuration parameters for the BRM Charging Driver. It is provided as a reference when you want to change configuration settings.

For a complete listing of the BCD section of the **eserv.config.example** file, see Appendix A.

The following topics in this section describe `bcdActionHandler` and `bcdBillingClient` structures in the `BCD` section of the **eserv.config** file.

Configuring bcdActionHandler

bcdActionHandler Content

The `bcdActionHandler` structure resides in the `BCD` section of the **eserv.config** file and defines the following:

- *Mapping Convergent Charging Controller sessions to BRM services* (on page 16)
- *Mapping Convergent Charging Controller currency codes to BRM values* (on page 19)
- *Mapping Convergent Charging Controller information to BRM fields* (on page 20)
- *Copying PCM input flist fields to an ACS EDR* (on page 20)
- *Mapping the location number* (on page 21)
- *Specifying custom opcodes* (on page 22)
- *Mapping BRM Piggyback Notifications to Convergent Charging Controller Profile Tags* (on page 22)
- *Configuring additional bcdActionHandler Parameters* (on page 25)

Mapping Convergent Charging Controller sessions to BRM services

The `ServiceProfileTagMapping` array allows you to map `ServiceKey` and `BearerID` combinations, which identify the type of Convergent Charging Controller session, to specific BRM services. If the `ServiceKey` value or the `BearerID` values cannot be found, -1 is used. The default configuration, therefore, is one with `ServiceKey` and `BearerID` values of -1, which specify a basic voice call. The following example illustrates the array entries:

```
        ServiceProfileTagMapping = [
            {
                # Default config. Basic duration measured voice call
                ServiceKey = -1 # default
                BearerID = -1  # default
                ScalingFactor = 1
                BRMField = "QUANTITY" # or BYTES_UPLINK, BYTES_DOWNLINK
                BRMReqMode = "DURATION" # or VOLUME
                BrmServicePoid = "/service/telco/gsm/telephony"
                BrmObjectType = "gsm"
                UsedUnitsCumulative = false
                DefaultUnitType = "QUANTITY" # or UP_BYTES, DOWN_BYTES
            }
            {
                # Specific configuration for data calls.
                ServiceKey = 1
                BearerID = 17
                ScalingFactor = 100000 # Bytes per deci-second (=1Mb/second)
                BRMField = "BYTES_UPLINK"
                BRMReqMode = "VOLUME"
                BrmServicePoid = "/service/telco/gsm/data"
                BrmObjectType = "gsm"
                UsedUnitsCumulative = false
                DefaultUnitType = "UP_BYTES"
            }
    ]
```

The BRM Charging Driver uses this array when it sends an authorize operation to BRM. For information on BRM opcodes, see the *BRM Developer's Reference* and *BRM Telco Integration*.

Convergent Charging Controller always calls a function called `InitialTimeReservation()` even when reserving an amount of data. If Convergent Charging Controller is reserving data, the DCA program (the Diameter interface) will set the bearer capability to tell the slee_acs process that this is a data call. When slee_acs returns a number of seconds, DCA multiplies the seconds by a scaling factor and grants that many bytes of data.

Convergent Charging Controller turns seconds into bytes using a scaling factor when sending an authorize operation to BRM, so that BRM can rate the usage.

The following are the fields in the `ServiceProfileTagMapping` array:

ServiceKey

Syntax: See Example

Description: The service key from the InitialDP that triggered this session. The BRM Charging Driver looks up the `ServiceProfileTagMapping` section of this configuration based on bearer capability and service key. It defaults to the entry for service key -1 if it does not find an entry. InitialDP is a voice operation but the Messaging Manager product (for SMS) and the DCA product (for data) translate their protocols into INAP and send InitialDPs also.

Type: Integer

Optionality: Required

Allowed: 0 to 2147483647

Default: -1

Example: `ServiceKey = 1`

Note: Convergent Charging Controller cannot receive a service key with a value of -1. It is a special value that means 'default' when a service is not found. The meaning of any other service key value is defined by the network operators.

BearerID

Syntax: See Example.

Description: The ITC of the bearer capability provided in the InitialDP and used by Convergent Charging Controller to determine the nature of the session - for example, voice or data.

Type: Integer

Optionality: Required

Allowed: 0 to 31

Default: -1

Example: `BearerID = 17`

Note: Convergent Charging Controller cannot receive a BearerID with a value of -1. It is a special value that means default when a service is not found. This document uses 0 for voice and 17 for data. For the meaning of other values, see the International Telecommunications Union Telecommunication Standardization Section (ITU-T) Recommendation Q.931. The BRM Charging Driver uses only the 5 bits defined for the information transfer capability.

ScalingFactor

Syntax: See Example.

Description: Determines the conversion ratio between the BRM defined unit for QUANTITY, such as bytes, for example, and deciseconds.

Type: Integer
Optionality: Required
Allowed: Any positive integer
Default: None.
Example: `ScalingFactor = 1`

BRMField

Syntax: See Example
Description: This field is required and must be set to QUANTITY, BYTES_UPLINK or BYTES_DOWNLINK. In the PCM_OP_UPDATE_AND_REAUTHORIZE or the PCM_OP_STOP_ACCOUNTING operations, the BRM Charging Driver can put the used units into the QUANTITY, BYTES_UPLINK, or BYTES_DOWNLINK fields. This parameter tells the BRM Charging Driver which field to populate. For more information, see the sections on preparing GSM-specific data in *BRM Telco Integration.*
Type: String
Optionality: Required
Allowed: QUANTITY, BYTES_UPLINK, BYTES_DOWNLINK
Default: None
Example: `BRMField = "QUANTITY"`

BRMReqMode

Syntax: See Example
Description: The unit for which to charge.
Type: String
Optionality: Required
Allowed: DURATION or VOLUME
Default: None
Example: `BRMReqMode = "DURATION"`

BrmServicePoid

Syntax: See Example
Description: This parameter is used in the PIN_FLD_POID parameter.
Type: String
Optionality: Required
Default: None
Example: `BrmServicePoid = "service/telco/gsm/data"`

BrmObjectType

Syntax: See Example
Description: This parameter is used as a suffix on the PIN_FLD_OBJ_TYPE field.
Type: String
Optionality: Required
Default: None
Example: `BrmObjectType = "gsm/ncc"`

`UsedUnitsCumulative`

Syntax: See Example

Description: When slee_acs receives an ApplyChargingReport operation, indicating how many bytes have been used, the value is either cumulative, meaning it is the total bytes used for this session, or it is non-cumulative, meaning it is the number of bytes used this time. A value of `true` in the UsedUnitsCumulative field indicates to the BRM Charging Driver that the used bytes are cumulative. A value of false indicates the used bytes are non-cumulative.

Type: Boolean

Optionality: Required

Default: None

Example: `UsedUnitsCumulative = false`

`DefaultUnitType`

Syntax: See Example

Description: Used to set the UNIT_TYPE parameter when reporting used units to BRM. Specifies which field is used to report usage to BRM in PCM opcodes: PIN_FLD_UP_BYTES, PIN_FLD_DOWN_BYTES, or PIN_FLD_QUANTITY

Type: String

Optionality: Required

Allowed: UP_BYTES, DOWN_BYTES, or QUANTITY

Default: None

Example: `DefaultUnitType = "UP_BYTES"`

Mapping Convergent Charging Controller currency codes to BRM values

The `BrmToNccCurrencyMapping` array maps BRM resource IDs to Convergent Charging Controller currency codes. You must add all currencies that Convergent Charging Controller accesses in BRM to this array. BRM resource IDs are defined in the BRM **pin_currency.h** file, but only those that Convergent Charging Controller uses need to be specified here. For the location of the **pin_currency.h** file, see your BRM documentation.

The following example illustrates the `BrmToNccCurrencyMapping` array:

```
BrmToNccCurrencyMapping = [
    {
        NCCCode = "NZD"
        BRMNum = 554
    }
    {
        NCCCode = "EUR"
        BRMNum = 978
    }
]
```

For more information about currencies and synchronizing monetary transactions between Convergent Charging Controller and BRM, see Creating Balance Type Mappings.

Mapping Convergent Charging Controller information to BRM fields

The `NccToBrmFieldMapping` array maps Convergent Charging Controller event detail record (EDR) information items, as well as other Convergent Charging Controller items, to BRM fields and their associated types. Each field that is defined can be sent in an Convergent Charging Controller information structure within opcodes that are sent to BRM. You trigger the opcodes by using feature nodes such as ChangeEDR or Set Tariff Plan in a control plan.

Note: You must create BRM custom fields to associate with Convergent Charging Controller items and perform the related BRM compilations before attempting to use these custom fields. For information about creating BRM custom fields for Convergent Charging Controller information items, see *About Adding Custom Fields* (on page 41).

Each entry in the `NccToBrmFieldMapping` array must contain the following parameters:

- NCCItem – The name of the Convergent Charging Controller EDR item or other information item
- BRMType – The BRM data type: "STRING, "DECIMAL", "INT", "ENUM", or "TIMESTAMP"
- BRMField – The numeric field ID of the custom field, found in the **pin_flds.h** file on BRM

If an EDR item is not listed here, it will not be copied into a BRM flist.

```
NccToBrmFieldMapping = [
    {
        NCCItem = "TARIFF_PLAN_ID"
        BRMType = "INT"
        BRMField = 10004
    }
    {
        NCCItem = "NUMBER_OF_EVENTS_ID"
        BRMType = "INT"
        BRMField = 10002
    }
    {
        NCCItem = "EXAMPLE"
        BRMType = "STRING"
        BRMField = 10005
    }
    {
        NCCItem = "VOUCHER"
        BRMType = "STRING"
        BRMField = 10007
    }
    {
        NCCItem = "PIN"
        BRMType = "STRING"
        BRMField = 10008
    }
]
```

Copying PCM input flist fields to an ACS EDR

The `BrmFieldToEdrMapping` array is an optional array in the `bcdActionHandler` structure that enables you to specify any non-array field sent to BRM in an input flist to be copied to an Advanced Control Services event detail record (EDR) for subsequent tracking and analysis. If the `BrmFieldToEdrMapping` array is not present in the **eserv.config** file, no flist fields are copied to the EDR.

To specify the flist fields that you want to copy to the EDR, create array items that assign values to the BRMField and EDRItem fields. The BRMField value specifies the integer value of the flist field to be saved and the EDRItem value specifies the name that will be associated with the field in the EDR.

For example, the following entries in `BrmFieldToEdrMapping` will cause PIN_FLD_AUTHORIZATION_ID to be stored in the EDR as AUTHORIZATION_ID=*auth_id_value* and PIN_FLD_SESSION_ID to be stored as SESSION_ID=*sess_id_value*:

```
BrmFieldToEdrMapping = [
    {
        BRMField = 7450
        EDRItem = "AUTHORIZATION_ID"
    }
    {
        BRMField = 3039
        EDRItem = "SESSION_ID"
    }
]
```

This adds entries like the following to the end of the EDR:

```
...|AUTHORIZATION_ID=brmClient-username-2013-3-12-2_session_1187270_0|SESSION_ID=3
```

To obtain the integer values of the flist fields that you want to store, see the **pin_flds.h** file on BRM.

Using output flist fields

The BRM Charging Driver records the following flist fields returned by BRM. The BRM Charging Driver stores these fields in temporary storage profile buffers, using the following profile tags:

Name	Type	Tag
BCD Result	Integer	7995400
BCD Reason	Integer	7995401
BCD Rating Status	Integer	7995402
BCD Stop Accounting Result	Integer	7995403
BCD Stop Accounting Reason	Integer	7995404
BCD Stop Accounting Rating Status	Integer	7995405

For an example of how to use these fields, imagine you have a control plan with a UATB feature node. The Billing Failure Treatment (BFT) branch of the feature node goes to a profile branching feature node that branches on BCD Reason. If the reason is 1011 (EXPIRED_CARD), the control plan goes to a play-announcement feature node to announce to the caller, "Sorry. Your card has expired." For any other value, the plan goes to an end feature node.

In addition, you can put an ACS Change CDR feature node in the control plan to write the temporary storage tags to any CDR fields that you want.

Mapping the location number

The fields in the `LocationNumberMapping` structure control how the location number is retrieved from ACS and sent to BRM. BRMField refers to the BRM field ID of a string field in the GSM_INFO substruct. If this is zero, the location number will not be passed to BRM.

The Primary location profile block and tag specify the primary place to look for the location number in ACS. This defaults to PT_CC_LOCATION_NUMBER. If either parameter is zero, the location number will not be passed to BRM.

The secondary location profile block and tag specify the second place to look for the location number in ACS. This defaults to PT_CC_LOCATION_INFO_LOCATION_NUMBER.

The secondary location is only used if the primary is specified, but no data is found when retrieving data from the profile block. If either of the secondary location parameters are zero, the profile block will not be searched.

```
LocationNumberMapping = {
```

```
        BRMField = 1251     # Or zero to disable sending of Location Number.
        PrimaryLocationNumberProfileBlock = 18      # Call Context
        PrimaryLocationNumberProfileTag = 327692    # PT_CC_LOCATION_NUMBER. Must
        be
                                                    # non-zero if BRMField is non-
        zero.
        SecondaryLocationNumberProfileBlock = 18    # Call Context
        SecondaryLocationNumberProfileTag = 327716 #
        PT_CC_LOCATION_INFO_LOCATION_NUMBER
                                                    # or zero to disable secondary
        choice
    }
```

Specifying custom opcodes

You can specify custom opcodes in place of standard PCM opcodes to allow for the application of custom business rules. To specify custom opcodes, use the `CustomOpCodeMapping` array with `BrmOpCode` and `CustomOpCode` name value pairs, as shown in the following example:

```
    CustomOpCodeMapping = [
        {
            BrmOpCode = 4007
            CustomOpCode = 11007
        }
        {
            BrmOpCode = 4026
            CustomOpCode = 11026
        }
    ]
```

Note: Only the operation code is changed. The flists that are sent to and received from BRM will be exactly the same as they would be for the standard opcodes.

Mapping BRM Piggyback Notifications to Convergent Charging Controller Profile Tags

The `InSessionNotificationMapping` section allows you to map the data received in the BRM PIGGYBACK_NOTIFICATIONS array fields to Convergent Charging Controller profile tags to be used in conjunction with an in-session control plan trigger or to be processed when the session has ended (for IVR channels).

This section is optional; if it is not present then In-Session Notifications will be fully disabled.

Sub-sections and individual items are optional; omitting a section or tag will result in that section/item not being copied into the Convergent Charging Controller profile tag data to be used by In-Session Notifications. Omitting a section description will also result in the data in that section not being copied.

For more information, see Using In-Session Notifications with BRM and BRM documentation on in-session notifications in *Oracle Communications Billing and Revenue Management Telco Integration*.

The following example illustrates the section entries:

```
    InSessionNotificationMapping = {
        ProfileBlock = 17    # NCC profile block to populate. Default = 17
        (TEMPORARY STORAGE)

        Language = {
            Description = "Preferred Language"   # Text to match in
            PIGGYBACK_NOTIFICATIONS
            ProfileTag = 37             # NCC profile tag to use (PT_LANGUAGE)
        }

        Channel = {
```

```
    Description = "Preferred Channel"    # Text to match in
    PIGGYBACK_NOTIFICATIONS
    ProfileTag = 1312050        # NCC profile tag to use
    (PT_ISN_PREF_CHANNEL)
    InSessionTrigger = [ "Email", "SMS" ] # Which channels require in-
    session trigger
}

Time = {
    Description = "Preferred Time"    # Text to match in
    PIGGYBACK_NOTIFICATIONS
    ProfileTag = 1312051        # NCC profile tag to use
    (PT_ISN_PREF_CHANNEL)
}

CreditThreshold = {
    Description = "Credit Threshold Breach"    # Text to match in
    PIGGYBACK_NOTIFICATIONS
    ProfileTag = 1312052        # NCC profile tag to use
    (PT_ISN_CT_BALANCE)
    BalanceTypeTag = 1312054    # NCC profile subtag for balance type ID
    (PT_ISN_CT_BAL_TYPE)
    BalanceNameTag = 1312055    # NCC profile subtag for balance type
    name (PT_ISN_CT_BAL_NAME)
    AmountTag = 1312053         # NCC profile subtag for balance amount
    (PT_ISN_CT_BAL_AMOUNT)
    CurrentBalanceTag = 1312056 # NCC profile subtag for current balance
    (PT_ISN_CT_BAL_CURRENT_BAL)
    GroupObjTag = 1312057       # NCC profile subtag for group object
    (PT_ISN_CT_BAL_GROUP_OBJ)
    PercentTag = 1312058        # NCC profile subtag for percent
    (PT_ISN_CT_BAL_PERCENT)
    SourceObjTag = 1312059      # NCC profile subtag for source object
    (PT_ISN_CT_BAL_SOURCE_OBJ)
    AlertTypeTag = 1312060      # NCC profile subtag for alert type
    (PT_ISN_CT_BAL_ALERT_TYPE)
    ReasonTag = 1312061         # NCC profile subtag for reason
    (PT_ISN_CT_BAL_REASON)
    CreditFloorTag = 1312062    # NCC profile subtag for credit floor
    (PT_ISN_CT_BAL_CREDIT_FLOOR)
    CreditLimitTag = 1312063    # NCC profile subtag for credit limit
    (PT_ISN_CT_BAL_CREDIT_LIMIT)
    CreditThresholdTag = 1312064   # NCC profile subtag for percent
    threshold (PT_ISN_CT_BAL_CREDIT_THRESH)
    CreditThresholdFixedTag = 1312065   # NCC profile subtag for fixed
    threshold (PT_ISN_CT_BAL_CREDIT_THRESH_FIXED)
    BalanceUnitNameTag = 1312078    # NCC profile subtag for balance unit
    name (PT_ISN_CT_BAL_UNIT_NAME)
}

SubscriptionExpiry = {
    Description = "Subscription Expired"    # Text to match in
    PIGGYBACK_NOTIFICATIONS
    ProfileTag = 1312066        # NCC profile tag to use
    (PT_ISN_SUB_EXPIRY)
    ExpiryDateTag = 1312067     # NCC profile subtag for expiry date
    (PT_ISN_SUB_EXPIRY_DATE)
}

StreamingThreshold = {
    Description = "Streaming Threshold reached"    # Text to match in
    PIGGYBACK_NOTIFICATIONS
```

```
            ProfileTag = 1312068        # NCC profile tag to use
            (PT_ISN_STREAM_THRESH)
            CurrentBalanceTag = 1312069 # NCC profile subtag for current balance
            (PT_ISN_STREAM_THRESH_CURRENT_BAL)
        }

        Balance = {
            ProfileTag = 1312070        # NCC profile tag to use (PT_ISN_BALANCE)
            BalanceTypeTag = 1312076    # NCC profile subtag for balance type ID
            (PT_ISN_BALANCE_TYPE)
            BalanceNameTag = 1312077    # NCC profile subtag for balance type
            name (PT_ISN_BALANCE_NAME)
            AmountTag = 1312071         # NCC profile subtag for amount
            (PT_ISN_BALANCE_AMOUNT)
            AvailLimitTag = 1312072     # NCC profile subtag for amount
            (PT_ISN_BALANCE_AVAIL_LIMIT)
            BalanceUnitNameTag = 1312079    # NCC profile subtag for balance unit
            name (PT_ISN_BALANCE_UNIT_NAME)
        }

        Status = {
            RatingStatusTag = 1312073   # NCC profile subtag for rating status
            (PT_ISN_RATING_STATUS)
            LifecycleStateTag = 1312074 # NCC profile subtag for lifecycle state
            (PT_ISN_LIFECYCLE_STATE)
            FailureReasonTag = 1312075  # NCC profile subtag for failure reason
            (PT_ISN_FAILURE_REASON)
        }
    } # End of InSessionNotificationMapping
```

The following are the fields in the `InSessionNotificationMapping` array:

ProfileBlock

The `ProfileBlock` field takes an integer that specifies the Convergent Charging Controller profile block to populate. The default value is 17, which specifies that the profile block is temporary storage.

Language

The `Language` array describes the subscriber's preferred language, such as English or Spanish, for example. The BCD actions library matches the preferred language to a language entry in ACS_LANGUAGE and stores the associated Convergent Charging Controller language ID in the profile tag.

Channel

The `Channel` array describes the subscriber's preferred channel for notifications. The preferred channel could be SMS, Email, or IVR, for example. The BCD actions library stores the preferred channel and compares it against the inSessionTrigger list to see if the selected channel requires a real-time control plan trigger to be armed for the notifications.

Time

The `Time` array contains the subscriber's preferred notification time, specified as a Coordinated Universal Time (UTC) timestamp in string form with three extra digits for milliseconds. The BCD actions library converts the timestamp to a profile date value and stores it in the configured profile tag.

CreditThreshold

The CreditThreshold array contains an entry for each balance that is breaching a credit threshold. A threshold breach can be either a breach up, which is an increased usage of credit or prepaid funds, or a breach down, which is a decreased usage of credit or prepaid funds following a payment or topup. Each balance entry contains fields that are stored in profile tags within a profile array.

Subscription Expiry

The `SubscriptionExpiry` array stores the timestamp that specifies when the subscription defined by the BRM lifecycle expires. The timestamp is stored as a profile Date.

StreamingThreshold

The `StreamingThreshold` array contains the current balance value when the notification was triggered. For postpaid balances, this is a positive number and it represents the currently used credit, including the reserved amount for the current call.

Note: Convergent Charging Controller represents postpaid balances as negative and adds a minus sign. For example the value 913.67 received from BRM for a balance with scaling factor set to 100 is conveyed to the Convergent Charging Controller control plan as -91367

For prepaid balances, this is a negative number that represents the currently available funds after taking into account the amount reserved for the current call.

Note: Convergent Charging Controller represents prepaid balances as positive and removes the minus sign. For example the value -45.88 received from BRM for a balance with scaling factor set to 100 is conveyed to the Convergent Charging Controller control plan as 4588

Balances

The Balances array contains an entry for each subscriber balance. Each balance entry contains an ID, the reserved amount delta for the call, and the remaining unreserved funds left available.

Status

The `Status` array contains statuses that indicate why a call was denied, including the result of the rating operation, the reason for an authorization failure, and the lifecycle state, which is included only if lifecycle management is enabled in BRM.

Configuring additional bcdActionHandler Parameters

The `bcdActionHandler` structure also contains the following additional parameters. For examples of these parameters, please see the sample **eserv.config** file in Appendix A.

`BrmBadPinEdrActive`

Syntax:	See Example
Description:	Specifies whether to trigger sending PCM_OP_ACTIVITY to BRM to request BRM to produce an event detail record (EDR) when the subscriber attempts to redeem a voucher using an invalid voucher number or PIN.
Type:	Boolean
Optionality:	Optional
Allowed:	true and false
Default:	false
Example:	`BrmBadPinEdrActive = true`

BrmBadPinResourceId

Syntax: `parameter = value`

Description: The CreateEDR action always sends the voucher number and PIN number that were used for a failed attempt to redeem a voucher. If BrmBadPinResourceId is not zero, this resource ID will be used as a counter of bad PIN attempts for this account and it will be incremented.

Type: Integer

Optionality: Optional

Allowed: See Description

Default: 0

Example: `BrmBadPinResourceId = 1000011`

BrmEdrObjectType

Syntax: See Example

Description: Specifies the BRM object type to use when sending PCM_OP_ACTIVITY to BRM to request BRM to produce an event detail record (EDR) when the subscriber attempts to redeem a voucher using an invalid voucher number or PIN.

Type: String

Optionality: Optional

Allowed: A valid BRM object type

Default: "/voucher"

Example: `BrmEdrObjectType = "/voucher"`

cacheTimeout

Syntax: See Example

Description: Specifies in seconds the maximum age of cached named event and balance type map data. If the data is older than the number of seconds specified, the cache will be refreshed when the data is needed.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: Any positive integer

Default: 60

Example: `cacheTimeout = 90`

clientIDString

Syntax: See Example

Description: Unique string that identifies a client that will be accessing a BRM server. Prevents multiple clients from accessing BRM with the same authentication IDs.

Type: String

Optionality: Optional

Allowed: See Description

Default: Defaults to value of hostname if not specified

Example: `clientIDString = "client1"`

loggedNotificationPeriod

Syntax: See Example

Description: Specifies in seconds how often to announce the number of message parse errors.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: Any positive integer

Default: 30

Example: `loggedNotificationPeriod = 300`

lowCreditBufferTime

Syntax: See Example

Description: Specifies the number of seconds to hold back as the low credit buffer. For example, if this is set to 10 seconds, the caller will hear a beep 10 seconds before funds expire and the call terminates.

Type: Integer

Optionality: Optional

Allowed: Any positive integer

Default: 10

Example: `lowCreditBufferTime = 10`

NccInfoFieldDummyEntry

Syntax: See Example

Description: Specifies the BRM field ID of a string field that is configured to be present in the PIN_FLD_NCC_INFO substruct. The field is set to "Present" by the action handler. It is required because PIN_FLD_NCC_INFO must be present in the flist that is passed to opcodes. It is not valid to have an empty substruct; thus the dummy value.

Type: Integer

Optionality: Required

Allowed: See Description

Default: 10001

Example: `NccInfoFieldDummyEntry = 10001`

NccInfoFieldNumber

Syntax: See Example

Description: Specifies the BRM field ID of the flist substruct field under which all the Convergent Charging Controller specific fields get added. This will normally be the numeric value associated with PIN_FLD_NCC_INFO when the BRM instance was customized.

Type: Integer

Optionality: Required

Allowed: See Description

Default: 10000

Example: `NccInfoFieldNumber = 10000`

poidPrefix

Syntax: See Example

Description: Specifies the first part of the POID string to be used by named events. The event class is appended to it to form the complete POID name. For example, if the poidPrefix is "/service/telco" and the eventClass name is "/gsm/sms", the complete POID name would be "/service/telco/gsm/sms".

When you define a product in the BRM Pricing Center, you specify the BRM object to which it applies. The POID (Portal Object Identifier) is the name of this object and determines which product is used for rating and charging.

Type: String

Optionality: Required

Allowed: See Description

Default: "/service/telco"

Example:
```
poidPrefix = "/service/telco"
```

roundingScheme

Syntax: See Example

Description: Specifies the rounding scheme for rounding sub-second durations into seconds. The allowable values are:

1 = floor (always round down)

2 = ceiling (always round up)

3 = nearest (>= .5 rounds up; < than .5 rounds down)

Type: Integer

Optionality: Optional

Allowed: 1, 2, or 3

Default: 3

Example:
```
roundingScheme = 3
```

serviceDomainInterfaceName

Syntax: See Example

Description: Specifies the interface name of the BCD Client, which is defined in **SLEE.cfg**. If you create multiple BCD Client start-up scripts, serviceDomainInterfaceName specifies the prefix of the interface name. For example, for the following INTERFACE parameters in **SLEE.cfg**, which are abbreviated here:

```
INTERFACE=bcdBeClient1 bcdBeClient1.sh ...
INTERFACE=bcdBeClient2 bcdBeClient2.sh ...
```

the value of serviceDomainInterfaceName would be "bcdBeClient". For more information, see *Configuring the SLEE to Run New BCD Client Startup Scripts* (on page 36).

Type: String

Optionality: Required

Allowed: See Description

Default: None

Example:
```
serviceDomainInterfaceName = "bcdBeClient"
```

voucherPinLength

Syntax: See Example

Description: Specifies the length of the voucher PIN.

Type:	Integer
Optionality:	Optional
Allowed:	Positive integer
Default:	4
Example:	`VoucherPinLength = 4`

Configuring bcdBillingClient

bcdBillingClient content

The `BCD` section of the **eserv.config** file also contains the `bcdBillingClient` structure, which defines the following:

- *Mapping opcodes* (on page 29)
- *Configuring BRM connections* (on page 30)
- *Specifying operation timeouts* (on page 29)
- *Configuring additional bcdBillingClient parameters* (on page 31)

Mapping opcodes

The `opCodeMapping` section of the `bcdBillingClient` structure specifies the following opcode mapping definitions to associate opcode integers with operation macros that Convergent Charging Controller passes to BRM to perform authorization and accounting operations.

Note: You should not need to change this section unless you specify custom opcodes. In that case, you should add the custom opcodes to the `opCodeMapping` array for the sake of meaningful debug output. See *Specifying custom opcodes* (on page 22) for more information:

```
        opCodeMapping = [
            {operation = "PCM_OP_BAL_GET_BALANCE" , opCode = 3701 }
            {operation = "PCM_OP_CUST_MODIFY_CUSTOMER" , opCode = 64 }
            {operation = "PCM_OP_PYMT_TOPUP" , opCode = 3726 }
            {operation = "PCM_OP_SEARCH" , opCode = 7 }
            {operation = "PCM_OP_SUBSCRIPTION_PURCHASE_DEAL" , opCode = 108 }
            {operation = "PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS" , opCode = 81 }
            {operation = "PCM_OP_TRANS_ABORT" , opCode = 13 }
            {operation = "PCM_OP_TRANS_OPEN" , opCode = 12 }
            {operation = "PCM_OP_WRITE_FLDS" , opCode = 5 }
            {operation = "PCM_OP_READ_FLDS" , opCode = 4 }
            {operation = "PCM_OP_ACT_ACTIVITY" , opCode = 151 }
            {operation = "PCM_OP_CUST_POL_GET_DEALS" , opCode = 278 }
            {operation = "PCM_OP_BILL_DEBIT" , opCode = 105 }
]
```

For information on BRM opcodes, see the *BRM Developer's Reference* and *BRM Telco Integration*.

Specifying operation timeouts

The `OperationTimeouts` array specifies timeout values for the operations in the `opCodeMapping` array. If the timeout value for an operation is too large, it will increase the delay when a failed connection switches from a failed Connection Manager to a working one. If the timeout value is too small, the operation can time out unnecessarily. Also, some operations will take longer than others. The ideal timeout values will vary from site to site based on network speeds, load, and the speed of the BRM servers. For more information, see the *BRM System Administrator's Guide*.

The `defaultOperationTimeout` value specifies a default timeout value for any operation that does not specify a timeout value in `OperationTimeouts`.

The following example shows the initial values of `defaultOperationTimeout` and the `OperationTimeouts` array, as they are provided in the **eserv.config** file:

```
# Default operation timeout to use if a specific opcode does not have an
# entry in the OperationTimeouts array
# Default = 250 milliseconds

        defaultOperationTimeout = 250
        OperationTimeouts = [
            {operation = "PCM_OP_BAL_GET_BALANCE" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_CUST_MODIFY_CUSTOMER" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_PYMT_TOPUP" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_SEARCH" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_SUBSCRIPTION_PURCHASE_DEAL" , timeoutMilliseconds =
            250 }
            {operation = "PCM_OP_TRANS_ABORT" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_TRANS_OPEN" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_WRITE_FLDS" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_ACT_ACTIVITY" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_READ_FLDS" , timeoutMilliseconds = 250 }
        ]
```

Configuring BRM connections

You must also define entries in the BCD section of the **eserv.config** file to configure connections to BRM Connection Managers. You do this by adding or modifying items in the `cmPointers` array in the `ConnectionManager` structure. The `ip` parameter specifies the IP address of a specific Connection Manager machine. If you supply multiple IP addresses in the `cmPointers` array, the BCD Client process selects the actual machine and port to use for a particular connection.

At start up, the BCD Client tries to establish all of the connections specified in the `cmPointers` array, for example 250 to the first Connection Manager, 500 to the second one, 250 to the third one, and so on.

If, for any reason, the BCD Client does not have a full complement of connections to any Connection Manager, the BCD Client tries to re-establish the connections as soon as possible. If an attempt to connect to a Connection Manager fails, the BCD Client process will not attempt to re-connect to that CM for a number of seconds equal to the value specified by the recoverCmPtrSeconds parameter. The attempt to establish the connection will time out after the number of milliseconds that is equal to the value specified by the contextOpenTimeoutMilliseconds parameter.

At any given time, some of the connections will be idle and some will be busy, a busy connection being one that is waiting for a response to an operation that the BCD Client has sent. When the BCD Client wants to choose a connection to send another operation it chooses the Connection Manager with the lowest ratio of busy connections to total connections.

The following example illustrates the definition of the `ConnectionManager` structure:

```
        ConnectionManager = {
            database = 1
            enableTLS = 1
            cipherList = "SSL_RSA_WITH_AES_128_CBC_SHA"
            TLSWalletDirectory = "/IN/service_packages/BCD/wallet"
            service = "/service/pcm_client"
            cmPointers = [
                { cmPtr = "ip 192.168.111.111 12010", poolSize = 15}
                { cmPtr = "ip 192.168.111.112 12010", poolSize = 50}
                { cmPtr = "ip 192.168.111.111 12011", poolSize = 15}
                { cmPtr = "ip 192.168.111.112 12011", poolSize = 15}
            ]
```

```
}
```

The `database` and `service` values should not be changed. Convergent Charging Controller passes these values to the BRM Connection Manager when they connect.

The `enableTLS` parameter sets whether or not Transport Layer Security (TLS) is enabled in the SSL context options when the brmBeClient connects to the CM. To enable TLS, specify 1, which is the default. To disable TLS, specify 0. Disable TLS for connections to ASPs that do not support TLS or that do not fully support SSL secure renegotiation.

The `cipherList` parameter is a string of ciphers to use for the TLS connection. The default cipher is "`SSL_RSA_WITH_AES_128_CBC_SHA`". The list may contain more than one cipher name, separated by commas. If `enableTLS` is set to 0, `cipherList` is not used.

The `TLSWalletDirectory` parameter is the directory that contains the Oracle wallet. The Oracle wallet contains the credentials (certificates) that are used to authenticate the CM when brmBeClient connects. You create the wallet and save the server certificate of the CM in the wallet. For instructions about creating the wallet, see *BRM System Administrator's Guide*.

The `poolSize` value specifies to the BCD client the number of connections to create with the Connection Manager specified by the IP address in the `ip` parameter. The Connection Manager accepts connections until it has too many or the BCD Client reaches the limit specified by the `poolSize` parameter. You need one connection for each simultaneous transaction. For example, if machine 192.168.111.112 is twice as fast as machine 192.168.111.111, you might want to set the value of the `poolSize` parameter for 192.168.111.112 to be twice as large.

Configuring additional bcdBillingClient parameters

The `bcdBillingClient` structure also contains the following additional parameters. For examples of these parameters, please see the sample **eserv.config** file in Appendix A.

`contextOpenTimeoutMilliseconds`

Syntax:	See Example
Description:	Specifies the length of time in milliseconds after which the BCD Client will stop trying to establish a context (connection) with a given BRM Connection Manager. If the connection is not established within this period, the connection attempt is abandoned and the BCD Client can try to establish a connection with the next Connection Manager.
Type:	Integer
Optionality:	Optional (default used if not set).
Allowed:	
Default:	5000
Example:	`contextOpenTimeoutMilliseconds = 5000`

`defaultOperationTimeout`

Syntax:	`paraMeter = value`
Description:	Used if a specific opcode does not have an entry in the `OperationTimeouts` array in the **eserv.config** file. Default is 250 milliseconds. For a description of the `OperationTimeouts` array, see *Specifying Operation Timeouts* (on page 29).
Type:	Integer
Optionality:	Optional (default used if not set).
Allowed:	See Description
Default:	250

Example: `defaultOperationTimeout = 250`

latencyStatisticsInterval

Syntax: See Example

Description: A positive integer value causes latency measurements to be logged in the BCD Client log under the bcdLatency debug section. This parameter specifies in seconds the interval at which measurements are logged. A value of 0 disables logging of latency measurements.

Type: Integer

Optionality: Optional (default used if not set)

Allowed: See Description.

Default: 300

Example: `latencyStatisticsInterval = 240`

maxContextIdleTimeSeconds

Syntax: See Example

Description: The number of seconds to leave an idle connection open before closing it.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: Positive integer

Default: 10

Example: `maxContextIdleTimeSeconds = 10`

maxOutstandingRequests

Syntax: See Example

Description: The maximum number of outstanding events for each BCD Client process. Events will not be sent to a BCD Client with more than this number of events outstanding.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: Any positive integer

Default: 1000

Example: `maxOutstandingRequests = 1000`

maxPollMilliseconds

Syntax: See Example

Description: The maximum number of milliseconds to elapse before checking SLEE events.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: Any positive integer

Default: 50

Example: `maxPollMilliseconds = 1`

maxSelectMicroseconds

Syntax: See Example

Description: The maximum number of microseconds to wait for PCM messages before checking for SLEE events. Oracle recommends that you leave this setting at the default value of 50.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: See Description

Default: 50

Example:
```
maxSelectMicroseconds = 50
```

maxTries

Syntax: See Example

Description: Specifies the maximum number of attempts, including the first, to send an operation to BRM.

Type: Integer

Optionality: Optional (default used if not set)

Allowed: A positive integer

Default: 3

Example:
```
maxTries = 3
```

recordCMIPAddressInStats

Syntax: See Example

Description: Specifies whether to include the IP address of the destination communications manager in statistics.

Type: Boolean

Optionality: Optional (default used if not set).

Allowed: true, false

Default: false

Example:
```
recordCMIPAdressInStats = false
```

recordOpcodeInStats

Syntax: See Example

Description: Specifies whether to include opcodes in statistics.

Type: Boolean

Optionality: Optional (default used if not set).

Allowed: true, false

Default: false

Example:
```
recordOpcodeInStats = false
```

recordPortInStats

Syntax: See Example

Description: Specifies whether to include the port number of the destination communications manager in statistics.

Type: Boolean

Optionality: Optional (default used if not set).
Allowed: true, false
Default: false
Example: `recordPortInStats = false`

recoverCmPtrSeconds

Syntax: See Example
Description: The number of seconds for which any IP address and port combination will be marked as not working. After this number of seconds elapses, the connection will be tried again for reuse.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Any positive integer
Default: 5
Example: `recoverCmPtrSeconds = 5`

oracleUserAndPassword

Syntax: See example
Description: The user credentials that BCD Client uses to connect to the local or remote SLC database.
Type: String
Optionality: Optional (default used if not set)
Allowed:
Default: /@SCP
Example: `oracleUserAndPassword = "/@SCP"`

Configuring Calls, Events, and Vouchers

Configuring Convergent Charging Controller for voice calls

Typically, voice calls are controlled in Convergent Charging Controller by the Universal Attempt Termination with Billing (UATB) feature node in the control plan. To charge a BRM account for a voice call, do one of the following:

- Select the BRM domain, which is the BCD domain type, when configuring the subscriber's wallet.
- Use a Set Active Domain feature node in the control plan with Wallet enabled for the BCD domain type.

To enable voice calls, you must create a voice call entry in the bcdActionHandler.ServiceProfileTagMapping list in the **eserv.config** file on the SLC The following example shows a sample entry:

```
ServiceProfileTagMapping = [
...
    {
        # Prepaid voice
        ServiceKey = 1 # Defined in triggering rules on the HLR
        BearerID = 0 # Information Transfer Capability 0 = Speech per Q.931
        ScalingFactor = 1
        BRMField = "QUANTITY"
        BRMReqMode = "DURATION"
        BrmServicePoid = "/service/telco/gsm/telephony"
```

```
            BrmObjectType = "gsm/ncc"
            UsedUnitsCumulative = false
            DefaultUnitType = "QUANTITY"
        }
```

For descriptions of the `ServiceProfileTagMapping` parameters in the **eserv.config** file, see *Mapping Convergent Charging Controller Sessions to BRM Services* (on page 16).

Configuring Convergent Charging Controller for data calls

Typically, data calls are controlled in Convergent Charging Controller by the Universal Attempt Termination with Billing (UATB) feature node in the control plan. To charge a BRM account for a data call, do one of the following:

- Select the BRM domain, which is the BCD domain type, when configuring the subscriber's wallet.
- Use a Set Active Domain feature node in the control plan with Wallet enabled for the BCD domain type.

To enable data calls, you must create a data call entry in the bcdActionHandler.ServiceProfileTagMapping list in the **eserv.config** file on the SLC. The following example shows a sample entry for a data call:

```
ServiceProfileTagMapping = [
...
    {
        # Configuration for Data Calls
        ServiceKey = 1
        BearerID = 17
        ScalingFactor = 100000  # Bytes per deci-second (= 1Mb per second)
        BRMField = "BYTES_UPLINK"
        BRMReqMode = "VOLUME"
        BrmServicePoid = "/service/telco/gsm/data"
        BrmObjectType = "gsm"
        UsedUnitsCumulative = false
        DefaultUnitType = "UP_BYTES"
    }
```

For descriptions of the `ServiceProfileTagMapping` parameters in the **eserv.config** file, see *Mapping Convergent Charging Controller Sessions to BRM Services* (on page 16).

Modifying the BCD Client Startup Script

Procedure to modify the script

When you install the BRM Charging Driver packages, the BCD Client start-up script (**bcdBeClient.sh**) is added in **/IN/service_packages/BCD/bin**.

You might want to create multiple copies of this script, for example, **bcdBeClient1.sh** and **bcdBeClient2.sh**, to improve performance by allowing each **bcdBeClient** process to run on a separate CPU or thread. If you do so, you must edit each **bcdBeClient*n*.sh** file to specify the correct names of the startup script and its log file. For example, if you change the name of the start-up script to **bcdBeClient1.sh**, you must also change the file's content to rename the **bcdBeClient** process **bcdBeClient1** and its log file **bcdBeClient1.log.**

The owner of the **bcdBeClient.sh** file is acs_oper. Follow these steps to modify the **bcdBeClient.sh** file:

Step	Action
1	Login to the SMS server as acs_oper.
2	Change directories to **/IN/service_packages/BCD/bin**.
3	Copy the existing start-up script and give the copy a new name. For example, **bcdBeClient2.sh**.

Step	Action

4	Using a text editor such as vi or vim, open the new start-up script and locate the following lines:
	`!/usr/bin/ksh`
	`exec /IN/service_packages/BCD/bin/bcdBeClient >>`
	`/IN/service_packages/BCD/tmp/bcdBeClient.log 2>&1`

5	Change the names of the BCD Client start-up script and its corresponding log file to match the name you gave to the new start-up script file. For example:
	`#!/usr/bin/ksh`
	`exec /IN/service_packages/BCD/bin/`**bcdBeClient2** `>>`
	`/IN/service_packages/BCD/tmp/`**bcdBeClient2.log** `2>&1`

6	Save and close the file.
7	Repeat steps 3 to 6 for each additional start-up script that you want to create.

Note: Given this example, you must also change the name of the original start-up script to **bcdBeClient1.sh** and the name of the process to **bcdBeClient1** and its log file to **bcdBeClient1.log**.

Configuring the SLEE to run the scripts

The BRM Charging Driver package installation adds the following line to the file **/IN/service_packages/SLEE/etc/SLEE.cfg**:

```
INTERFACE=bcdBeClient bcdBeClient.sh /IN/service_packages/BCD/bin EVENT
```

This line tells the SLEE to run the BCD Client start-up script. If you create multiple start-up scripts, you must add an INTERFACE entry to the **SLEE.cfg** file for each script you create. For example, if you create scripts **bcdBeClient1.sh** and **bcdBeClient2.sh**, you must add the following lines to the **SLEE.cfg** file:

```
INTERFACE=bcdBeClient1 bcdBeClient1.sh /IN/service_packages/BCD/bin EVENT
INTERFACE=bcdBeClient2 bcdBeClient2.sh /IN/service_packages/BCD/bin EVENT
```

The BRM Charging Driver installation package also adds the following line to the **eserv.config** file:

```
serviceDomainInterfaceName="bcdBeClient"
```

The value following the equal sign (bcdBeClient) defines the root name of the BCD Client start-up script.

Note: If you create multiple start-up scripts, you *do not* need to add the corresponding lines for the names of those scripts to the **eserv.config** file.

The names of the BCD Client in the INTERFACE parameters must be the same as the value of the `serviceDomainInterfaceName` parameter in the **eserv.config** file, appended with a number. The numbers must be sequential, beginning with 1 - for example, **bcdBeClient1**, **bcdBeClient2**, and so on.

The SLEE will distribute the load evenly across the specified BCD Client processes. If you leave a gap in the numbering, the SLEE will only distribute the load across the number of processes prior to the gap.

Starting the BCD Client processes

Start the SLEE to initiate the **bcdBeClient** processes that are defined in the new BCD Client start-up scripts, as well as the other SLEE applications. For information on stopping and starting the SLEE, see *Service Logic Execution Environment Technical Guide*.

Creating Balance Type Mappings

About Creating Balance Type Mappings

For all resources that are used by both Convergent Charging Controller and BRM, you must create Convergent Charging Controller balance types that match BRM resource IDs.

Convergent Charging Controller has two concepts for account balances:

- Currency (for example, Euros)
- Balance type (for example, General Cash)

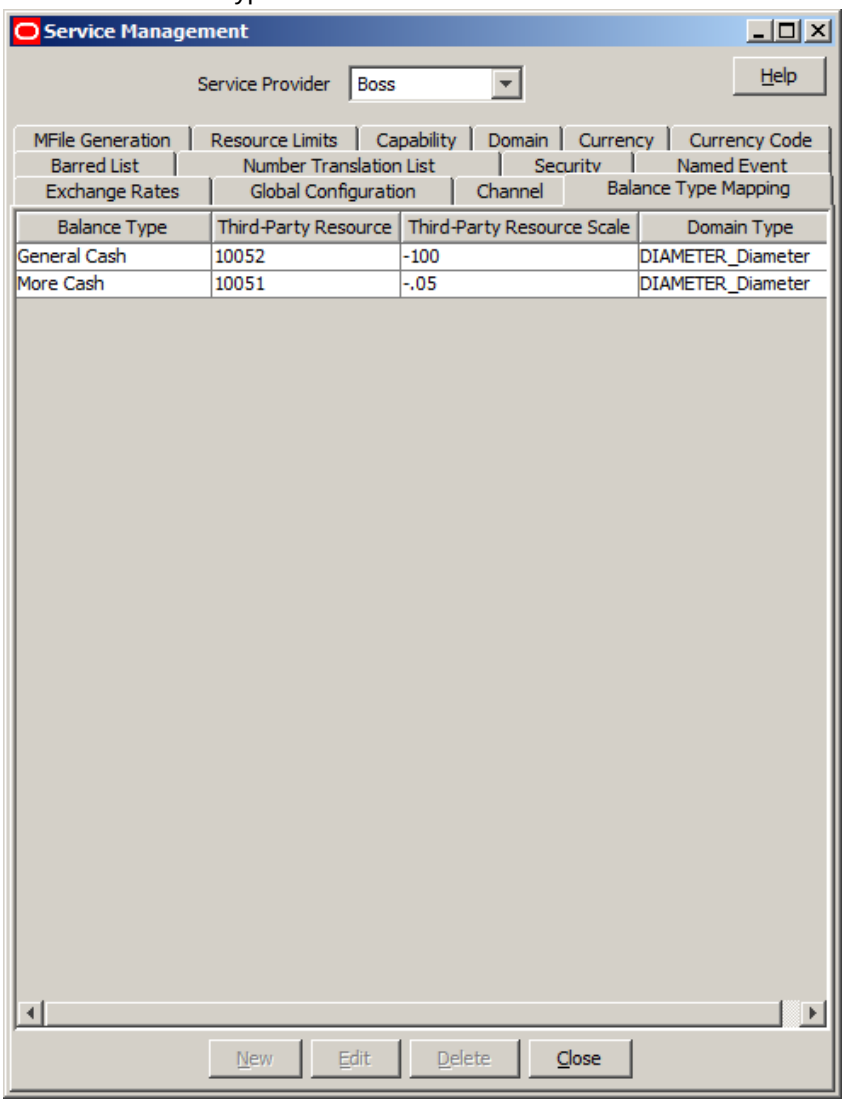
BRM, however, has only one concept: resource ID.

When Convergent Charging Controller queries BRM for a balance, Convergent Charging Controller expects both the currency and the balance type:

- BRM Charging Driver determines the currency by reading the **Currency Code** tab of the SMS **Service Management** screen. For example, the tab might show that Spain has a currency code of EUR.
- Convergent Charging Controller determines the balance type as follows:
  a) It uses the currency code to find the associated BRM resource ID in the `BrmToNccCurrencyMapping` array of the **eserv.config** file. For example, the following `BrmToNccCurrencyMapping` array shows that the EUR currency code has a BRM resource ID of 10052:

```
BrmToNccCurrencyMapping = [
   {
      NCCCode = "EUR"
      BRMNum =10052
   }
]
```

b) It uses the BRM resource ID (10052) to find the associated Convergent Charging Controller balance type in the **Balance Type Mapping** tab of the SMS **Service Management** screen. In the following example screen, the BRM resource ID of 10052 has the Convergent Charging Controller balance type of General Cash:



For example, if a balance in BRM is 15 with a resource ID of 10052, Convergent Charging Controller converts the balance to 15 Euros with a balance type of General Cash.

To configure Convergent Charging Controller to retrieve the currency and balance type for a BRM resource ID:

Step	Action
1	Define your Convergent Charging Controller currency codes by using the **Currency Code** tab of the SMS **Service Management** screen. See *Charging Control Services User's Guide*.
2	Map your Convergent Charging Controller currency codes to BRM resource IDs by using the `BrmToNccCurrencyMapping` array in the **eserv.config** file. See *Mapping Convergent Charging Controller currency codes to BRM values* (on page 19).
3	Define your Convergent Charging Controller balance types by using the **Balance Types** tab of the SMS **Wallet Management** screen. See *Charging Control Services User's Guide*.

Step	Action
4	Map your Convergent Charging Controller balance types to BRM resource IDs and specify any scaling that is required between Convergent Charging Controller and BRM by using the **Balance Type Mapping** tab of the SMS **Service Management** screen. See *Charging Control Services User's Guide.*

About Convergent Charging Controller and BRM Balances

In Convergent Charging Controller, a balance of $50 means the subscriber has $50 to spend, and a balance of -$10 means the subscriber owes $10. However, in BRM, a balance of $50 means the subscriber owes $50, and a balance of -$10 means the subscriber has $10 to spend.

BRM Charging Driver changes the sign when it presents balances to Convergent Charging Controller users. For example, a BRM balance of -$50 is shown in a Convergent Charging Controller user's text message as $50.

Chapter 4

Configuring BRM for the BRM Charging Driver

About Configuring BRM

Introduction

This chapter describes the BRM configuration tasks that you must perform to integrate BRM with Convergent Charging Controller.

To integrate BRM with Convergent Charging Controller, you must create custom fields and storable classes. You can also create subclasses of existing classes. This section describes the custom fields and storable classes that you must create to integrate BRM with Convergent Charging Controller. For more detailed descriptions of the steps to create custom fields and storable classes, see *Oracle Communications Billing and Revenue Management Developer's Guide*.

Note: Before you can create BRM custom fields and storable classes for Convergent Charging Controller, you must edit the Data Manager configuration file, **pin.conf**, to make the data dictionary writable. See the section on modifying the **pin.conf** file in the *BRM Developer's Guide,* and make sure you restore the settings to make the data dictionary unwritable after you are finished.

The BRM documentation describes how to perform the particular steps that are required by some of the tasks presented in this section.

In this chapter

This chapter contains the following topics.

Adding Custom Fields

About adding custom fields

Creating a custom field in BRM allows you to add a tag and value pair that BRM can add to records that it produces. For example, you can use the Convergent Charging Controller Set BE EDR feature node in a control plan to add an arbitrary tag and value pair to an event data record. To enable this capability in BRM, you must create a custom field and configure BRM to add that field to the records it produces.

You must also create the mapping between any Convergent Charging Controller information items and the corresponding BRM custom fields by adding the appropriate values to the bcdActionHandler.NccToBrmFieldMapping list in the Convergent Charging Controller **eserv.config** file. For more information on creating the mapping from Convergent Charging Controller to BRM fields, see *Mapping Convergent Charging Controller Information to BRM Fields* (on page 20).

You add the custom fields PIN_FLD_NCC_INFO and PIN_FLD_NCC_FIELD to make them available for the storable classes that you will define. Adding PIN_FLD_NCC_INFO creates a container for putting other Convergent Charging Controller fields in when Convergent Charging Controller sends operations to BRM. The PIN_FLD_NCC_FIELD is required because the PIN_FLD_NCC_INFO container must not be empty when Convergent Charging Controller sends an opcode to BRM, but for some operations there is nothing useful to put in the container.

Note: When you assign an ID to a custom field, the ID must be greater or equal to 10000 and must not already be in use. Some customization of BRM for other purposes might have already occurred at the time you configure it for integration with Convergent Charging Controller, so you cannot assume that ID 10000 will be free.

For examples of Convergent Charging Controller operations (opcodes) that include the custom fields you define, see the sections containing messages in *Usage Scenarios* (on page 51).

Use the BRM Storable Class Editor in the BRM Developer Center to create the following fields:

`PIN_FLD_NCC_INFO`

Name:	PIN_FLD_NCC_INFO
Type:	PIN_FLD_SUBSTRUCT
Description:	Convergent Charging Controller substruct containing custom fields
Default Field ID:	10000

Note: You must set `bcdActionHandler.NccInfoFieldNumber` in the Convergent Charging Controller **eserv.config** file to the assigned field ID. The default ID of 10000 is the ID assigned in the default **eserv.config** file during installation. If you need to use a different ID, you must change it in the **eserv.config** file as well.

`PIN_FLD_NCC_FIELD`

Name	PIN_FLD_NCC_FIELD
Type:	PIN_FLDT_STR
Default Field ID	10001
Description	Dummy field required to keep the PIN_FLD_NCC_INFO substruct from being empty when Convergent Charging Controller sends an opcode to BRM.

Note: You must set the corresponding `bcdActionHandler.NccInfoFieldDummyEntry` value in the Convergent Charging Controller **eserv.config** file to the assigned field ID. The default ID of 10001 is the ID assigned in the default **eserv.config** file during installation. If you need to use a different ID, you must change it in the **eserv.config** file as well.

`PIN_FLD_LOCATION`

GSM networks pass Convergent Charging Controller a location number, which gives an indication of where the caller is. It does not identify an individual phone, but does indicate where the caller is. For example, the location number for Ipswich in the United Kingdom is 00441473, where 00 is the international dialing prefix, 44 is the country code for the United Kingdom, and 1473 is the area code for Ipswich.

If you want to support location numbers, you must create a location field or use an existing one.

Note: If you will not be sending location numbers to BRM, set `bcdActionHandler.LocationNumberMapping.BRMField` in the Convergent Charging Controller **eserv.config** file to 0. Otherwise, set `bcdActionHandler.LocationNumberMapping.BRMField` to the assigned field ID.

The ID 1251 is the pre-defined BRM field ID of PIN_FLD_LOCATION. If LocationNumberMapping.BRMField is some other number, you must set the Default Field ID to that number. For more information, see *Mapping the Location Number* (on page 21).

Name: PIN_FLD_LOCATION
Type: PIN_FLDT_STR
Default Field ID: 1251

Creating additional custom fields

Create additional custom fields for other information items that you wish to support. For any additional fields that you define, you must also create the mapping between Convergent Charging Controller and BRM fields by adding the appropriate values to the `bcdActionHandler.NccToBrmFieldMapping` list in the Convergent Charging Controller **eserv.config** file. Each entry should contain the following parameters:

- `NCCItem` - The text name of the Convergent Charging Controller concept
- `BRMType` - "STRING, "DECIMAL", "INT", "ENUM", or "TIMESTAMP"
- `BRMField` - The field ID of the custom field

The following custom field definitions are provided as examples.

`PIN_FLD_NCC_NUMBER_OF_EVENTS`

Add the PIN_FLD_NCC_NUMBER_OF_EVENTS field if you wish to support named events such as SMS, MMS, or email.

Name: PIN_FLD_NCC_NUMBER_OF_EVENTS
Type: PIN_FLDT_INT

Add the following values to the `bcdActionHandler.NccToBrmFieldMapping` list:

- `NCCItem`: "NUMBER_OF_EVENTS_ID"
- `BRMType`: "INT"

`PIN_FLD_NCC_TARIFF_PLAN_ID`

Add the PIN_FLD_NCC_TARIFF_PLAN_ID field if you wish to support a tariff plan ID.

Name: PIN_FLD_NCC_TARIFF_PLAN_ID
Type: PIN_FLDT_INT

Add the following values to the `bcdActionHandler.NccToBrmFieldMapping` list:

- `NCCItem`: "TARIFF_PLAN_ID"
- `BRMType`: "INT"

Adding Storable Classes

Creating custom classes

The BRM data dictionary uses storable classes to define various types of data. There are storable classes, for example, that define an account, a service object, an activity event, and a session event. To store Convergent Charging Controller activities, you must create custom fields and storable classes to define those activities for the BRM data dictionary.

Note: Before you can change or add storable classes for Convergent Charging Controller, you must edit the Data Manager configuration file, **pin.conf**, to make the data dictionary writable. See the section on modifying the **pin.conf** file in the *BRM Developer's Guide,* and be sure you restore the settings to make the data dictionary unwritable when you are finished.

The **/event/activity/telco/gsm** class is the standard BRM class for handling GSM mobile calls and the **/active_session/telco/gsm** is a sister class necessary for doing real-time charging. You must create subclasses for both classes so you can add the custom fields to them.

Use the BRM Storable Class Editor, which is part of the BRM Developer Center, to create an Convergent Charging Controller subclass for each of the following classes:

```
/active_session/telco/gsm
/event/activity/telco/gsm
```

The following example illustrates the appropriate subclasses:

```
/active_session/telco/gsm/ncc
/event/activity/telco/gsm/ncc
```

Using the Storable Class Editor, add the custom fields PIN_FLD_NCC_INFO and PIN_FLD_NCC_FIELD, as well as any others that you have defined, to each of these subclasses. For more information, see the section on creating custom fields and storable classes in *Oracle Communications Billing and Revenue Management Developer's Guide*.

If all of the products, plans and deals that you will define in BRM Pricing Center refer to **/event/activity/telco/gsm**, then extending these two classes is sufficient. If your products, plans, and deals refer to other classes, then you must create subclasses for those classes as well, and add the Convergent Charging Controller custom fields to them.

Creating Header and Library Files for the Custom Classes

Creating the custom header and library files

Follow these steps to create the BRM header and library files required to make your Convergent Charging Controller custom fields available to BRM applications.

Step	Action
1	Use the BRM Developer Center to generate a header file, for example *ncc_flds.h*, for the custom Convergent Charging Controller fields that you added.
2	Copy the header file to the appropriate location.
3	Create a library file from the header file.
4	Copy the library file to the appropriate location and make it available to your applications.
5	Restart processes.

See the section on making custom fields available to your application in the *BRM Developer's Guide* for the specific directions to accomplish these general steps.

Generating the Custom JAR File

Steps to generate JAR file

In addition to creating the custom header and library files, you must generate a custom **.jar** file for the custom fields and storable classes and configure it for BRM client application such as BRM Developer Center, Pricing Center, or Customer Center.

You created the necessary **.java** files and specified a location for them when you generated the header file for the custom fields in Developer Center.

For information how to compile the **.java** files create the custom **.jar** file, see the section on using custom fields in Java applications in *BRM Developer's Guide*. These are the general steps:

Step	Action
1	Compile the **.java** files.
2	Move the compiled **.class** files to a directory.
3	Create a Java archive (.jar) file.
4	Copy the **.jar** file into a directory that client applications can access
5	Edit the Developer Center start-up script, which is platform specific, to include the custom **.jar** file.
6	Add the new custom fields generated by Developer Center to the **Infranet.properties** file of the relevant BRM client applications. **Infranet.properties** is a configuration file used by BRM client applications such as Pricing Center, Developer Center and Customer Center. The **Infranet.properties** file must be changed on every computer on which the client applications run.

Modifying the BRM Configuration Files

Steps to modify BRM configuration files

To integrate BRM with Convergent Charging Controller, you must add entries to the following BRM configuration files: **pin_event_map**, **pin_rum**, and **pin_config_reservation_aaa_prefs**.

The **pin_event_map** file configures the mapping of a service type with an event type. You must add this mapping for any custom service and event object types. This enables BRM to associate the price offers for a specific service with the given event types. As delivered, the **pin_event_map** file contains the mapping for the default service and event types that are supported. For more information, see the section on mapping event types to services in *BRM Setting Up Pricing and Rating.*

The event data that you use to rate an event is called ratable usage metrics. Common examples of ratable usage metrics are duration, in which you rate based on the length of time an event lasts, and occurrence, in which you rate based on the number of events that occur, regardless of their duration. The **pin_rum** file specifies the ratable usage metrics for an event type. For information on specifying ratable usage metrics, see the section on setting up ratable usage metrics in *BRM Setting up Pricing and Rating*.

For information on the **pin_config_reservation_aaa_prefs** file, see the section on specifying default AAA preferences in *BRM Telco Integration*.

Follow these steps to modify the BRM configuration files for integration with Convergent Charging Controller.

Step	Action

1	If they are not already present, add the BRM events that you will use with Convergent Charging Controller to your **pin_event_map** file, for example, **pin_event_map_telco__gsm**. For the example products used with these instructions, you would add the following events:
	`/service/telco/gsm : /event/session/telco/gsm : Real Time Telco GSM Service`
	`/service/telco : /event/activity/telco/gsm/ncc : Real Time Telco GSM Convergent Charging Controller Activity`
	`/service/telco : /event/activity/gsm/ncc : Real Time GSM Convergent Charging Controller Activity`

2	Load the **pin_event_map** file, following instructions for the **load_pin_event** utility in *BRM Setting Up Pricing and Rating*

Step	Action
3	If you require support for named events, add the following line to the **pin_rum** file, substituting your event class, if it is different than the one shown. `/event/activity/telco/gsm/ncc : Number Of Events : PIN_FLD_NCC_INFO.PIN_FLD_NCC_NUMBER_OF_EVENTS : none`
4	Load the **pin_rum** file, following instructions for the **load_pin_rum** utility in *BRM Setting Up Pricing and Rating*.
5	If you require support for data calls, modify the **pin_config_reservation_aaa_prefs** file, for example **pin_config_reservation_aaa_prefs_gsm_data**, multiplying appropriate QUANTITY fields by the scaling factor. The scaling factor is defined in the ScalingFactor parameter of the data call entry in the bcdActionHandler.ServiceProfileTagMapping list in the Convergent Charging Controller **eserv.config** file on the SLC. For more information on the scaling factor, see *Mapping Convergent Charging Controller Sessions to BRM Services* (on page 16). For example, modify the default **pin_config_reservation_aaa_prefs_gsm_data** file to match the scaling factor: 0 PIN_FLD_RESERVATION_INFO ARRAY [0] 1 PIN_FLD_QUANTITY DECIMAL [0] <50 x scaling factor> 1 PIN_FLD_MIN_QUANTITY DECIMAL [0] 0 1 PIN_FLD_INCR_QUANTITY DECIMAL [0] <50 x scaling factor> 1 PIN_FLD_RUM_NAME STR [0] "Size" 1 PIN_FLD_REQ_MODE ENUM [0] 4 1 PIN_FLD_UNIT ENUM [0] 0 0 PIN_FLD_RESERVATION_INFO ARRAY [1] 1 PIN_FLD_QUANTITY DECIMAL [0] 50 1 PIN_FLD_MIN_QUANTITY DECIMAL [0] 0 1 PIN_FLD_INCR_QUANTITY DECIMAL [0] 50 1 PIN_FLD_RUM_NAME STR [0] "Amount" 1 PIN_FLD_REQ_MODE ENUM [0] 1 1 PIN_FLD_UNIT ENUM [0] 0
6	Load the **pin_config_reservation_aaa_prefs** file following instructions in *BRM Telco Integration* for **load_config_reservation_aaa_prefs**. See the section on telco integration utilities.
7	Restart the BRM Connection Manager.

For more information about modifying these configuration files, see *BRM Setting Up Pricing and Rating*.

Chapter 5

Creating Products and Deals

About Creating Products and Deals

Introduction

This chapter describes how to create BRM products, plans and deals to set up rating and charging on BRM for the features that you require. Add the relevant products, plans and deals and assign the plans to your customers. Plans, for example, allow you to specify charges for local and international voice calls during peak and off peak hours, as well as charges for data and short messages, and discounts for friends and family, and so on.

The following sections provide examples of products and deals for a sampling of features. In general, you can set up the products however you like as long as you create subclasses for the classes referred to by the products and add the Convergent Charging Controller custom fields. You must also match the corresponding Convergent Charging Controller parameters to the product values as follows:

- The Convergent Charging Controller `BrmServicePoid` parameter must match the product value "Applies to".
- The Convergent Charging Controller `BrmObjectType` parameter must match the product value "Event".
- The Convergent Charging Controller `BrmReqMode` parameter must match the product value "Measured by".

The service and event names used are only examples. The examples, however, are intended to be sensible and usable.

In this chapter

This chapter contains the following topics.

Creating a Product and Deal for Voice Calls

Creating the BRM product and deal for voice calls

Using the Pricing Center on BRM, create a product for voice calls with the following properties:

- Applies to: **/service/telco/gsm/telephony**

  On Convergent Charging Controller, this service is defined in the `BrmServicePoid` parameter for the voice call entry in the **eserv.config** file. For example:

  ```
  BrmServicePoid = "/service/telco/gsm/telephony"
  ```
- Event set to **/event/session/telco/gsm**

  On Convergent Charging Controller, the `BrmObjectType` parameter for the voice call entry must match the suffix of the BRM event that follows the **/event/session/telco** prefix. In this example `BrmObjectType = "gsm"`.
- Measured by: Duration

On Convergent Charging Controller, the `BRMReqMode` parameter in the **eserv.config** file must be set to `"DURATION"`.

Create a deal for this product and a plan containing the relevant deals.

Creating a Product and Deal for Data Calls

Creating the BRM product and deal for data calls

This is a real time telco GSM session that is measured by Volume. On BRM, create a product with the following properties:

- Applies to: **/service/telco/gsm/data**
  On Convergent Charging Controller, this service is defined in the `BrmServicePoid` parameter for the data call entry. In this case, for example:
  `BrmServicePoid = "/service/telco/gsm/data"`
- Event set to **/event/session/telco/gsm**
  On Convergent Charging Controller, the `BrmObjectType` parameter for the data call must match the suffix of the BRM event that follows the **/event/session/telco** prefix. In this example, `BrmObjectType = "gsm"`
- Measured by: Volume
  On Convergent Charging Controller, `BrmReqMode` parameter must be set to `"VOLUME"`

Create a deal for this product and a plan containing the relevant deals.

Creating a Customer

Creating a customer

For any Convergent Charging Controller subscriber who will use BRM for rating and charging, you must create a customer record for the subscriber on BRM.

Before creating a customer on BRM, you must select the BRM Charging Driver domain (for example, BCD) when you create the subscriber's wallet. For information on creating a subscriber's wallet, see *Charging Control Services User's Guide*.

For information on creating a new BRM customer using the BRM Customer Center, see *BRM Managing Customers* and BRM Customer Center Help. Use the following steps as a general guide:

Step	Action
1	Select the plan containing the deals that you defined in *Creating Products and Deals* (on page 47).
2	Assign a number and SIM to each service that is included in the plan.
3	If you require support for vouchers, create a separate BRM customer for the Voucher Administration Center. • Select CSR Plan • Assign an ID and password. You login to Voucher Administration Center using this name and password when generating vouchers.

In practice, you might want to add Convergent Charging Controller subscribers in bulk to the BRM database. For information on mapping data from another system to create new BRM customers in bulk, see *BRM Managing Customers*.

Chapter 6

Generating Statistics and Reports

Overview of Statistics and Reports

Introduction

This chapter explains the statistics collection and reporting that the BRM Charging Driver performs.

In this chapter

This chapter contains the following topics.

About Statistics and Reports

Generating statistics

The BRM Charging Driver uses the SMS statistics mechanism to store statistics. To collect BRM Charging Driver statistics, you must first ensure that the replication group for the SMF_STDEF_BCD table is replicated to all SLC machines. You configured replication for the SMF_STDEF_BCD table as part of configuring Convergent Charging Controller for the BRM Charging Driver. For more information, see *Configuring Replication for the BRM Charging Driver Tables* (on page 14).

For information on the SMS statistics mechanism and on configuring replication for SMF database tables, see *Service Management System Technical Guide* and *Service Management System User's Guide*.

To begin collecting statistics on all SLC machines, log in to each SLC machine as smf_oper and execute the following command:

```
kill -HUP process ID of smsStatsDaemon
```

Convergent Charging Controller immediately begins collecting BRM Charging Driver statistics for the SLC machines.

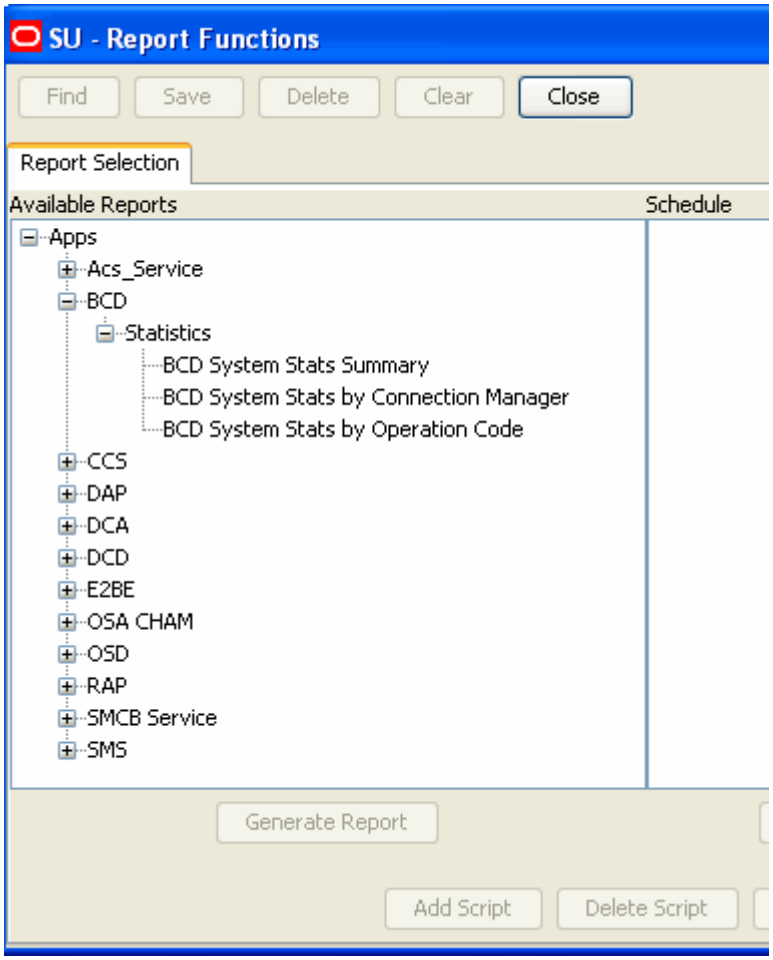
By default, Convergent Charging Controller collects all BRM Charging Driver statistics every five minutes.You can set the frequency through the SMS UI. Select the **Operator Functions** menu, then **Statistics Management**, and then **Statistics**.

Generating reports

To generate reports on BRM Charging Driver statistics, go to the **Operator Functions** menu in the Convergent Charging Controller SMS UI and select **Report Functions**. On the **Report Selection** tab, select one of the following three reports from the **Statistics** branch under **BCD**.

- BCD System Stats Summary
- BCD System Stats by Connection Manager
- BCD System Stats by Operation Code

The following figure shows the Report Functions window and the **Report Selection** tab.



For more information about generating reports, see the discussion on report functions in *Service Management System User's Guide*.

Chapter 7

Usage Scenarios

About Usage Scenarios

Introduction

This chapter presents some common usage scenarios that describe the interactions of key components when Convergent Charging Controller integrates BRM into the charging and account settlement process.

In this chapter

This chapter contains the following topics.

Recharge using VWS Vouchers

Preconditions to recharge using VWS vouchers

The scenarios in this section describe a subscriber who has a VWS voucher and uses it to recharge the amount on a BRM account using either IVR or USSD. These scenarios assume the following preconditions:

- The subscriber is provisioned so that voice calls run a control plan with a Named Event feature node.
- The subscriber's MSISDN is provisioned on the BRM with prepaid GSM service enabled.
- The voucher is provisioned on the VWS.

IVR VWS voucher recharge

IVR VWS voucher recharge flow

Here is an example message flow covering the IVR, VWS voucher recharge scenario.

IVR VWS voucher recharge scenario

This scenario describes subscriber who successfully recharge the amount on a BRM account using a VWS voucher.

See Preconditions for Recharge Using VWS Vouchers for information about the preconditions for this scenario.

Action	Description
1	• The subscriber calls the toll free number for a voucher recharge. • MSC sends InitialDP to the slee_acs process with serviceKey set to a special value indicating voucher recharge. • xmsTrigger sends InitialDP to the slee_acs process.

Action	Description
2	• The CCS service loader looks up the subscriber and wallet in the SCP database on the SLC and determines that the wallet information is on the BRM domain and VWS is used for the voucher. It also uses the serviceKey to select the voucher recharge control plan. • The slee_acs process runs the control plan. • The slee_acs process reaches a Voucher Recharge feature node. • The Voucher Recharge feature node sends ConnectToResource, PromptAndCollectUserInformation to the MSC, instructing the MSC to prompt the caller for the voucher number.
3	• MSC plays the specified announcement to the caller and collects the voucher number. • MSC sends PromptAndCollectUserInformation result, containing the voucher number to the slee_acs process.
4	• The Voucher Recharge feature node invokes the VoucherRedeem action on the FOX actions library. • FOX actions library sends VR_Req to BeClient. • BeClient sends VR_Req to VWS.
5	• VWS sends VR_Ack to BeClient, indicating that the voucher has been reserved, and returning the recharge amounts for each balance type. • BeClient sends VR_Ack to the slee_acs process.
6	• The Voucher Recharge feature node calls the WalletRecharge action on the BCD actions library. • BCD actions library can optionally construct an event (BcdSleeEvent) containing PCM_OP_ACT_FIND and sends it to the BCD Client. • The BCD Client calls the PCM_OP_ACT_FIND opcode and sets a timer to the configured value for this type of operation.
7	• BRM responds to the operation, providing account and balance group object strings.
8	• BCD actions library constructs an event (BcdSleeEvent) containing PCM_OP_BILL_DEBIT and sends it to the BCD Client. • The BCD Client calls the PCM_OP_BILL_DEBIT opcode and sets a timer to the configured value for this type of operation.
9	• BRM responds to the operation, indicating that the account has been successfully recharged. • The BCD Client receives the operation output flist and sends it in a BcdSleeEvent to the slee_acs process. It also marks the BRM connection as free for reuse and cancels the operation timer.
10	• The VoucherRecharge feature node invokes the VoucherConfirm action on the FOX actions library. • The FOX actions library sends CVR_Req to BeClient. • BeClient sends CVR_Req to VWS.
11	• VWS sends CVR_Ack to BeClient, indicating that the voucher has been permanently marked as redeemed. • BeClient sends CVR_Ack to the slee_acs process.
12	• The VoucherRecharge feature node sends PlayAnnouncement to MSC. • MSC plays an announcement to the caller, stating that the recharge was successful.

Action	Description
13	• The announcement finishes and MSC sends SpecializedResourceReport to the slee_acs process. • The control plan reaches a Play Voucher Redeem Balances feature node, which invokes a VoucherInfo action on the FOX actions library. • The FOX actions library returns the balance update information returned in the VR_Ack.
14	• The Play Voucher Redeem Balances feature node sends a PlayAnnouncement to the MSC. • MSC plays the balance update information to the caller in an announcement.
15	The announcement ends and the MSC sends SpecializedResourceReport to the slee_acs process.
16	• The control plan reaches an end node and ACS sends DisconnectForwardConnection, ReleaseCall to the MSC and clears the call context. • The caller is disconnected.

Messages: redemption of VWS voucher against BRM account

The following messages include operations sent to BRM and results returned by BRM for the redemption of a VWS voucher against a BRM account with two balance groups, with one phone number in each group. The general message format is: nesting level (0; 1, or 2); field; data type; value.

Operation: Send PCM_OP_ACT_FIND (159)

```
Flags = PCM_OPFLG_READ_RESULT
0 PIN_FLD_POID      POID [0] 0.0.0.1 /service/telco/gsm/telephony -1 0
0 PIN_FLD_LOGIN           STR [0] "0049100701"
```

Result: Receive for operation PCM_OP_ACT_FIND (159)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony 89343 29
0 PIN_FLD_CREATED_T    TSTAMP [0] (1480252152) Sun Nov 27 13:09:12 2016
0 PIN_FLD_MOD_T        TSTAMP [0] (1484782017) Wed Jan 18 23:26:57 2017
0 PIN_FLD_READ_ACCESS     STR [0] "L"
0 PIN_FLD_WRITE_ACCESS    STR [0] "L"
0 PIN_FLD_AAC_ACCESS      STR [0] ""
0 PIN_FLD_AAC_PACKAGE     STR [0] ""
0 PIN_FLD_AAC_PROMO_CODE    STR [0] ""
0 PIN_FLD_AAC_SERIAL_NUM    STR [0] ""
0 PIN_FLD_AAC_SOURCE STR [0] ""
0 PIN_FLD_AAC_VENDOR      STR [0] ""
0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 135267 0
0 PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 136803 166
0 PIN_FLD_CLOSE_WHEN_T TSTAMP [0] (0) <null>
0 PIN_FLD_EFFECTIVE_T  TSTAMP [0] (1484782015) Wed Jan 18 23:26:55 2017
0 PIN_FLD_ITEM_POID_LIST    STR [0] "0.0.0.1|/item/cycle_forward 2972997 0"
0 PIN_FLD_LASTSTAT_CMNT    STR [0] ""
0 PIN_FLD_LAST_STATUS_T TSTAMP [0] (1480252151) Sun Nov 27 13:09:11 2016
0 PIN_FLD_LIFECYCLE_STATE    INT [0] 0
0 PIN_FLD_LOGIN      STR [0] "144-20161127-050912-0-22971--152156464-den00bqf"
0 PIN_FLD_NAME            STR [0] "PIN Service Object"
0 PIN_FLD_NEXT_ITEM_POID_LIST    STR [0] ""
0 PIN_FLD_OBJECT_CACHE_TYPE   ENUM [0] 0
0 PIN_FLD_PASSWD     STR [0]
"sha1|0471FA6494968D7344714951EDED8578903F4077|24F3B5A01FD45A7F0387F9D96972070A"
0 PIN_FLD_PASSWD_EXPIRATION_T TSTAMP [0] (0) <null>
0 PIN_FLD_PASSWD_STATUS   ENUM [0] 0
0 PIN_FLD_PROFILE_OBJ    POID [0] 0.0.0.0  0 0
0 PIN_FLD_SERVICE_ID      STR [0] ""
0 PIN_FLD_SERVICE_STATE_EXPIRATION_T TSTAMP [0] (0) <null>
0 PIN_FLD_STATUS         ENUM [0] 10100
0 PIN_FLD_STATUS_FLAGS    INT [0] 0
0 PIN_FLD_SUBSCRIPTION_OBJ   POID [0] 0.0.0.0  0 0
```

```
0 PIN_FLD_TYPE      ENUM [0] 0
0 PIN_FLD_ALIAS_LIST    ARRAY [0] allocated 20, used 1
1     PIN_FLD_NAME            STR [0] "0049100701"
0 PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 1
1     PIN_FLD_PRIMARY_NUMBER    INT [0] 0
0 PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 2
1     PIN_FLD_BEARER_SERVICE    STR [0] "Bearer Service"
1     PIN_FLD_IMEI            STR [0] "IMEI"
```

Operation: Send PCM_OP_BILL_DEBIT (105)

```
Flags = 0
0 PIN_FLD_POID          POID [0] 0.0.0.1 /account 135267 0
0 PIN_FLD_PROGRAM_NAME   STR [0] "NCC_BCD_Client"
0 PIN_FLD_SESSION_ID     INT [0] 0
0 PIN_FLD_AUTHORIZATION_ID   STR [0] "brmClient-XXX-2014-4-28-7_recharge_142092_0"
0 PIN_FLD_OBJ_TYPE       STR [0] "gsm/ncc"
0 PIN_FLD_DEBIT_INFO   ARRAY [0] allocated 20, used 2
1   PIN_FLD_BAL_GRP_OBJ      POID [0] 0.0.0.1 /balance_group 136803 6
1   PIN_FLD_DEBIT           ARRAY [840] allocated 20, used 2
2        PIN_FLD_BAL_OPERAND      DECIMAL [0] -0.500000000000000
2        PIN_FLD_FLAGS                INT [0] 1
0 PIN_FLD_DESCR          STR [0] "VOUCHER=6"
```

Note: We are debiting -50 Euro cents which is the same as crediting 50 Euro cents because this is a 50 Euro cent voucher.

Result: Received for operation PCM_OP_BILL_DEBIT (105)

```
0 PIN_FLD_POID        POID [0] 0.0.0.1 /account 313603 0
0 PIN_FLD_RESULTS    ARRAY [0] allocated 20, used 4
1    PIN_FLD_BAL_IMPACTS      ARRAY [0] allocated 20, used 17
2        PIN_FLD_IMPACT_TYPE        ENUM [0] 2
2        PIN_FLD_RESOURCE_ID         INT [0] 840
2        PIN_FLD_RESOURCE_ID_ORIG    INT [0] 0
2        PIN_FLD_TAX_CODE            STR [0] ""
2        PIN_FLD_RATE_TAG            STR [0] ""
2        PIN_FLD_ACCOUNT_OBJ        POID [0] 0.0.0.1 /account 135267 0
2        PIN_FLD_RATE_OBJ           POID [0] 0.0.0.0 0 0
2        PIN_FLD_DISCOUNT        DECIMAL [0] 0
2        PIN_FLD_PERCENT         DECIMAL [0] 0
2        PIN_FLD_QUANTITY        DECIMAL [0] 0
2        PIN_FLD_AMOUNT_DEFERRED DECIMAL [0] 0
2        PIN_FLD_AMOUNT          DECIMAL [0] -0.50
2        PIN_FLD_AMOUNT_ORIG     DECIMAL [0] NULL pin_decimal_t ptr
2        PIN_FLD_BAL_GRP_OBJ        POID [0] 0.0.0.1 /balance_group 136803 6
2        PIN_FLD_GL_ID               INT [0] 0
2        PIN_FLD_ITEM_OBJ           POID [0] 0.0.0.1 /item/misc 355675 0
2        PIN_FLD_LINEAGE             STR [0] NULL str ptr
1    PIN_FLD_SUB_BAL_IMPACTS   ARRAY [0] allocated 20, used 3
2        PIN_FLD_BAL_GRP_OBJ        POID [0] 0.0.0.1 /balance_group 136803 7
2        PIN_FLD_RESOURCE_ID         INT [0] 840
2        PIN_FLD_SUB_BALANCES      ARRAY [2] allocated 20, used 8
3            PIN_FLD_VALID_FROM          TSTAMP [0] (1395039600) Mon Mar 17 07:00:00 2014
3            PIN_FLD_VALID_TO            TSTAMP [0] (0) <null>
3            PIN_FLD_VALID_FROM_DETAILS     INT [0] 1
3            PIN_FLD_VALID_TO_DETAILS       INT [0] 0
3            PIN_FLD_GRANTOR_OBJ           POID [0] 0.0.0.1 /purchased_product 3132510
3            PIN_FLD_ROLLOVER_DATA          INT [0] 0
3            PIN_FLD_CONTRIBUTOR_STR        STR [0] ""
3            PIN_FLD_AMOUNT              DECIMAL [0] -0.50
1    PIN_FLD_ACCOUNT_OBJ         POID [0] 0.0.0.1 /account 135267 0
1    PIN_FLD_POID                POID [0] 0.0.0.1 /event/billing/debit 284782307687359323 0
```

Messages: redemption of VWS voucher using incorrect voucher number

The following messages include operations sent to BRM and results returned by BRM for the redemption of a VWS voucher against a BRM account using a bad voucher number. The general message format is: nesting level (0; 1, or 2); field; data type; value.

Operation: Send PCM_OP_ACT_ACTIVITY (151)

```
Flags = 0
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 135267 0
0 PIN_FLD_PROGRAM_NAME    STR [0] "NCC_BCD_Client"
0 PIN_FLD_OBJ_TYPE        STR [0] "/voucher"
0 PIN_FLD_INHERITED_INFO SUBSTRUCT [0] allocated 20, used 1
1     10000               SUBSTRUCT [0] allocated 20, used 2
2         10007                   STR [0] "9876543214567"
0 PIN_FLD_BAL_IMPACTS   ARRAY [1000011] allocated 20, used 4
1     PIN_FLD_AMOUNT       DECIMAL [0] 1
1     PIN_FLD_RESOURCE_ID     INT [0] 1000011
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 135267 0
1     PIN_FLD_IMPACT_TYPE    ENUM [0] 2
```

Note: Field 10007 has been defined as "VOUCHER_SERIAL" respectively in the NccToBrmFieldMapping section of **eserv.config**. It has also been defined in BRM using Developer Center. BRM can be configured to produce an EDR showing the voucher serial number.

Result: Received for operation PCM_OP_ACT_ACTIVITY (151)

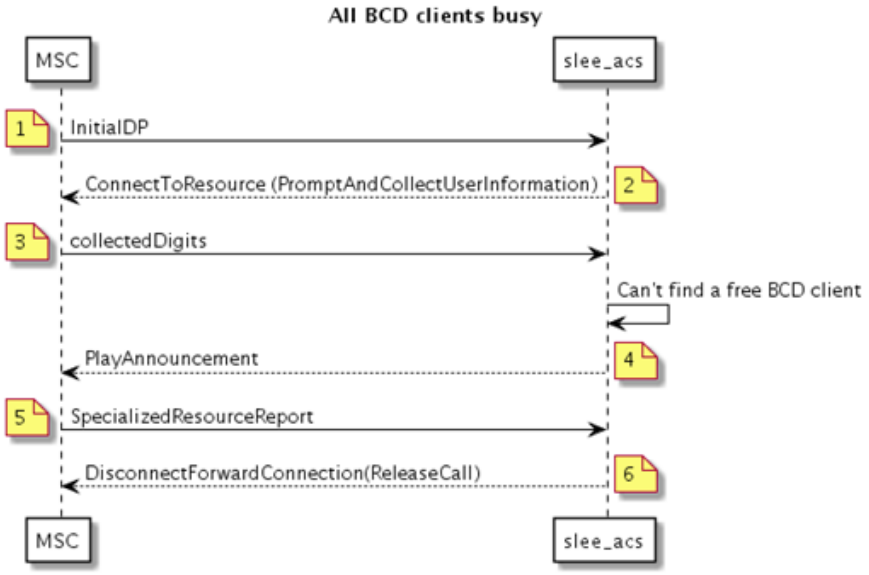
```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 135267 0
0 PIN_FLD_RESULTS       ARRAY [0] allocated 20, used 4
1     PIN_FLD_BAL_IMPACTS   ARRAY [1000011] allocated 20, used 8
2         PIN_FLD_AMOUNT       DECIMAL [0] 1
2         PIN_FLD_RESOURCE_ID     INT [0] 1000011
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 135267 0
2         PIN_FLD_IMPACT_TYPE    ENUM [0] 2
2         PIN_FLD_GL_ID           INT [0] 0
2         PIN_FLD_ITEM_OBJ       POID [0] 0.0.0.1 /item/misc 139539 0
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 136803 247
2         PIN_FLD_LINEAGE         STR [0] NULL str ptr
1     PIN_FLD_SUB_BAL_IMPACTS  ARRAY [1000011] allocated 20, used 3
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 136803 248
2         PIN_FLD_RESOURCE_ID     INT [0] 1000011
2         PIN_FLD_SUB_BALANCES  ARRAY [4] allocated 20, used 8
3             PIN_FLD_VALID_FROM   TSTAMP [0] (1331753228) Wed Mar 14 19:27:08 2012
3             PIN_FLD_VALID_TO     TSTAMP [0] (0) <null>
3             PIN_FLD_ROLLOVER_DATA    INT [0] 0
3             PIN_FLD_VALID_FROM_DETAILS    INT [0] 0
3             PIN_FLD_VALID_TO_DETAILS    INT [0] 0
3             PIN_FLD_GRANTOR_OBJ    POID [0] 0.0.0.0  0 0
3             PIN_FLD_CONTRIBUTOR_STR    STR [0] ""
3             PIN_FLD_AMOUNT       DECIMAL [0] 1
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 135267 0
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/activity/voucher 271148363502725395 0
```

Other Scenarios

All BCD clients busy

All BCD clients busy flow

Here is an example of message flow that illustrates the sequence of messages when all BCD Clients are busy.



All BCD clients busy scenario

This scenario describes the sequence of messages that occurs when a subscriber makes a call and all BCD Clients are busy.

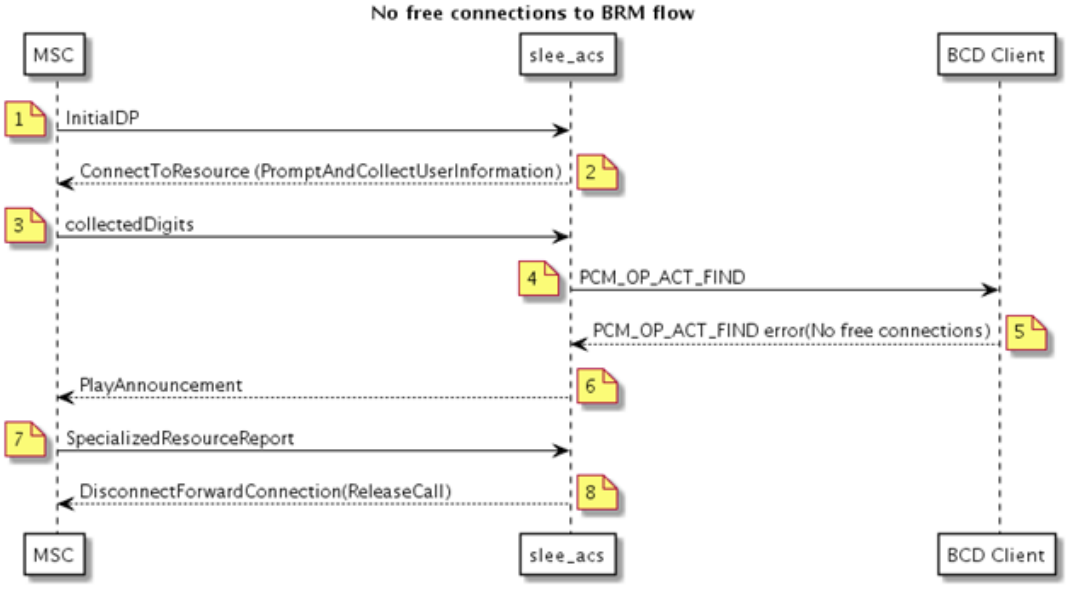
Step	Action
1.	• The subscriber dials the “subscribe to service” toll free number. • MSC sends InitialDP to the slee_acs process, with serviceKey set to a special value indicating “subscribe to service”. • The CCS service loader searches for the subscriber and wallet in the SCP database on the SLC and determines the wallet information on the BRM domain. It also determines the control plan to run, using the serviceKey to select the “subscribe to service” control plan. • The slee_acs process runs the control plan. • The slee_acs process reaches a Selection Dependent Routing feature node.
2.	• The Selection Dependent Routing feature node sends ConnectToResource (PromptAndCollectUserInformation) to the MSC, instructing the MSC to prompt for a single digit from a menu. • MSC plays the announcement to the caller and collects the menu choice.

Step	Action
3.	• MSC sends PromptAndCollectUserInformation result, containing the menu choice digit, to the slee_acs process. • The SDR feature node takes an exit to a Named Event feature node and specifies a direct named event for a non-reservable event. • The BCD actions library constructs a BcdSleeEvent to invoke an authorize operation but is unable to send it to the least busy BCD Client. The BCD actions library returns an error response to the Direct Named Event action. • The nevt feature node takes the Billing Fault exit and reaches a Play Announcement node for an error announcement.
4.	• A Play Announcement feature node sends PlayAnnouncement to the MSC. • MSC plays an announcement to the caller, stating that the transaction was not successful.
5.	• The announcement finishes and MSC sends SpecializedResourceReport to the slee_acs process.
6.	• The control plan reaches an end node. ACS sends DisconnectForwardConnection and ReleaseCall actions to MSC and clears the call context. • The caller is disconnected.

No free connections to BRM

No free connections to BRM flow

Here is an example of message flow that illustrates the sequence of messages when there are no free connections to BRM.



No free connections to BRM scenario

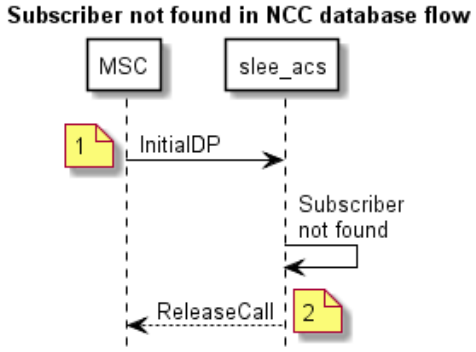
This scenario describes the sequence of messages that occurs when a subscriber's wallet is in the BRM domain and there are no free connections to BRM.

Step	Action
1	• The subscriber dials the “subscribe to service” toll free number. • MSC sends InitialDP to the slee_acs process, with serviceKey set to a special value indicating “subscribe to service”. • The CCS service loader searches for the subscriber and wallet in the SCP database on the SLC and determines the wallet information on the BRM domain. It also determines the control plan to run, using the serviceKey to select the “subscribe to service” control plan. • The slee_acs process runs the control plan. • The slee_acs process reaches a Selection Dependent Routing feature node.
2	• The Selection Dependent Routing feature node sends ConnectToResource (PromptAndCollectUserInformation) to the MSC, instructing the MSC to prompt for a single digit from a menu. • MSC plays the announcement to the caller and collects the menu choice.
3	• MSC sends PromptAndCollectUserInformation result, containing the menu choice digit, to the slee_acs process. • The SDR feature node takes an exit to a Named Event feature node and specifies a direct named event for a non-reservable event.
4	• The BCD actions library constructs a BcdSleeEvent to invoke an authorize operation and sends it to the least busy BCD Client.
5	• The BCD Client has no free BRM connections so sends a BcdSleeEvent to slee_acs with a status indicating that the BRM could not be contacted. • The nevt feature node takes the Billing Fault exit and reaches a Play Announcement node for an error announcement.
6	• A Play Announcement feature node sends PlayAnnouncement to the MSC. • MSC plays an announcement to the caller, stating that the transaction was not successful.
7	• The announcement finishes and MSC sends SpecializedResourceReport to the slee_acs process.
8	• The control plan reaches an end node. ACS sends DisconnectForwardConnection and ReleaseCall actions to MSC and clears the call context. • The caller is disconnected.

Subscriber not found in Convergent Charging Controller database

Subscriber not found in Convergent Charging Controller database flow

Here is an example message flow that illustrates the message sequence when a subscriber is not found in the Convergent Charging Controller database.



Subscriber not found in Convergent Charging Controller database scenario

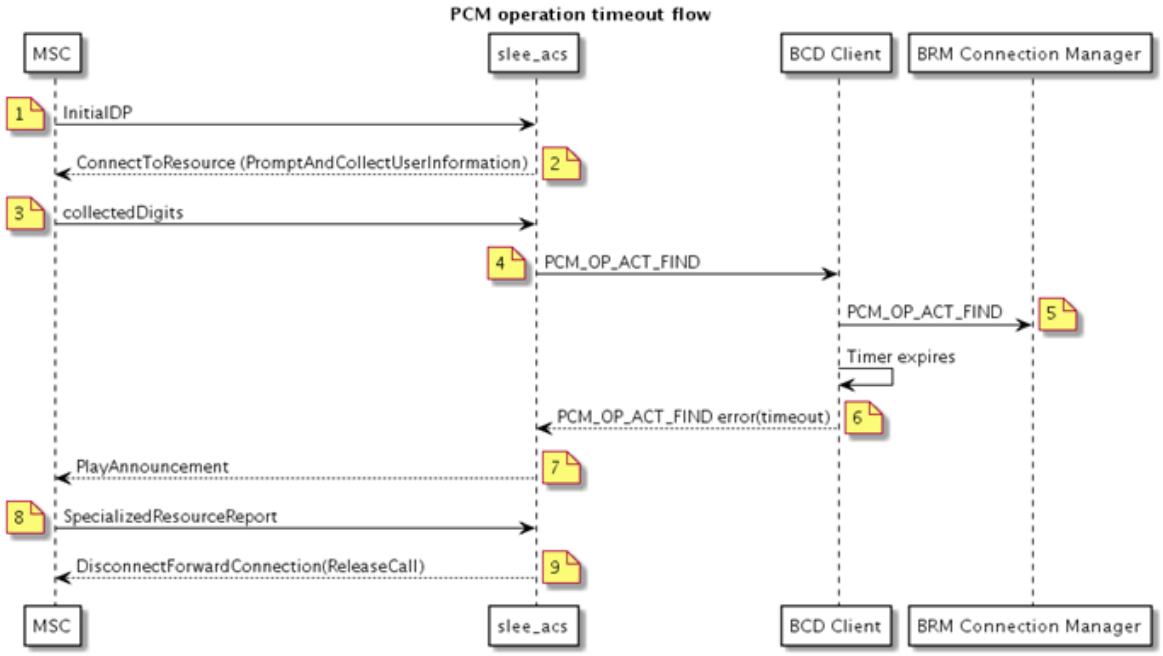
This scenario describes the sequence of messages that occurs when the subscriber is not found in the Convergent Charging Controller database.

Step	Action
1	• The subscriber makes a call. • MSC sends an InitialDP operation to the SLC • The slee_acs process receives the InitialDP and passes it to the CCS service loader. • The CCS service loader looks up the subscriber and wallet in the SCP database on the SLC and determines that the wallet information in on the BRM domain. The subscriber is not found and there is no default control plan to run.
2	• The slee_acs process sends a TC_END(ReleaseCall) message to MSC. • The call is disconnected.

PCM operation timeout

PCM operation timeout flow

Here is an example of message flow that illustrates the message sequence when a PCM operation timeout occurs.



PCM operation timeout scenario

This scenario describes the sequence of messages when a subscriber makes a call and PCM operation sent from BCD Client to BRM expires.

Step	Action
1	• The subscriber dials the “subscribe to service” toll free number. • MSC sends InitialDP to the slee_acs process, with serviceKey set to a special value indicating “subscribe to service”. • The CCS service loader searches for the subscriber and wallet in the SCP database on the SLC and determines the wallet information on the BRM domain. It also determines the control plan to run, using the serviceKey to select the “subscribe to service” control plan. • The slee_acs process runs the control plan. • The slee_acs process reaches a Selection Dependent Routing feature node.
2	• The Selection Dependent Routing feature node sends ConnectToResource (PromptAndCollectUserInformation) to the MSC, instructing the MSC to prompt for a single digit from a menu. • MSC plays the announcement to the caller and collects the menu choice.
3	• MSC sends PromptAndCollectUserInformation result, containing the menu choice digit, to the slee_acs process. • The SDR feature node takes an exit to a Named Event feature node and specifies a direct named event for a non-reservable event.

Step	Action
4	• The BCD actions library constructs a BcdSleeEvent to invoke an authorize operation and sends it to the least busy BCD Client. • The BCD Client finds a free connection – the one with the lowest proportion of its connections currently in use.
5	• The BCD Billing Client invokes the authorize operation via the PCM_OP_SEND() function and sets a timer to the configured value for this type of operation. • The operation timer expires.
6	• The BCD Client closes the connection and sends a BcdSleeEvent to the slee_acs process with a status indicating that the operation timed out. • The nevt feature node takes the Billing Fault exit and reaches a Play Announcement node for an error announcement.
7	• A Play Announcement feature node sends PlayAnnouncement to the MSC. • MSC plays an announcement to the caller, stating that the transaction was not successful.
8	• The announcement finishes and MSC sends SpecializedResourceReport to the slee_acs process.
9	• The control plan reaches an end node. ACS sends DisconnectForwardConnection and ReleaseCall actions to MSC and clears the call context. • The caller is disconnected.

Messages: PCM operation timeout

The following messages include operations sent to BRM and results returned by BRM for a PCM operation timeout. The general message format is: nesting level (0; 1, or 2); field; data type; value.

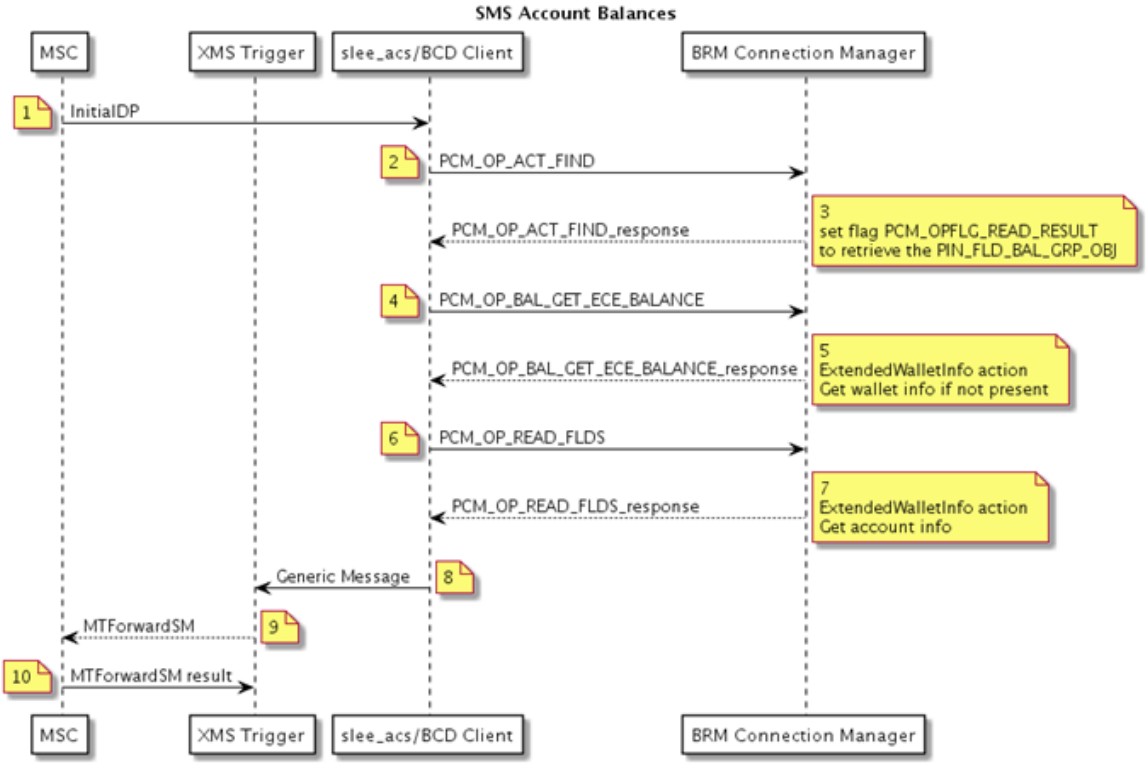
Operation: send PCM_OP_ACT_FIND (159)

```
Flags = 0
0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony -1 0
0 PIN_FLD_LOGIN           STR [0] "0049100701"
```

SMS account balance

SMS account balance flow

Here is an example of message flow to get account balances.



SMS account balance scenario

This scenario describes the sequence of messages that occurs when the subscriber calls to check the account balance.

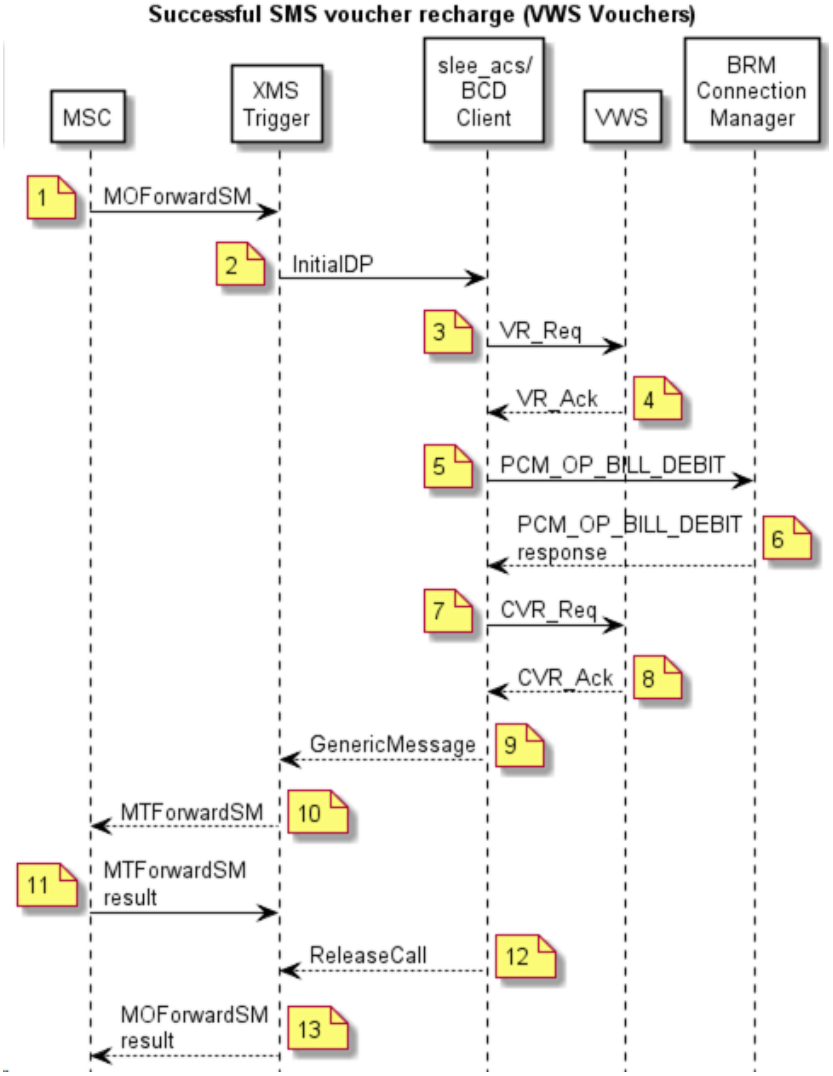
Action	Description
1	• The subscriber makes a voice call. • The MSC sends an InitialDP operation to the SLC.
2	• The slee_acs process on the SLC receives the InitialDP and passes it to the CCS service loader. • The CCS service loader searches for the subscriber and wallet in the SCP database on the SLC and determines that the wallet information is on the BRM domain. It also determines the control plan to run. • The slee_acs process runs the control plan. • The slee_acs process reaches an SMS Account Balances feature node. • The SMS Account Balances feature node invokes a GetWallet action on the FOX actions library. The action searches in the SCP database on the SLC and returns. (The FOX actions library is used because FOX is used for the subscriber domain and GetWallet is a subscriber domain action.) • The SMS Account Balances feature node invokes a WalletInfo action on the BCD actions library.

Action	Description
3.	• BRM Connection Manager responds to the operation indicating that the account has been successfully recharged. • The BRM Connection Manager sends the output flist from the operation in an event (BcdSleeEvent) to slee_acs. It also marks the BRM connection as free and cancels the operation timer.
4.	• The BCD actions library constructs a BcdSleeEvent to invoke PCM_OP_BAL_GET_ECE_BALANCE operation and sends it to the least busy BCD Client. • The BCD Client invokes the PCM_OP_BAL_GET_ECE_BALANCE operation and sets a timer to the configured value for this type of operation.
5.	• BRM responds to the operation. • The BCD Client receives the output flist and sends it in a BcdSleeEvent object to the slee_acs process. It also marks the BRM connection as free for reuse and cancels the operation timer. • The BCD actions library translates the output flist into the response of the WalletInfo action.
6.	• PCM_OP_READ_FLDS(Retrieve Account Details).
7.	• PCM_OP_READ_FLDS response.
8.	• The SMS Account Balances feature node constructs an MMX GenericMessage object containing balances information and sends it to xmsTrigger.
9.	• xmsTrigger sends an MTForwardSM operation to the MSC, containing balances information.
10.	• The MSC sends an SMS containing the balances information to the caller. • The slee_acs process takes the success branch of the SMS Account Balances feature node and reaches an end node.

Successful SMS recharge using a VWS voucher

Successful SMS recharge using a VWS voucher flow

Here is an example message flow for a scenario of a successful SMS recharge using a VWS voucher.



Successful SMS recharge using a VWS voucher scenario

This scenario describes the sequence of messages that occur during a successful SMS recharge using a VWS voucher. In this scenario, the account is on BRM while the voucher is in the Convergent Charging Controller VWS.

Step	Action
1	• MSC sends MOForwardSM to xmsTrigger. • The subscriber has sent an SMS, which contains the voucher and PIN, from a GSM phone to the SMS recharge number.

Step	Action
2	• xmsTrigger sends InitialDP to the slee_acs process, with the service key set to a special value indicating SMS recharge. • The slee_acs process on the SLC receives the InitialDP and passes it to the CCS service loader. • The CCS service loader looks up the subscriber and wallet in the SCP database on the SLC and determines that the wallet information is in the BRM domain. It also determines the control plan to run, based on the serviceKey. • The slee_acs process runs the SMS recharge control plan. • The slee_acs process reaches an Extract Content feature node, which extracts the voucher number and PIN from the SMS and places them in temporary storage. • The slee_acs process reaches a Voucher Recharge feature node, which extracts the voucher number and PIN from temporary storage and uses them to invoke a VoucherRedeem action on the FOX actions library.
3	• FOX actions library sends VR_Req to BeClient. • BeClient send VR_Req to VWS.
4	• VWS sends VR_Ack to BeClient, indicating that the voucher has been reserved and returning the recharge amounts for each balance type. • BeClient send VR_Ack to the slee_acs process. • The Voucher Recharge feature node invokes a WalletRecharge action on the BCD actions library.
5	• The slee_acs process creates an event (FlistSleeEvent) for a PCM_OP_BILL_DEBIT operation and sends it to the BCD Client • The BCD Client invokes the PCM_OP_BILL_DEBIT operation and sets a timer to the configured value for this type of operation.
6	• BRM responds to the operation indicating that the account has been successfully recharged. • The BCD Client receives the operation output flist and sends it in BcdSleeEvent to the slee_acs process. It also marks the BRM connection as free for reuse and cancels the operation timer. • The BCD actions library translates the output flist into the response of the WalletRecharge action. • The Voucher Recharge feature node invokes the VoucherConfirm action on FOX actions library.
7	• FOX actions library sends CVR_Req to BeClient. • BeClient sends CVR_Req to VWS. • VWS sends CVR_Ack to BeClient, indicating that the voucher has been permanently marked as redeemed.
8	• BeClient sends CVR_Ack to the slee_acs process. • The slee_acs process takes the success branch of the VoucherRecharge feature node and reaches a Send Short Message Notification feature node, which specifies sending a success text message.
9	• The SSMN feature node creates an MMX GenericMessage containing the success message and sends it to xmsTrigger.
10	• xmsTrigger sends an MTForwardSM operation to the MSC, containing the success message.
11	• MSC sends an SMS containing the success message to the caller.
12	• The slee_acs process reaches an Accept feature node which sends ReleaseCall to xmsTrigger.

Step	Action
13	• xmsTrigger sends an MOForwardSM result to the MSC. • The MSC sends a notification of successful delivery of the original SMS to the caller • The slee_acs process reaches an end node and clears the call connection.

Messages: successful SMS recharge using a VWS voucher

The following messages include operations sent to BRM and results returned by BRM for a VWS voucher recharge to a BRM account using SMS. The general message format is: nesting level (0; 1, or 2); field; data type; value.

Operation: send is PCM_OP_ACT_FIND (159)

```
Flags = 0
0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony -1 0
0 PIN_FLD_LOGIN           STR [0] "0049100701"
```

Result: received for operation PCM_OP_ACT_FIND (159)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony 91119 8
0 PIN_FLD_CREATED_T    TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
0 PIN_FLD_MOD_T        TSTAMP [0] (1485826647) Tue Jan 31 01:37:27 2017
0 PIN_FLD_READ_ACCESS     STR [0] "L"
0 PIN_FLD_WRITE_ACCESS    STR [0] "L"
0 PIN_FLD_AAC_ACCESS      STR [0] ""
0 PIN_FLD_AAC_PACKAGE     STR [0] ""
0 PIN_FLD_AAC_PROMO_CODE    STR [0] ""
0 PIN_FLD_AAC_SERIAL_NUM    STR [0] ""
0 PIN_FLD_AAC_SOURCE      STR [0] ""
0 PIN_FLD_AAC_VENDOR      STR [0] ""
0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 93423 0
0 PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 92655 0
0 PIN_FLD_CLOSE_WHEN_T TSTAMP [0] (0) <null>
0 PIN_FLD_EFFECTIVE_T  TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
0 PIN_FLD_ITEM_POID_LIST    STR [0] "0.0.0.1|/item/misc 100941 0"
0 PIN_FLD_LASTSTAT_CMNT    STR [0] ""
0 PIN_FLD_LAST_STATUS_T TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
0 8052                      INT [0] 0
0 PIN_FLD_LOGIN           STR [0] "893-20170130-173726-0-30615--152246576-den00bkf"
0 PIN_FLD_NAME            STR [0] "PIN Service Object"
0 PIN_FLD_NEXT_ITEM_POID_LIST    STR [0] ""
0 PIN_FLD_OBJECT_CACHE_TYPE   ENUM [0] 0
0 PIN_FLD_PASSWD          STR [0]
"sha1|50C2F101B640E2A5F7DB3F42C083EA52EE05516B|A6CBC7C27AD6BCE053D3BFBA7522E87E"
0 PIN_FLD_PASSWD_EXPIRATION_T TSTAMP [0] (0) <null>
0 PIN_FLD_PASSWD_STATUS   ENUM [0] 0
0 PIN_FLD_PROFILE_OBJ    POID [0] 0.0.0.0  0 0
0 PIN_FLD_SERVICE_ID      STR [0] ""
0 8053                 TSTAMP [0] (0) <null>
0 PIN_FLD_STATUS         ENUM [0] 10100
0 PIN_FLD_STATUS_FLAGS    INT [0] 0
0 PIN_FLD_SUBSCRIPTION_OBJ   POID [0] 0.0.0.0  0 0
0 PIN_FLD_TYPE           ENUM [0] 0
0 PIN_FLD_ALIAS_LIST    ARRAY [0] allocated 20, used 1
1     PIN_FLD_NAME            STR [0] "0049100701"
0 PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 1
1     PIN_FLD_PRIMARY_NUMBER    INT [0] 0
0 PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 2
1     PIN_FLD_BEARER_SERVICE    STR [0] "Bearer Service"
1     PIN_FLD_IMEI              STR [0] "IMEI"
```

Operation: send is PCM_OP_BILL_DEBIT (105)

```
Flags = 0
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
0 PIN_FLD_PROGRAM_NAME    STR [0] "NCC_BCD_Client"
0 PIN_FLD_OBJ_TYPE        STR [0] "gsm/ncc"
0 PIN_FLD_DEBIT         ARRAY [978] allocated 20, used 1
```

```
1     PIN_FLD_BAL_OPERAND  DECIMAL [0] -0.500000000000000
```

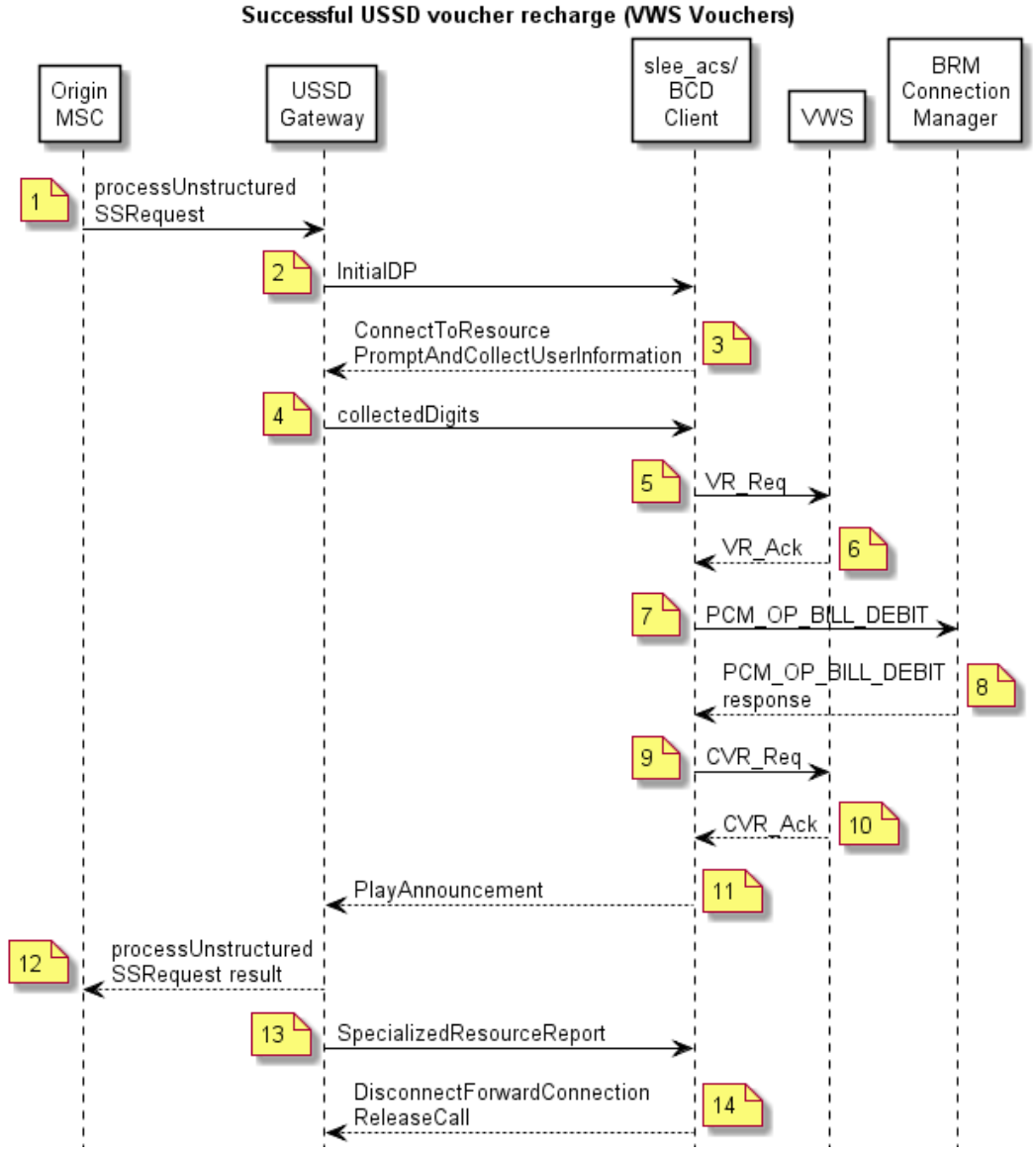
Result: received for operation PCM_OP_BILL_DEBIT (105)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
0 PIN_FLD_RESULTS       ARRAY [0] allocated 20, used 4
1     PIN_FLD_BAL_IMPACTS   ARRAY [0] allocated 20, used 17
2         PIN_FLD_IMPACT_TYPE    ENUM [0] 2
2         PIN_FLD_RESOURCE_ID     INT [0] 978
2         PIN_FLD_RESOURCE_ID_ORIG    INT [0] 0
2         PIN_FLD_TAX_CODE        STR [0] ""
2         PIN_FLD_RATE_TAG        STR [0] ""
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
2         PIN_FLD_RATE_OBJ       POID [0] 0.0.0.0  0 0
2         PIN_FLD_DISCOUNT     DECIMAL [0] 0
2         PIN_FLD_PERCENT      DECIMAL [0] 0
2         PIN_FLD_QUANTITY     DECIMAL [0] 0
2         PIN_FLD_AMOUNT_DEFERRED DECIMAL [0] 0
2         PIN_FLD_AMOUNT       DECIMAL [0] -0.500
2         PIN_FLD_AMOUNT_ORIG  DECIMAL [0] NULL pin_decimal_t ptr
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 127667 0
2         PIN_FLD_GL_ID           INT [0] 0
2         PIN_FLD_ITEM_OBJ       POID [0] 0.0.0.1 /item/misc 129715 1466
2         PIN_FLD_LINEAGE         STR [0] NULL str ptr
1     PIN_FLD_SUB_BAL_IMPACTS  ARRAY [0] allocated 20, used 3
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 127667 6447
2         PIN_FLD_RESOURCE_ID     INT [0] 978
2         PIN_FLD_SUB_BALANCES  ARRAY [0] allocated 20, used 3
3             PIN_FLD_AMOUNT       DECIMAL [0] -0.500
3             PIN_FLD_VALID_FROM   TSTAMP [0] (0) <null>
3             PIN_FLD_VALID_TO     TSTAMP [0] (0) <null>
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/billing/debit 272643699316523851 0
```

Successful USSD recharge using a VWS voucher

Successful USSD recharge using a VWS voucher flow

Here is an example message flow for a scenario of a successful USSD recharge using a VWS voucher.



Successful USSD recharge using a VWS voucher scenario

This scenario describes the sequence of messages that occurs for successful USSD recharge using a VWS voucher. In this scenario, the account is on BRM while the voucher is in the Convergent Charging Controller VWS.

Step	Action
1	• MSC sends processUnstructuredSS-Request to USSD Gateway • The calling subscriber has sent a USSD message that contains the service code for a voucher recharge, the voucher number and PIN.

Step	Action
2	• The USSD Gateway sends InitialDP to the slee_acs process, using the USSD voucher recharge service key as determined by the service code configuration. • The CCS service loader looks up the subscriber and wallet in the SCP database on the SLC and determines that the wallet information is on the BRM domain. It also determines the control plan to run, using the serviceKey to select the USSD voucher recharge control plan. • The slee_acs process runs the control plan. • The slee_acs process reaches a Voucher Recharge feature node.
3	• The Voucher Recharge feature node sends ConnectToResource, PromptAndCollectUserInformation to USSD Gateway, instructing it to prompt the caller for the voucher number and PIN. • USSD Gateway retrieves the voucher number and PIN from the received processUnstructuredSS-Request.
4	• USSD Gateway sends the PromptAndCollectUserInformation result, containing the voucher number and PIN, to the slee_acs process. • The Voucher Recharge feature node invokes the VoucherRedeem action on the FOX actions library.
5	• The FOX actions library sends VR_Req to BeClient • BeClient sends VR_Req to VWS
6	• VWS sends VR_Ack to BeClient, indicating that the voucher has been reserved, and returning the recharge amounts for each balance type. • BeClient sends VR_Ack to the slee_acs process. • The Voucher Recharge feature node invokes the WalletRecharge action on the BCD actions library.
7	• The slee_acs process creates an FlistSleeEvent for a PCM_OP_BILL_DEBIT operation and sends it to the BCD Client. • The BCD actions library creates an event (FlistSleeEvent) that contains a PCM_OP_BILL_DEBIT operation and sends it to the BCD Client • The BCD Client invokes the PCM_OP_BILL_DEBIT operation and sets a timer to the configured value for this type of operation.
8	• BRM responds to the operation indicating that the account has been successfully recharged. • The BCD Client receives the operation output flist and sends it in an event (BcdSleeEvent) to the slee_acs process. It also marks the BRM connection as free for reuse and cancels the operation timer. • The BCD actions library translates the output flist into the response of the WalletRecharge action. • The Voucher Recharge feature node invokes the Voucher Confirm action on the FOX actions library.
9	• The FOX actions library sends CVR_Req to BeClient • BeClient sends CVR_Req to VWS.
10	• VWS sends CVR_Ack to BeClient, indicating that the voucher has been permanently marked as redeemed. • BeClient sends CVR_Ack to the slee_acs process.
11	• The Voucher Recharge feature node sends PlayAnnouncement to USSD Gateway. • USSD Gateway translates the announcement into text that states that the recharge was successful, based on configuration.

Step	Action
12	• USSD Gateway sends processUnstructuredSS-Request result to the MSC. • USSD text, which states that the recharge was successful, is displayed on the caller's handset.
13	• USSD Gateway sends SpecializedResouceReport to the slee_acs process.
14	• The control plan reaches an end node and ACS sends DisconnectForwardConnection and ReleaseCall actions to USSD Gateway.

Messages: successful USSD recharge using a VWS voucher

The following messages include operations sent to BRM and results returned by BRM for a VWS voucher recharge to a BRM account using USSD. The general message format is: nesting level (0; 1, or 2); field; data type; value.

Operation: send is PCM_OP_ACT_FIND (159)

```
Flags = 0
0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony -1 0
0 PIN_FLD_LOGIN           STR [0] "0049100701"
```

Result: received for operation PCM_OP_ACT_FIND (159)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony 91119 8
0 PIN_FLD_CREATED_T    TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
0 PIN_FLD_MOD_T        TSTAMP [0] (1485826647) Tue Jan 31 01:37:27 2017
0 PIN_FLD_READ_ACCESS     STR [0] "L"
0 PIN_FLD_WRITE_ACCESS    STR [0] "L"
0 PIN_FLD_AAC_ACCESS      STR [0] ""
0 PIN_FLD_AAC_PACKAGE     STR [0] ""
0 PIN_FLD_AAC_PROMO_CODE    STR [0] ""
0 PIN_FLD_AAC_SERIAL_NUM    STR [0] ""
0 PIN_FLD_AAC_SOURCE      STR [0] ""
0 PIN_FLD_AAC_VENDOR      STR [0] ""
0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 93423 0
0 PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 92655 0
0 PIN_FLD_CLOSE_WHEN_T TSTAMP [0] (0) <null>
0 PIN_FLD_EFFECTIVE_T  TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
0 PIN_FLD_ITEM_POID_LIST    STR [0] "0.0.0.1|/item/misc 100941 0"
0 PIN_FLD_LASTSTAT_CMNT    STR [0] ""
0 PIN_FLD_LAST_STATUS_T TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
0 8052                     INT [0] 0
0 PIN_FLD_LOGIN           STR [0] "893-20170130-173726-0-30615--152246576-den00bkf"
0 PIN_FLD_NAME            STR [0] "PIN Service Object"
0 PIN_FLD_NEXT_ITEM_POID_LIST    STR [0] ""
0 PIN_FLD_OBJECT_CACHE_TYPE   ENUM [0] 0
0 PIN_FLD_PASSWD          STR [0]
"sha1|50C2F101B640E2A5F7DB3F42C083EA52EE05516B|A6CBC7C27AD6BCE053D3BFBA7522E87E"
0 PIN_FLD_PASSWD_EXPIRATION_T TSTAMP [0] (0) <null>
0 PIN_FLD_PASSWD_STATUS   ENUM [0] 0
0 PIN_FLD_PROFILE_OBJ    POID [0] 0.0.0.0  0 0
0 PIN_FLD_SERVICE_ID      STR [0] ""
0 8053                 TSTAMP [0] (0) <null>
0 PIN_FLD_STATUS         ENUM [0] 10100
0 PIN_FLD_STATUS_FLAGS    INT [0] 0
0 PIN_FLD_SUBSCRIPTION_OBJ   POID [0] 0.0.0.0  0 0
  0 PIN_FLD_TYPE           ENUM [0] 0
0 PIN_FLD_ALIAS_LIST    ARRAY [0] allocated 20, used 1
1     PIN_FLD_NAME            STR [0] "0049100701"
0 PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 1
1     PIN_FLD_PRIMARY_NUMBER    INT [0] 0
0 PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 2
1     PIN_FLD_BEARER_SERVICE    STR [0] "Bearer Service"
1     PIN_FLD_IMEI              STR [0] "IMEI"
```

Operation: send is PCM_OP_BILL_DEBIT (105)

```
Flags = 0
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
0 PIN_FLD_PROGRAM_NAME    STR [0] "NCC_BCD_Client"
0 PIN_FLD_OBJ_TYPE        STR [0] "gsm/ncc"
0 PIN_FLD_DEBIT         ARRAY [978] allocated 20, used 1
1     PIN_FLD_BAL_OPERAND  DECIMAL [0] -0.500000000000000
```

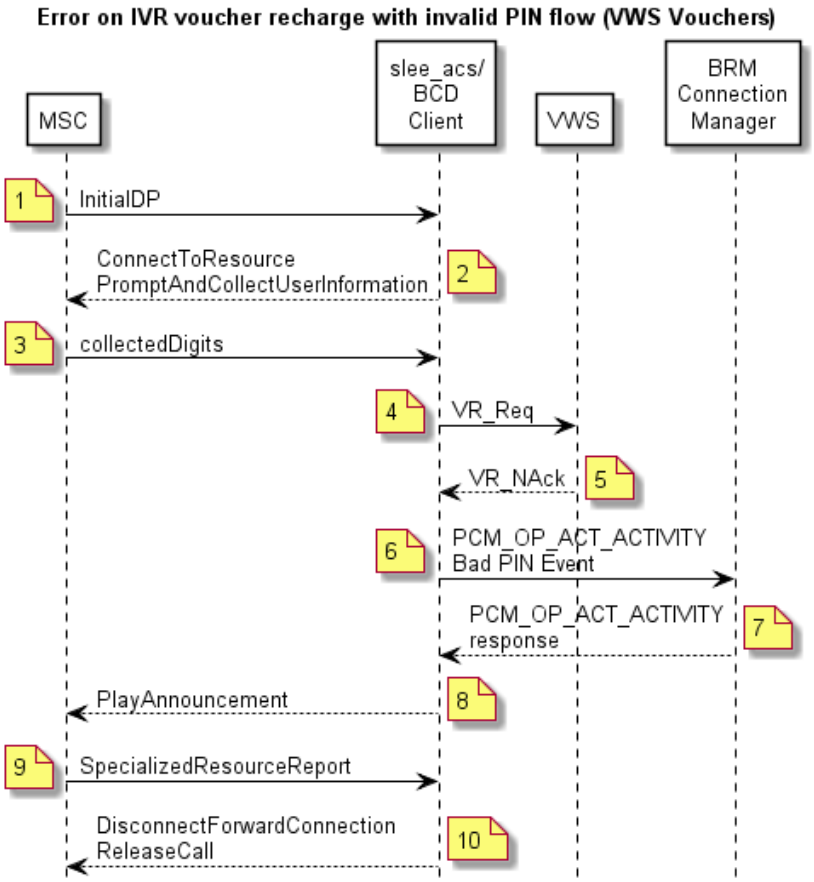
Result: received for operation PCM_OP_BILL_DEBIT (105)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
0 PIN_FLD_RESULTS       ARRAY [0] allocated 20, used 4
1     PIN_FLD_BAL_IMPACTS   ARRAY [0] allocated 20, used 17
2         PIN_FLD_IMPACT_TYPE    ENUM [0] 2
2         PIN_FLD_RESOURCE_ID     INT [0] 978
2         PIN_FLD_RESOURCE_ID_ORIG    INT [0] 0
2         PIN_FLD_TAX_CODE         STR [0] ""
2         PIN_FLD_RATE_TAG         STR [0] ""
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
2         PIN_FLD_RATE_OBJ       POID [0] 0.0.0.0  0 0
2         PIN_FLD_DISCOUNT     DECIMAL [0] 0
2         PIN_FLD_PERCENT      DECIMAL [0] 0
2         PIN_FLD_QUANTITY     DECIMAL [0] 0
2         PIN_FLD_AMOUNT_DEFERRED DECIMAL [0] 0
2         PIN_FLD_AMOUNT       DECIMAL [0] -0.500
2         PIN_FLD_AMOUNT_ORIG  DECIMAL [0] NULL pin_decimal_t ptr
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 127667 0
2         PIN_FLD_GL_ID           INT [0] 0
2         PIN_FLD_ITEM_OBJ       POID [0] 0.0.0.1 /item/misc 129715 1466
2         PIN_FLD_LINEAGE         STR [0] NULL str ptr
1     PIN_FLD_SUB_BAL_IMPACTS  ARRAY [0] allocated 20, used 3
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 127667 6447
2         PIN_FLD_RESOURCE_ID     INT [0] 978
2         PIN_FLD_SUB_BALANCES  ARRAY [0] allocated 20, used 3
3             PIN_FLD_AMOUNT       DECIMAL [0] -0.500
3             PIN_FLD_VALID_FROM   TSTAMP [0] (0) <null>
3             PIN_FLD_VALID_TO     TSTAMP [0] (0) <null>
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/billing/debit 272643699316523851 0
```

Error on IVR Recharge using a VWS voucher with invalid PIN

Error on IVR Recharge using a VWS voucher with invalid PIN flow

Here is an example message flow for an error on an IVR Recharge using a VWS voucher with an invalid PIN.



Error on IVR recharge using a VWS voucher with invalid PIN scenario

This scenario describes the sequence of messages that occurs for an error when the subscriber calls the toll free number to request a voucher recharge but the subscriber's PIN is invalid. In this scenario, the account is on BRM while the voucher is in the Convergent Charging Controller VWS.

Step	Action
1	• The subscriber calls the toll free number for a voucher recharge • MSC sends InitialDP to the slee_acs process with serviceKey set to a special value indicating voucher recharge. • xmsTrigger sends InitialDP to the slee_acs process.

Step	Action
2	• The CCS service loader looks up the subscriber and wallet in the SCP database on the SLC and determines that the wallet information is on the BRM domain and VWS is used for the voucher. It also uses the serviceKey to select the voucher recharge control plan. • The slee_acs process runs the control plan. • The slee_acs process reaches a Voucher Recharge feature node. • The Voucher Recharge feature node sends ConnectToResource, PromptAndCollectUserInformation to the MSC, instructing the MSC to prompt the caller for the voucher number and PIN.
3	• MSC plays the specified announcement to the caller and collects the voucher number and PIN. • MSC sends PromptAndCollectUserInformation result, containing voucher number and PIN to the slee_acs process.
4	• The Voucher Recharge feature node invokes the VoucherRedeem action on the FOX actions library. • FOX actions library sends VR_Req to BeClient. • BeClient sends VR_Req to VWS.
5	• VWS sends VR_Nack to BeClient, indicating that the voucher PIN is incorrect. • The Voucher Recharge feature node invokes the VoucherBadPIN action on the BCD actions library. • The VoucherBadPIN action does nothing and returns success. • The Voucher Recharge feature node invokes the VoucherCreateEDR action on the BCD actions library to create a record for the bad PIN attempt.
6	• The VoucherCreateEDR action creates an event (FlistSleeEvent) that contains a PCM_OP_ACT_ACTIVITY operation for the special voucher bad PIN event and sends it to BCD Client. • The BCD Client invokes the PCM_OP_ACT_ACTIVITY operation and sets a timer to the configured value for this type of operation.
7	• BRM creates a record for the bad PIN event and responds to the operation by indicating that this has been done. • The BCD Client receives the operation output flist and sends it in an event (BcdSleeEvent) to the slee_acs process. It also marks the BRM connection as free for reuse and cancels the operation timer. • The BCD actions library translates the output flist into the response of the VoucherCreateEDR action. **Note**:Recording bad PIN attempts would allow BRM to freeze accounts against which a large number of bad PIN attempts have been made. Some customization would be required, however, for BRM to freeze an account.
8	• The Voucher Recharge feature node sends PlayAnnouncement to the MSC. • MSC plays an announcement to the caller, stating that the recharge was unsuccessful.
9	• The announcement finishes and MSC sends SpecializedResourceReport to the slee_acs process.
10	• The control plan reaches an end node and ACS sends DisconnectForwardConnection and ReleaseCall actions to the MSC and clears the call context. • The caller is disconnected.

Messages: error on IVR recharge using a VWS voucher with an invalid PIN

The following messages include operations sent to BRM and results returned by BRM for an IVR recharge using a VWS voucher with an invalid PIN. The general message format is: nesting level (0; 1, or 2); field; data type; value.

Operation: send is PCM_OP_ACT_FIND (159)

```
Flags = 0
0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony -1 0
0 PIN_FLD_LOGIN           STR [0] "0049100701"
```

Result: received for operation PCM_OP_ACT_FIND (159)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony 91119 8
0 PIN_FLD_CREATED_T    TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
0 PIN_FLD_MOD_T        TSTAMP [0] (1485826647) Tue Jan 31 01:37:27 2017
0 PIN_FLD_READ_ACCESS     STR [0] "L"
0 PIN_FLD_WRITE_ACCESS    STR [0] "L"
0 PIN_FLD_AAC_ACCESS      STR [0] ""
0 PIN_FLD_AAC_PACKAGE     STR [0] ""
0 PIN_FLD_AAC_PROMO_CODE    STR [0] ""
0 PIN_FLD_AAC_SERIAL_NUM    STR [0] ""
0 PIN_FLD_AAC_SOURCE      STR [0] ""
0 PIN_FLD_AAC_VENDOR      STR [0] ""
0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 93423 0
0 PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 92655 0
0 PIN_FLD_CLOSE_WHEN_T TSTAMP [0] (0) <null>
0 PIN_FLD_EFFECTIVE_T  TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
0 PIN_FLD_ITEM_POID_LIST    STR [0] "0.0.0.1|/item/misc 100941 0"
0 PIN_FLD_LASTSTAT_CMNT    STR [0] ""
0 PIN_FLD_LAST_STATUS_T TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
0 8052                     INT [0] 0
0  PIN_FLD_LOGIN           STR [0] "893-20170130-173726-0-30615--152246576-den00bkf"
0 PIN_FLD_NAME            STR [0] "PIN Service Object"
0 PIN_FLD_NEXT_ITEM_POID_LIST    STR [0] ""
0 PIN_FLD_OBJECT_CACHE_TYPE   ENUM [0] 0
0 PIN_FLD_PASSWD          STR [0]
"sha1|50C2F101B640E2A5F7DB3F42C083EA52EE05516B|A6CBC7C27AD6BCE053D3BFBA7522E87E"
0 PIN_FLD_PASSWD_EXPIRATION_T TSTAMP [0] (0) <null>
0 PIN_FLD_PASSWD_STATUS   ENUM [0] 0
0 PIN_FLD_PROFILE_OBJ    POID [0] 0.0.0.0  0 0
0 PIN_FLD_SERVICE_ID      STR [0] ""
0 8053                 TSTAMP [0] (0) <null>
0 PIN_FLD_STATUS         ENUM [0] 10100
0 PIN_FLD_STATUS_FLAGS    INT [0] 0
0 PIN_FLD_SUBSCRIPTION_OBJ   POID [0] 0.0.0.0  0 0
0 PIN_FLD_TYPE            ENUM [0] 0
0 PIN_FLD_ALIAS_LIST    ARRAY [0] allocated 20, used 1
1     PIN_FLD_NAME            STR [0] "0049100701"
0 PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 1
1     PIN_FLD_PRIMARY_NUMBER    INT [0] 0
0 PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 2
1     PIN_FLD_BEARER_SERVICE    STR [0] "Bearer Service"
1     PIN_FLD_IMEI            STR [0] "IMEI"
```

Operation: send is PCM_OP_ACT_ACTIVITY (151)

```
Flags = 0
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
0 PIN_FLD_PROGRAM_NAME    STR [0] "NCC_BCD_Client"
0 PIN_FLD_OBJ_TYPE        STR [0] "/voucher"
0 PIN_FLD_INHERITED_INFO SUBSTRUCT [0] allocated 20, used 1
1     10000               SUBSTRUCT [0] allocated 20, used 2
2         10007                   STR [0] "9876543214"
2         10008                   STR [0] "3210"
0 PIN_FLD_BAL_IMPACTS   ARRAY [1000011] allocated 20, used 4
1     PIN_FLD_AMOUNT       DECIMAL [0] 1
1     PIN_FLD_RESOURCE_ID     INT [0] 1000011
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
```

```
1     PIN_FLD_IMPACT_TYPE    ENUM [0] 2
```

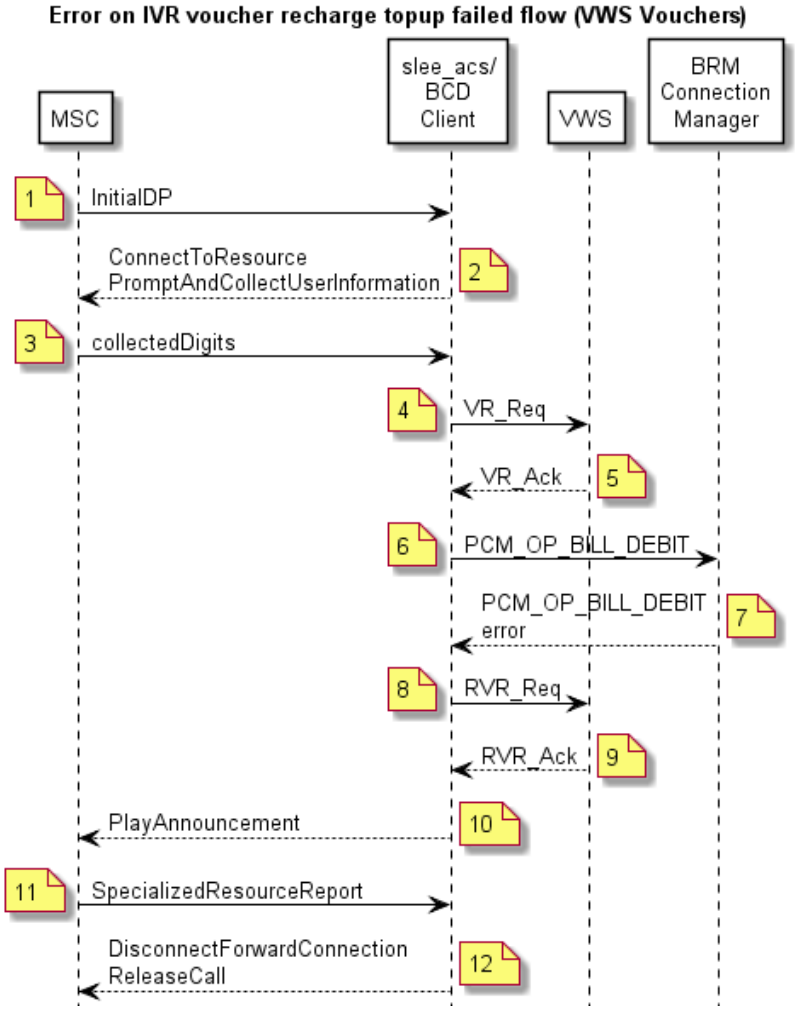
Result: received for operation PCM_OP_ACT_ACTIVITY (151)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
0 PIN_FLD_RESULTS       ARRAY [0] allocated 20, used 4
1     PIN_FLD_BAL_IMPACTS   ARRAY [1000011] allocated 20, used 8
2         PIN_FLD_AMOUNT       DECIMAL [0] 1
2         PIN_FLD_RESOURCE_ID     INT [0] 1000011
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
2         PIN_FLD_IMPACT_TYPE    ENUM [0] 2
2         PIN_FLD_GL_ID           INT [0] 0
2         PIN_FLD_ITEM_OBJ       POID [0] 0.0.0.1 /item/misc 175577 0
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 127667 6530
2         PIN_FLD_LINEAGE         STR [0] NULL str ptr
1     PIN_FLD_SUB_BAL_IMPACTS  ARRAY [1000011] allocated 20, used 3
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 127667 6531
2         PIN_FLD_RESOURCE_ID     INT [0] 1000011
2         PIN_FLD_SUB_BALANCES  ARRAY [4] allocated 20, used 3
3             PIN_FLD_AMOUNT       DECIMAL [0] 1
3             PIN_FLD_VALID_FROM   TSTAMP [0] (0) <null>
3             PIN_FLD_VALID_TO     TSTAMP [0] (0) <null>
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/activity/voucher 272643699316532185 0
```

Error on IVR recharge account topup failed

Error on IVR recharge account topup failed flow

Here is an example message flow for an IVR Recharge in which an account topup failed.



Error on IVR recharge account topup failed scenario

This scenario describes the sequence of messages that occurs when a subscriber calls a toll-free number to request a voucher recharge but the action to recharge the subscriber's wallet on BRM fails. In this scenario, the account is on BRM while the voucher is in the Convergent Charging Controller VWS.

Step	Action
1	• The subscriber calls the toll free number for a voucher recharge. • MSC sends InitialDP to the slee_acs process with serviceKey set to a special value indicating voucher recharge.

Step	Action
2	• The CCS service loader looks up the subscriber and wallet in the SCP database on the SLC and determines that the wallet information is on the BRM domain and VWS is used for the voucher. It also uses the serviceKey to select the voucher recharge control plan. • The slee_acs process runs the control plan. • The slee_acs process reaches a Voucher Recharge feature node. • The Voucher Recharge feature node sends ConnectToResource, PromptAndCollectUserInformation to the MSC, instructing the MSC to prompt the caller for the voucher number and PIN.
3	• MSC plays the specified announcement to the caller and collects the voucher number and PIN. • MSC sends PromptAndCollectUserInformation result, containing voucher number and PIN to the slee_acs process.
4	• The Voucher Recharge feature node invokes the VoucherRedeem action on the FOX actions library. • FOX actions library sends VR_Req to BeClient. • BeClient sends VR_Req to VWS.
5	• VWS sends VR_Ack to BeClient, indicating that the voucher has been reserved, and returning the recharge amounts for each balance type. • BeClient sends VR_Ack to the slee_acs process.
6	• The Voucher Recharge feature node invokes the WalletRecharge action on the BCD actions library. • The BCD actions library constructs an event (BcdSleeEvent) containing PCM_OP_BILL_DEBIT and sends it to the BCD Client. • The BCD Client invokes the PCM_OP_BILL_DEBIT operation and sets a timer to the configured value for this type of operation.
7	• BRM responds to the operation by indicating that the account has not been recharged. • The BCD Client receives the operation output flist and sends it in an event (BcdSleeEvent) to the slee_acs process. It also marks the BRM connection as free for reuse and cancels the operation timer. • The BCD actions library translates the output flist into the error response of the WalletRecharge action. • The Voucher Recharge feature node invokes the VoucherRevoke action on the FOX actions library.
8	• The FOX actions library sends RVR_Req to BeClient. • BeClient sends RVR_Req to VWS.
9	• VWS sends RVR_Ack to BeClient, indicating that the voucher reservation has been revoked. • BeClient sends CVR_Ack to the slee_acs process.
10	• The Voucher Recharge feature node sends PlayAnnouncement to the MSC. • MSC plays an announcement to the caller, stating that the recharge was not successful.
11	• The announcement finishes and MSC sends SpecializedResourceReport to the slee_acs process.
12	• The control plan reaches an end node and ACS sends DisconnectForwardConnection and ReleaseCall actions to the MSC and clears the call connection. • The caller is disconnected.

Messages: Error on IVR recharge account topup failed

The following messages include operations sent to BRM and results returned by BRM for an IVR recharge when the account topup failed. The general message format is: nesting level (0; 1, or 2); field; data type; value.

Operation: send is PCM_OP_ACT_FIND (159)

```
Flags = 0
0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony -1 0
0 PIN_FLD_LOGIN           STR [0] "0049100701"
```

Result: received for operation PCM_OP_ACT_FIND (159)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony 91119 8
0 PIN_FLD_CREATED_T    TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
0 PIN_FLD_MOD_T        TSTAMP [0] (1485826647) Tue Jan 31 01:37:27 2017
0 PIN_FLD_READ_ACCESS     STR [0] "L"
0 PIN_FLD_WRITE_ACCESS    STR [0] "L"
0 PIN_FLD_AAC_ACCESS      STR [0] ""
0 PIN_FLD_AAC_PACKAGE     STR [0] ""
0 PIN_FLD_AAC_PROMO_CODE    STR [0] ""
0 PIN_FLD_AAC_SERIAL_NUM    STR [0] ""
0 PIN_FLD_AAC_SOURCE      STR [0] ""
0 PIN_FLD_AAC_VENDOR      STR [0] ""
0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 93423 0
0 PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 92655 0
0 PIN_FLD_CLOSE_WHEN_T TSTAMP [0] (0) <null>
0 PIN_FLD_EFFECTIVE_T  TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
0 PIN_FLD_ITEM_POID_LIST    STR [0] "0.0.0.1|/item/misc 100941 0"
0 PIN_FLD_LASTSTAT_CMNT    STR [0] ""
0 PIN_FLD_LAST_STATUS_T TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
0 8052                     INT [0] 0
0 PIN_FLD_LOGIN           STR [0] "893-20170130-173726-0-30615--152246576-den00bkf"
0 PIN_FLD_NAME            STR [0] "PIN Service Object"
0 PIN_FLD_NEXT_ITEM_POID_LIST    STR [0] ""
0 PIN_FLD_OBJECT_CACHE_TYPE   ENUM [0] 0
0 PIN_FLD_PASSWD          STR [0]
"sha1|50C2F101B640E2A5F7DB3F42C083EA52EE05516B|A6CBC7C27AD6BCE053D3BFBA7522E87E"
0 PIN_FLD_PASSWD_EXPIRATION_T TSTAMP [0] (0) <null>
0 PIN_FLD_PASSWD_STATUS   ENUM [0] 0
0 PIN_FLD_PROFILE_OBJ    POID [0] 0.0.0.0  0 0
0 PIN_FLD_SERVICE_ID      STR [0] ""
0 8053                 TSTAMP [0] (0) <null>
0 PIN_FLD_STATUS         ENUM [0] 10100
0 PIN_FLD_STATUS_FLAGS    INT [0] 0
0 PIN_FLD_SUBSCRIPTION_OBJ   POID [0] 0.0.0.0  0 0
0 PIN_FLD_TYPE            ENUM [0] 0
0 PIN_FLD_ALIAS_LIST    ARRAY [0] allocated 20, used 1
1     PIN_FLD_NAME            STR [0] "0049100701"
0 PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 1
1     PIN_FLD_PRIMARY_NUMBER    INT [0] 0
0 PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 2
1     PIN_FLD_BEARER_SERVICE    STR [0] "Bearer Service"
1     PIN_FLD_IMEI              STR [0] "IMEI"
```

Operation: send is PCM_OP_BILL_DEBIT (105)

```
Flags = 0
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
0 PIN_FLD_PROGRAM_NAME    STR [0] "NCC_BCD_Client"
0 PIN_FLD_OBJ_TYPE        STR [0] "gsm/ncc"
0 PIN_FLD_DEBIT         ARRAY [978] allocated 20, used 1
1     PIN_FLD_BAL_OPERAND  DECIMAL [0] -0.500000000000000
```

Result: received for operation PCM_OP_BILL_DEBIT(105)

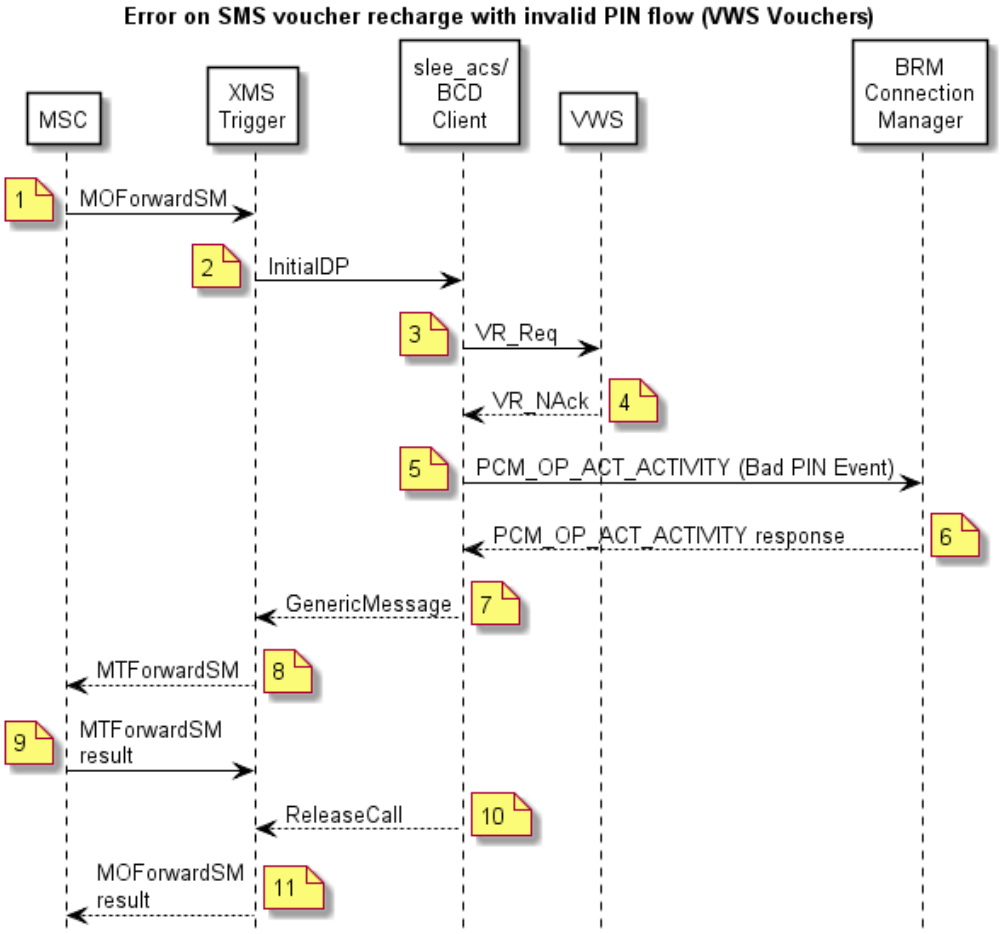
```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /error_poid 128819 0

Error specified:
 location=5
```

```
pin_errclass=4
pin_err=3
field=83893688
rec_id=0
reserved=0
facility=0
msg_id=0
version=0
```

Error on SMS recharge with invalid PIN

Error on SMS recharge with invalid PIN flow

Here is an example message flow for an error on an SMS recharge with an invalid PIN.



Error on SMS recharge with invalid PIN scenario

This scenario describes the sequence of messages that occurs when a subscriber's attempt to recharge a voucher fails due to an invalid PIN and MSC sends a short text message to inform the caller. In this scenario, the account is on BRM while the voucher is in the Convergent Charging Controller VWS.

Step	Action
1	• MSC sends MOForwardSM to xmsTrigger. • The subscriber has sent an SMS, which contains the voucher and PIN, from a GSM phone to the SMS recharge number.

Step	Action
2	• xmsTrigger sends InitialDP to the slee_acs process, with the service key set to a special value indicating SMS recharge. • The slee_acs process on the SLC receives the InitialDP and passes it to the CCS service loader. • The CCS service loader looks up the subscriber and wallet in the SCP database on the SLC and determines that the wallet information is in the BRM domain. It also determines the control plan to run, based on the serviceKey. • The slee_acs process runs the SMS recharge control plan. • The slee_acs process reaches an Extract Content feature node, which extracts the voucher number and PIN from the SMS and places them in temporary storage. • The slee_acs process reaches a Voucher Recharge feature node, which extracts the voucher number and PIN from temporary storage and uses them to invoke a VoucherRedeem action on the FOX actions library.
3	• FOX actions library sends VR_Req to BeClient. • BeClient send VR_Req to VWS.
4	• VWS sends VR_Nack to BeClient, indicating that the voucher PIN is incorrect. • The Voucher Recharge feature node invokes a VoucherBadPIN action on the BCD actions library. • The VoucherBadPIN action does nothing and returns success. • The Voucher Recharge feature node invokes the VoucherCreateEDR action on the BCD actions library, to create a record for the bad PIN attempt.
5	• VoucherCreateEDR action constructs an event (FlistSleeEvent) that contains a PCM_OP_ACT_ACTIVITY operation for the special voucher bad PIN event and sends it to BCD Client. • The BCD Client invokes the PCM_OP_ACT_ACTIVITY operation and sets a timer to the configured value for this type of operation.
6	• BRM creates a record for the bad PIN event and responds to the operation by indicating that this has been done. • The BCD Client receives the operation output flist and sends it in an event (BcdSleeEvent) to the slee_acs process. It also marks the BRM connection as free for reuse and cancels the operation timer. • The BCD actions library translates the output flist into the response of the VoucherCreateEDR action. • The slee_acs process takes the Voucher Invalid branch of the Voucher Recharge feature node and reaches a Send Short Message Notification feature node, which specifies sending a failure text message.
7	• The SSMN feature node creates an MMX GenericMessage containing the failure message and sends it to xmsTrigger .
8	• xmsTrigger sends an MTForwardSM operation to the MSC, containing the failure message.
9	• The MSC sends an SMS containing the failure message to the caller.
10	• The slee_acs process reaches an Accept feature node, which sends ReleaseCall to xmsTrigger.
11	• xmsTrigger sends a MOForwardSM result to the MSC. • The MSC sends a notification of successful delivery of the original SMS to the caller. • The slee_acs process reaches an end node and clears the call context.

Messages: error on SMS recharge with invalid PIN

The following messages include operations sent to BRM and results returned by BRM when an error occurs for a voucher recharge with an invalid PIN using SMS. The general message format is: nesting level (0; 1, or 2); field; data type; value.

Operation: send is PCM_OP_ACT_FIND (159)

```
 Flags = 0
 0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony -1 0
 0 PIN_FLD_LOGIN           STR [0] "0049100701"
```

Result: received for operation PCM_OP_ACT_FIND (159)

```
 0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony 91119 8
 0 PIN_FLD_CREATED_T    TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
 0 PIN_FLD_MOD_T        TSTAMP [0] (1485826647) Tue Jan 31 01:37:27 2017
 0 PIN_FLD_READ_ACCESS     STR [0] "L"
 0 PIN_FLD_WRITE_ACCESS    STR [0] "L"
 0 PIN_FLD_AAC_ACCESS      STR [0] ""
 0 PIN_FLD_AAC_PACKAGE     STR [0] ""
 0 PIN_FLD_AAC_PROMO_CODE    STR [0] ""
 0 PIN_FLD_AAC_SERIAL_NUM    STR [0] ""
 0 PIN_FLD_AAC_SOURCE      STR [0] ""
 0 PIN_FLD_AAC_VENDOR      STR [0] ""
 0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 93423 0
 0 PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 92655 0
 0 PIN_FLD_CLOSE_WHEN_T TSTAMP [0] (0) <null>
 0 PIN_FLD_EFFECTIVE_T  TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
 0 PIN_FLD_ITEM_POID_LIST    STR [0] "0.0.0.1|/item/misc 100941 0"
 0 PIN_FLD_LASTSTAT_CMNT    STR [0] ""
 0 PIN_FLD_LAST_STATUS_T TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
 0 8052                    INT [0] 0
 0 PIN_FLD_LOGIN           STR [0] "893-20170130-173726-0-30615--152246576-den00bkf"
 0 PIN_FLD_NAME            STR [0] "PIN Service Object"
 0 PIN_FLD_NEXT_ITEM_POID_LIST    STR [0] ""
 0 PIN_FLD_OBJECT_CACHE_TYPE   ENUM [0] 0
 0 PIN_FLD_PASSWD          STR [0]
"sha1|50C2F101B640E2A5F7DB3F42C083EA52EE05516B|A6CBC7C27AD6BCE053D3BFBA7522E87E"
 0 PIN_FLD_PASSWD_EXPIRATION_T TSTAMP [0] (0) <null>
 0 PIN_FLD_PASSWD_STATUS   ENUM [0] 0
 0 PIN_FLD_PROFILE_OBJ    POID [0] 0.0.0.0  0 0
 0 PIN_FLD_SERVICE_ID      STR [0] ""
 0 8053                 TSTAMP [0] (0) <null>
 0 PIN_FLD_STATUS         ENUM [0] 10100
 0 PIN_FLD_STATUS_FLAGS    INT [0] 0
 0 PIN_FLD_SUBSCRIPTION_OBJ   POID [0] 0.0.0.0  0 0
 0 PIN_FLD_TYPE           ENUM [0] 0
 0 PIN_FLD_ALIAS_LIST    ARRAY [0] allocated 20, used 1
 1     PIN_FLD_NAME            STR [0] "0049100701"
 0 PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 1
 1     PIN_FLD_PRIMARY_NUMBER    INT [0] 0
 0 PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 2
 1     PIN_FLD_BEARER_SERVICE    STR [0] "Bearer Service"
 1     PIN_FLD_IMEI              STR [0] "IMEI"
```

Operation: send is PCM_OP_ACT_ACTIVITY (151)

```
 Flags = 0
 0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
 0 PIN_FLD_PROGRAM_NAME    STR [0] "NCC_BCD_Client"
 0 PIN_FLD_OBJ_TYPE        STR [0] "/voucher"
 0 PIN_FLD_INHERITED_INFO SUBSTRUCT [0] allocated 20, used 1
 1     10000                SUBSTRUCT [0] allocated 20, used 2
 2         10007                   STR [0] "9876543214"
 2         10008                   STR [0] "3210"
 0 PIN_FLD_BAL_IMPACTS   ARRAY [1000011] allocated 20, used 4
 1     PIN_FLD_AMOUNT       DECIMAL [0] 1
 1     PIN_FLD_RESOURCE_ID     INT [0] 1000011
 1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
 1     PIN_FLD_IMPACT_TYPE    ENUM [0] 2
```

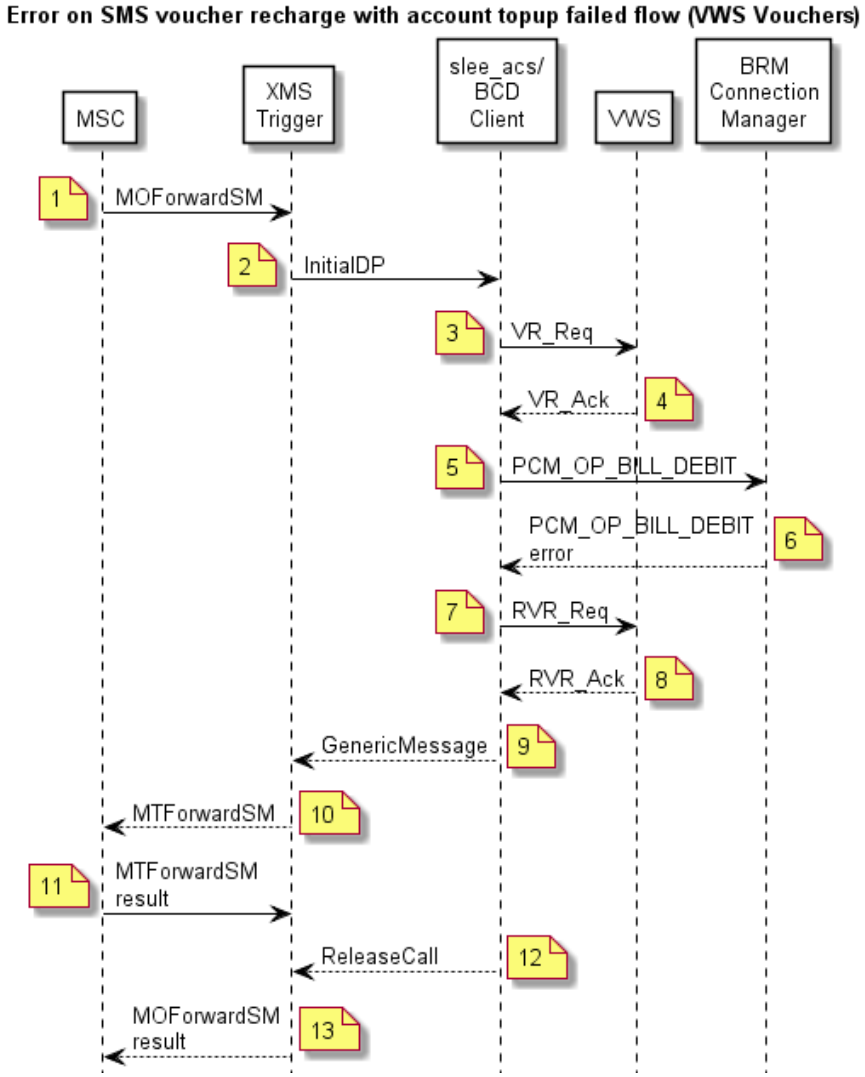
Result: received for operation PCM_OP_ACT_ACTIVITY (151)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
0 PIN_FLD_RESULTS       ARRAY [0] allocated 20, used 4
1     PIN_FLD_BAL_IMPACTS   ARRAY [1000011] allocated 20, used 8
2         PIN_FLD_AMOUNT       DECIMAL [0] 1
2         PIN_FLD_RESOURCE_ID     INT [0] 1000011
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
2         PIN_FLD_IMPACT_TYPE    ENUM [0] 2
2         PIN_FLD_GL_ID           INT [0] 0
2         PIN_FLD_ITEM_OBJ       POID [0] 0.0.0.1 /item/misc 175577 0
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 127667 6530
2         PIN_FLD_LINEAGE         STR [0] NULL str ptr
1     PIN_FLD_SUB_BAL_IMPACTS  ARRAY [1000011] allocated 20, used 3
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 127667 6531
2         PIN_FLD_RESOURCE_ID     INT [0] 1000011
2         PIN_FLD_SUB_BALANCES  ARRAY [4] allocated 20, used 3
3             PIN_FLD_AMOUNT       DECIMAL [0] 1
3             PIN_FLD_VALID_FROM   TSTAMP [0] (0) <null>
3             PIN_FLD_VALID_TO     TSTAMP [0] (0) <null>
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/activity/voucher 272643699316532185 0
```

Error on SMS recharge with account topup failed

Error on SMS recharge with account topup failed flow

Here is an example flow for an error on an SMS recharge when an account topup failed.



Error on SMS recharge with account topup failed scenario

This scenario describes the sequence of messages that occurs when the subscriber sends an SMS to recharge a voucher but the action to recharge the subscriber's wallet on BRM fails. In this scenario, the account is on BRM while the voucher is in the Convergent Charging Controller VWS.

Step	Action
1	• MSC sends MOForwardSM to xmsTrigger. • The subscriber has sent an SMS, which contains the voucher and PIN, from a GSM phone to the SMS recharge number.

Step	Action
2	• xmsTrigger sends InitialDP to the slee_acs process, with the service key set to a special value indicating SMS recharge. • The slee_acs process on the SLC receives the InitialDP and passes it to the CCS service loader. • The CCS service loader looks up the subscriber and wallet in the SCP database on the SLC and determines that the wallet information is in the BRM domain. It also determines the control plan to run, based on the serviceKey. • The slee_acs process runs the SMS recharge control plan. • The slee_acs process reaches an Extract Content feature node, which extracts the voucher number and PIN from the SMS and places them in temporary storage. • The slee_acs process reaches a Voucher Recharge feature node, which extracts the voucher number and PIN from temporary storage and uses them to invoke a VoucherRedeem action on the FOX actions library.
3	• FOX actions library sends VR_Req to BeClient. • BeClient send VR_Req to VWS.
4	• VWS sends VR_Ack to BeClient, indicating that the voucher has been reserved and returning the recharge amounts for each balance type. • BeClient send VR_Ack to the slee_acs process. • The Voucher Recharge feature node invokes a WalletRecharge action on the BCD actions library.
5	• The slee_acs process creates an event (FlistSleeEvent) for a PCM_OP_BILL_DEBIT operation and sends it to the BCD Client • The BCD Client invokes the PCM_OP_BILL_DEBIT operation and sets a timer to the configured value for this type of operation.
6	• BRM responds to the operation indicating that the account has been not been successfully recharged. • The BCD Client receives the operation output flist and sends it in an event (BcdSleeEvent) to the slee_acs process. It also marks the BRM connection as free for reuse and cancels the operation timer. • The BCD actions library translates output flist and into the error response of the WalletRecharge action. • The Voucher Recharge feature node invokes the VoucherRevoke action on the FOX actions library.
7	• FOX actions library sends RVR_Req to BeClient. • BeClient sends RVR_Req to VWS.
8	• VWS sends RVR_Ack to BeClient, indicating that the voucher reservation has been revoked. • BeClient sends RVR_Ack to the slee_acs process. • The slee_acs process takes the unsupported branch of the Voucher Recharge feature node and reaches a Send Short Message Notification feature node, which specifies sending an error text message.
9	• The SSMN feature node constructs an MMX GenericMessage containing the failure message and sends it to xmsTrigger.
10	• xmsTrigger sends an MTForwardSM operation to the MSC, containing the failure message.
11	• The MSC sends an SMS containing the failure message to the caller.
12	• The slee_acs process reaches an Accept feature node, which sends ReleaseCall to xmsTrigger.

Step	Action
13	• xmsTrigger sends a MOForwardSM result to the MSC. • The MSC sends a notification of the successful delivery of the original SMS to the caller. • The slee_acs process reaches an end node and clears the call connection.

Messages: error on SMS recharge with account topup failed

The following messages include operations sent to BRM and results returned by BRM for an error on an SMS recharge when the account topup failed. The general message format is: nesting level (0; 1, or 2); field; data type; value.

Operation: send is PCM_OP_ACT_FIND (159)

```
  Flags = 0
  0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony -1 0
  0 PIN_FLD_LOGIN           STR [0] "0049100701"
```

Result: received for operation PCM_OP_ACT_FIND (159)

```
  0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony 91119 8
  0 PIN_FLD_CREATED_T    TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
  0 PIN_FLD_MOD_T        TSTAMP [0] (1485826647) Tue Jan 31 01:37:27 2017
  0 PIN_FLD_READ_ACCESS     STR [0] "L"
  0 PIN_FLD_WRITE_ACCESS    STR [0] "L"
  0 PIN_FLD_AAC_ACCESS      STR [0] ""
  0 PIN_FLD_AAC_PACKAGE     STR [0] ""
  0 PIN_FLD_AAC_PROMO_CODE    STR [0] ""
  0 PIN_FLD_AAC_SERIAL_NUM    STR [0] ""
  0 PIN_FLD_AAC_SOURCE      STR [0] ""
  0 PIN_FLD_AAC_VENDOR      STR [0] ""
  0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 93423 0
  0 PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 92655 0
  0 PIN_FLD_CLOSE_WHEN_T TSTAMP [0] (0) <null>
  0 PIN_FLD_EFFECTIVE_T  TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
  0 PIN_FLD_ITEM_POID_LIST    STR [0] "0.0.0.1|/item/misc 100941 0"
  0 PIN_FLD_LASTSTAT_CMNT    STR [0] ""
  0 PIN_FLD_LAST_STATUS_T TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
  0 8052                     INT [0] 0
  0 PIN_FLD_LOGIN            STR [0] "893-20170130-173726-0-30615--152246576-den00bkf"
  0 PIN_FLD_NAME             STR [0] "PIN Service Object"
  0 PIN_FLD_NEXT_ITEM_POID_LIST    STR [0] ""
  0 PIN_FLD_OBJECT_CACHE_TYPE   ENUM [0] 0
  0 PIN_FLD_PASSWD           STR [0]
"sha1|50C2F101B640E2A5F7DB3F42C083EA52EE05516B|A6CBC7C27AD6BCE053D3BFBA7522E87E"
  0 PIN_FLD_PASSWD_EXPIRATION_T TSTAMP [0] (0) <null>
  0 PIN_FLD_PASSWD_STATUS   ENUM [0] 0
  0 PIN_FLD_PROFILE_OBJ    POID [0] 0.0.0.0  0 0
  0 PIN_FLD_SERVICE_ID      STR [0] ""
  0 8053                 TSTAMP [0] (0) <null>
  0 PIN_FLD_STATUS          ENUM [0] 10100
  0 PIN_FLD_STATUS_FLAGS    INT [0] 0
  0 PIN_FLD_SUBSCRIPTION_OBJ   POID [0] 0.0.0.0  0 0
  0 PIN_FLD_TYPE            ENUM [0] 0
  0 PIN_FLD_ALIAS_LIST    ARRAY [0] allocated 20, used 1
  1     PIN_FLD_NAME            STR [0] "0049100701"
  0 PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 1
  1     PIN_FLD_PRIMARY_NUMBER    INT [0] 0
  0 PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 2
  1     PIN_FLD_BEARER_SERVICE    STR [0] "Bearer Service"
  1     PIN_FLD_IMEI              STR [0] "IMEI"
```

Operation: send is PCM_OP_BILL_DEBIT (105)

```
 Flags = 0
 0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
 0 PIN_FLD_PROGRAM_NAME    STR [0] "NCC_BCD_Client"
 0 PIN_FLD_OBJ_TYPE        STR [0] "gsm/ncc"
 0 PIN_FLD_DEBIT         ARRAY [978] allocated 20, used 1
 1     PIN_FLD_BAL_OPERAND  DECIMAL [0] -0.500000000000000
```

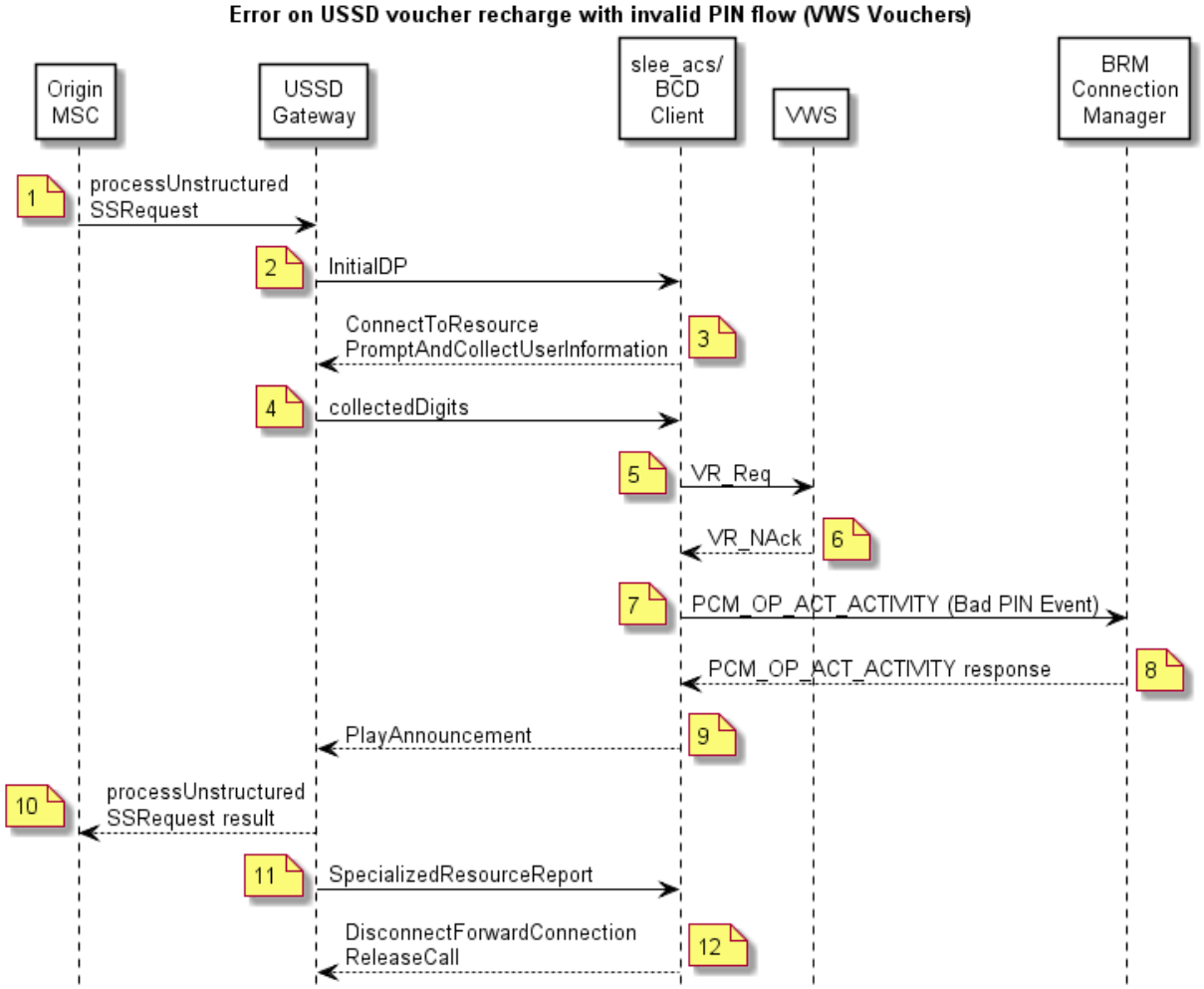
Result: received for operation PCM_OP_BILL_DEBIT(105)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /error_poid 128819 0

Error specified:
 location=5
 pin_errclass=4
 pin_err=3
 field=83893688
 rec_id=0
 reserved=0
 facility=0
 msg_id=0
 version=0
```

Error on USSD recharge with invalid PIN

Error on USSD recharge with invalid PIN flow

Here is an example flow for an error on a USSD recharge with an invalid PIN.



Error on USSD recharge with invalid Pin scenario

This scenario describes the sequence of messages that occurs when the subscriber sends a USSD message to recharge a voucher but the subscriber's pin is invalid and a USSD message is displayed on the caller's handset. In this scenario, the account is on BRM while the voucher is in the Convergent Charging Controller VWS.

Step	Action
1	• MSC sends processUnstructuredSS-Request to USSD Gateway • The calling subscriber has sent a USSD message that contains the service code for a voucher recharge, the voucher number and PIN.

Step	Action
2	• The USSD Gateway sends InitialDP to the slee_acs process, using the USSD voucher recharge service key as determined by the service code configuration. • The CCS service loader looks up the subscriber and wallet in the SCP database on the SLC and determines that the wallet information is on the BRM domain. It also determines the control plan to run, using the serviceKey to select the USSD voucher recharge control plan. • The slee_acs process runs the control plan. • The slee_acs process reaches a Voucher Recharge feature node.
3	• The Voucher Recharge feature node sends ConnectToResource, PromptAndCollectUserInformation to USSD Gateway, instructing it to prompt the caller for the voucher number and PIN. • USSD Gateway retrieves the voucher number and PIN from the received processUnstructuredSS-Request.
4	• USSD Gateway sends the PromptAndCollectUserInformation result, containing the voucher number and PIN, to the slee_acs process. • The Voucher Recharge feature node invokes the VoucherRedeem action on the FOX actions library.
5	• The FOX actions library sends VR_Req to BeClient • BeClient sends VR_Req to VWS.
6	• VWS sends VR_Nack to BeClient, indicating that the voucher PIN is incorrect. • The Voucher Recharge feature node invokes a VoucherBadPIN action on the BCD actions library. • The VoucherBadPIN action does nothing and returns success. • The Voucher Recharge feature node invokes the VoucherCreateEDR action on the BCD actions library to create a record for the bad PIN attempt.
7	• VoucherCreateEDR action constructs a FlistSleeEvent containing a PCM_OP_ACT_ACTIVITY operation for the special voucher bad PIN event and sends it to BCD Client. • The BCD Client invokes the PCM_OP_ACT_ACTIVITY operation and sets a timer to the configured value for this type of operation.
8	• BRM creates a record for the bad PIN event and responds to the operation by indicating that this has been done. • The BCD Client receives the operation output flist and sends it in an event (BcdSleeEvent) to the slee_acs process. It also marks the BRM connection as free for reuse and cancels the operation timer. • The BCD actions library translates the output flist into the response of the VoucherCreateEDR action.
9	• The Voucher Recharge feature node sends PlayAnnouncement to USSD Gateway. • USSD Gateway translates the announcement ID into text stating that the recharge was not successful, based on the configuration.
10	• USSD Gateway sends the processUnstructuredSS-Request result to the MSC. • USSD text stating that the recharge was not successful is displayed on the caller's handset.
11	• USSD Gateway sends a SpecializedResourceReport to the slee_acs process.
12	• The control plan reaches an end node and ACS sends DisconnectForwardConnection and ReleaseCall actions to the USSD Gateway and clears the call connection.

Messages: error on USSD recharge with invalid PIN

The following messages include operations sent to BRM and results returned by BRM for an error on a USSD recharge with an invalid PIN. The general message format is: nesting level (0; 1, or 2); field; data type; value.

Operation: send is PCM_OP_ACT_FIND (159)

```
 Flags = 0
 0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony -1 0
 0 PIN_FLD_LOGIN           STR [0] "0049100701"
```

Result: received for operation PCM_OP_ACT_FIND (159)

```
 0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony 91119 8
 0 PIN_FLD_CREATED_T    TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
 0 PIN_FLD_MOD_T        TSTAMP [0] (1485826647) Tue Jan 31 01:37:27 2017
 0 PIN_FLD_READ_ACCESS     STR [0] "L"
 0 PIN_FLD_WRITE_ACCESS    STR [0] "L"
 0 PIN_FLD_AAC_ACCESS      STR [0] ""
 0 PIN_FLD_AAC_PACKAGE     STR [0] ""
 0 PIN_FLD_AAC_PROMO_CODE    STR [0] ""
 0 PIN_FLD_AAC_SERIAL_NUM    STR [0] ""
 0 PIN_FLD_AAC_SOURCE      STR [0] ""
 0 PIN_FLD_AAC_VENDOR      STR [0] ""
 0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 93423 0
 0 PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 92655 0
 0 PIN_FLD_CLOSE_WHEN_T TSTAMP [0] (0) <null>
 0 PIN_FLD_EFFECTIVE_T  TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
 0 PIN_FLD_ITEM_POID_LIST    STR [0] "0.0.0.1|/item/misc 100941 0"
 0 PIN_FLD_LASTSTAT_CMNT    STR [0] ""
 0 PIN_FLD_LAST_STATUS_T TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
 0 8052                    INT [0] 0
 0 PIN_FLD_LOGIN           STR [0] "893-20170130-173726-0-30615--152246576-den00bkf"
 0 PIN_FLD_NAME            STR [0] "PIN Service Object"
 0 PIN_FLD_NEXT_ITEM_POID_LIST    STR [0] ""
 0 PIN_FLD_OBJECT_CACHE_TYPE   ENUM [0] 0
 0 PIN_FLD_PASSWD          STR [0]
"sha1|50C2F101B640E2A5F7DB3F42C083EA52EE05516B|A6CBC7C27AD6BCE053D3BFBA7522E87E"
 0 PIN_FLD_PASSWD_EXPIRATION_T TSTAMP [0] (0) <null>
 0 PIN_FLD_PASSWD_STATUS   ENUM [0] 0
 0 PIN_FLD_PROFILE_OBJ    POID [0] 0.0.0.0  0 0
 0 PIN_FLD_SERVICE_ID      STR [0] ""
 0 8053                 TSTAMP [0] (0) <null>
 0 PIN_FLD_STATUS         ENUM [0] 10100
 0 PIN_FLD_STATUS_FLAGS    INT [0] 0
 0 PIN_FLD_SUBSCRIPTION_OBJ   POID [0] 0.0.0.0  0 0
 0 PIN_FLD_TYPE           ENUM [0] 0
 0 PIN_FLD_ALIAS_LIST    ARRAY [0] allocated 20, used 1
 1     PIN_FLD_NAME            STR [0] "0049100701"
 0 PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 1
 1     PIN_FLD_PRIMARY_NUMBER    INT [0] 0
 0 PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 2
 1     PIN_FLD_BEARER_SERVICE    STR [0] "Bearer Service"
 1     PIN_FLD_IMEI              STR [0] "IMEI"
```

Operation: send is PCM_OP_ACT_ACTIVITY (151)

```
 Flags = 0
 0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
 0 PIN_FLD_PROGRAM_NAME    STR [0] "NCC_BCD_Client"
 0 PIN_FLD_OBJ_TYPE        STR [0] "/voucher"
 0 PIN_FLD_INHERITED_INFO SUBSTRUCT [0] allocated 20, used 1
 1     10000                SUBSTRUCT [0] allocated 20, used 2
 2         10007                   STR [0] "9876543214"
 2         10008                   STR [0] "3210"
 0 PIN_FLD_BAL_IMPACTS   ARRAY [1000011] allocated 20, used 4
 1     PIN_FLD_AMOUNT       DECIMAL [0] 1
 1     PIN_FLD_RESOURCE_ID     INT [0] 1000011
 1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
 1     PIN_FLD_IMPACT_TYPE    ENUM [0] 2
```

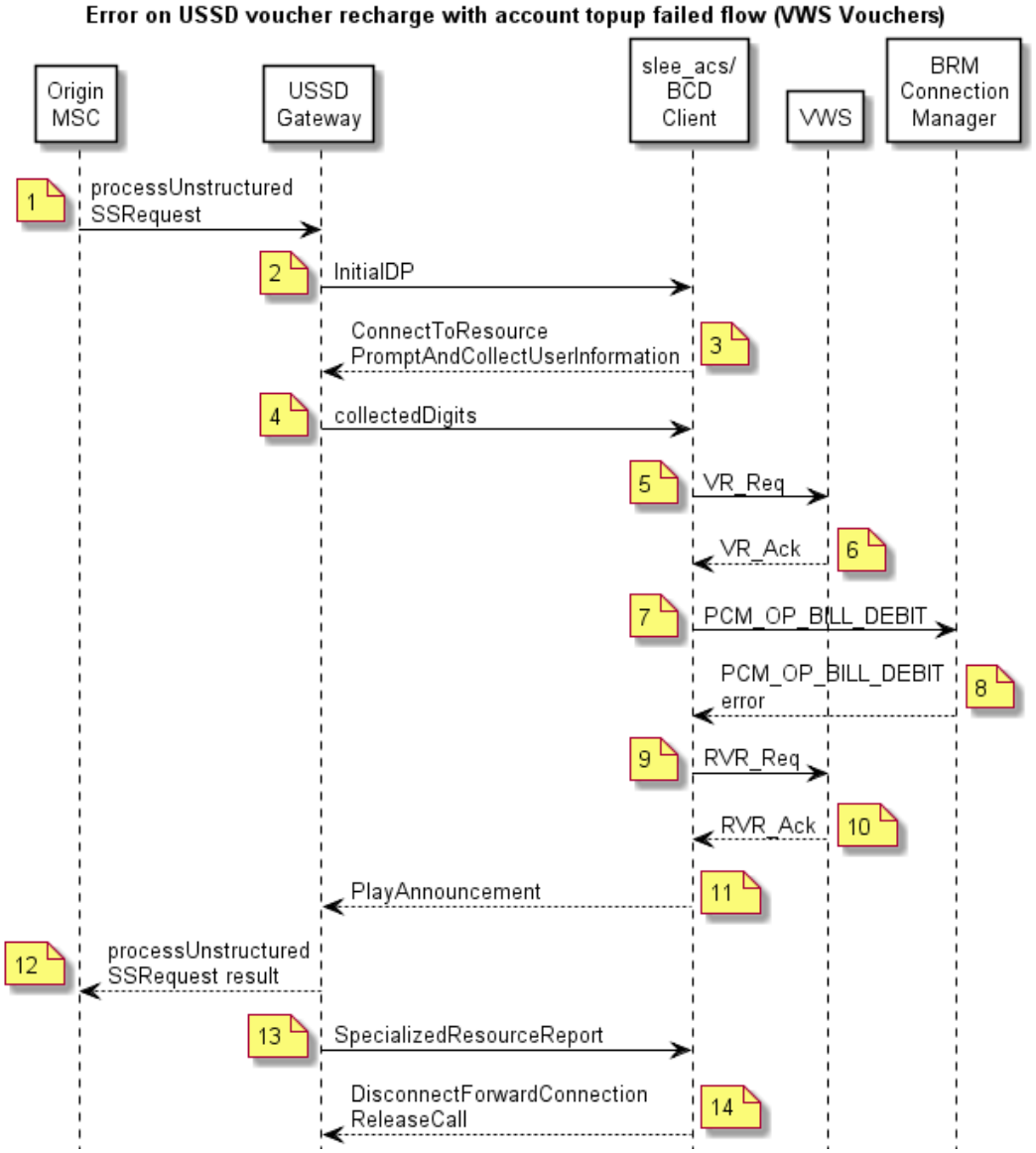
Result: received for operation PCM_OP_ACT_ACTIVITY (151)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
0 PIN_FLD_RESULTS       ARRAY [0] allocated 20, used 4
1     PIN_FLD_BAL_IMPACTS   ARRAY [1000011] allocated 20, used 8
2         PIN_FLD_AMOUNT       DECIMAL [0] 1
2         PIN_FLD_RESOURCE_ID     INT [0] 1000011
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
2         PIN_FLD_IMPACT_TYPE    ENUM [0] 2
2         PIN_FLD_GL_ID           INT [0] 0
2         PIN_FLD_ITEM_OBJ       POID [0] 0.0.0.1 /item/misc 175577 0
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 127667 6530
2         PIN_FLD_LINEAGE         STR [0] NULL str ptr
1     PIN_FLD_SUB_BAL_IMPACTS  ARRAY [1000011] allocated 20, used 3
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 127667 6531
2         PIN_FLD_RESOURCE_ID     INT [0] 1000011
2         PIN_FLD_SUB_BALANCES  ARRAY [4] allocated 20, used 3
3             PIN_FLD_AMOUNT       DECIMAL [0] 1
3             PIN_FLD_VALID_FROM   TSTAMP [0] (0) <null>
3             PIN_FLD_VALID_TO     TSTAMP [0] (0) <null>
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/activity/voucher 272643699316532185 0
```

Error on USSD recharge with account topup failed

Error on USSD recharge with account topup failed flow

Here is an example message flow for an error on a USSD Recharge with an account topup that failed.



Error on USSD recharge with account topup failed scenario

This scenario describes the sequence of messages that occurs when the subscriber sends a USSD message requesting a voucher recharge but the wallet recharge action fails on BRM and a USSD text message is displayed on the caller's display. In this scenario, the account in on BRM while the voucher is in the Convergent Charging Controller VWS.

Step	Action
1	• MSC sends processUnstructuredSS-Request to USSD Gateway • The calling subscriber has sent a USSD message that contains the service code for a voucher recharge, the voucher number and PIN.

Step	Action
2	• The USSD Gateway sends InitialDP to the slee_acs process, using the USSD voucher recharge service key as determined by the service code configuration. • The CCS service loader looks up the subscriber and wallet in the SCP database on the SLC and determines that the wallet information is on the BRM domain. It also determines the control plan to run, using the serviceKey to select the USSD voucher recharge control plan. • The slee_acs process runs the control plan. • The slee_acs process reaches a Voucher Recharge feature node.
3	• The Voucher Recharge feature node sends ConnectToResource, PromptAndCollectUserInformation to USSD Gateway, instructing it to prompt the caller for the voucher number and PIN. • USSD Gateway retrieves the voucher number and PIN from the received processUnstructuredSS-Request.
4	• USSD Gateway sends the PromptAndCollectUserInformation result, containing the voucher number and PIN, to the slee_acs process. • The Voucher Recharge feature node invokes the VoucherRedeem action on the FOX actions library.
5	• The FOX actions library sends VR_Req to BeClient. • BeClient sends VR_Req to VWS.
6	• VWS sends VR_Ack to BeClient, indicating that the voucher has been reserved, and returning the recharge amounts for each balance type. • BeClient sends VR_Ack to the slee_acs process. • The Voucher Recharge feature node invokes the WalletRecharge action on the BCD actions library.
7	• The slee_acs process creates a FlistSleeEvent for a PCM_OP_BILL_DEBIT operation and sends it to the BCD Client. • The BCD actions library creates an event (FlistSleeEvent) that contains a PCM_OP_BILL_DEBIT operation and sends it to the Client. • The BCD Client invokes the PCM_OP_BILL_DEBIT operation and sets a timer to the configured value for this type of operation.
8	• BRM responds to the operation by indicating that the account has not been successfully recharged. • The BCD Client receives the operation output flist and sends it in an event (BcdSleeEvent) to the slee_acs process. It also marks the BRM connection as free for reuse and cancels the operation timer. • The BCD actions library translates the output flist into the response of the WalletRecharge action. • The Voucher Recharge feature node invokes the VoucherRevoke action on the FOX actions library.
9	• FOX actions library sends RVR_Req to BeClient. • BeClient sends RVR_Req to VWS.
10	• VWS sends RVR_Ack to BeClient, indicating that the voucher reservation has been revoked. • BeClient sends RVR_Ack to the slee_acs process.
11	• Voucher Recharge node sends PlayAnnouncement to USSD Gateway. • USSD Gateway translates the announcement ID into some text that states that the recharge was not successful, based on the configuration.

Step	Action
12	• USSD Gateway sends processUnstructuredSS-Request result to the MSC. • USSD text stating that the recharge was not successful is displayed on the caller's handset.
13	• USSD Gateway sends SpecializedResourceReport to the slee_acs process.
14	• The control plan reaches an end node and ACS sends DisconnectForwardConnection and ReleaseCall actions to the USSD Gateway and clears the call connection.

Messages: error on USSD recharge with account topup failed

The following messages include operations sent to BRM and results returned by BRM for an error on a USSD recharge with a failed account topup. The general message format is: nesting level (0; 1, or 2); field; data type; value.

Operation: send is PCM_OP_ACT_FIND (159)

```
 Flags = 0
 0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony -1 0
 0 PIN_FLD_LOGIN           STR [0] "0049100701"
```

Result: received for operation PCM_OP_ACT_FIND (159)

```
 0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony 91119 8
 0 PIN_FLD_CREATED_T    TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
 0 PIN_FLD_MOD_T        TSTAMP [0] (1485826647) Tue Jan 31 01:37:27 2017
 0 PIN_FLD_READ_ACCESS     STR [0] "L"
 0 PIN_FLD_WRITE_ACCESS    STR [0] "L"
 0 PIN_FLD_AAC_ACCESS      STR [0] ""
 0 PIN_FLD_AAC_PACKAGE     STR [0] ""
 0 PIN_FLD_AAC_PROMO_CODE    STR [0] ""
 0 PIN_FLD_AAC_SERIAL_NUM    STR [0] ""
 0 PIN_FLD_AAC_SOURCE      STR [0] ""
 0 PIN_FLD_AAC_VENDOR      STR [0] ""
 0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 93423 0
 0 PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 92655 0
 0 PIN_FLD_CLOSE_WHEN_T TSTAMP [0] (0) <null>
 0 PIN_FLD_EFFECTIVE_T  TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
 0 PIN_FLD_ITEM_POID_LIST    STR [0] "0.0.0.1|/item/misc 100941 0"
 0 PIN_FLD_LASTSTAT_CMNT    STR [0] ""
 0 PIN_FLD_LAST_STATUS_T TSTAMP [0] (1485826646) Tue Jan 31 01:37:26 2017
 0 8052                     INT [0] 0
 0 PIN_FLD_LOGIN           STR [0] "893-20170130-173726-0-30615--152246576-den00bkf"
 0 PIN_FLD_NAME            STR [0] "PIN Service Object"
 0 PIN_FLD_NEXT_ITEM_POID_LIST    STR [0] ""
 0 PIN_FLD_OBJECT_CACHE_TYPE   ENUM [0] 0
 0 PIN_FLD_PASSWD          STR [0]
"sha1|50C2F101B640E2A5F7DB3F42C083EA52EE05516B|A6CBC7C27AD6BCE053D3BFBA7522E87E"
 0 PIN_FLD_PASSWD_EXPIRATION_T TSTAMP [0] (0) <null>
 0 PIN_FLD_PASSWD_STATUS   ENUM [0] 0
 0 PIN_FLD_PROFILE_OBJ    POID [0] 0.0.0.0  0 0
 0 PIN_FLD_SERVICE_ID      STR [0] ""
 0 8053                 TSTAMP [0] (0) <null>
 0 PIN_FLD_STATUS         ENUM [0] 10100
 0 PIN_FLD_STATUS_FLAGS    INT [0] 0
 0 PIN_FLD_SUBSCRIPTION_OBJ   POID [0] 0.0.0.0  0 0
 0 PIN_FLD_TYPE           ENUM [0] 0
 0 PIN_FLD_ALIAS_LIST    ARRAY [0] allocated 20, used 1
 1     PIN_FLD_NAME            STR [0] "0049100701"
 0 PIN_FLD_TELCO_INFO   SUBSTRUCT [0] allocated 20, used 1
 1     PIN_FLD_PRIMARY_NUMBER    INT [0] 0
 0 PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 2
 1     PIN_FLD_BEARER_SERVICE    STR [0] "Bearer Service"
 1     PIN_FLD_IMEI              STR [0] "IMEI"
```

Operation: send is PCM_OP_BILL_DEBIT (105)

```
 Flags = 0
 0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
 0 PIN_FLD_PROGRAM_NAME    STR [0] "NCC_BCD_Client"
 0 PIN_FLD_OBJ_TYPE        STR [0] "gsm/ncc"
 0 PIN_FLD_DEBIT         ARRAY [978] allocated 20, used 1
 1     PIN_FLD_BAL_OPERAND  DECIMAL [0] -0.500000000000000
```

Result: received for operation PCM_OP_BILL_DEBIT(105)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /error_poid 128819 0

Error specified:
 location=5
 pin_errclass=4
 pin_err=3
 field=83893688
 rec_id=0
 reserved=0
 facility=0
 msg_id=0
 version=0
```

Messages: top up with invalid voucher number or PIN

The following messages include operations sent to BRM and results returned by BRM for a top up request using a bad voucher number or PIN. The general message format is: nesting level (0; 1, or 2); field; data type; value.

Result: Received for operation PCM_OP_PYMT_TOPUP (3726)

```
Error specified:
 location=5
 pin_errclass=4
 pin_err=3
 field=83893688
 rec_id=0
 reserved=0
 facility=0
 msg_id=0
 version=0
```

The error code of 3 is defined like this:

```
#define PIN_ERR_NOT_FOUND                   3
```

Operation: Send PCM_OP_READ_FLDS (4)

```
Flags = 0
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 135267 0
0 PIN_FLD_CURRENCY        INT [0] 0
0 PIN_FLD_STATUS         ENUM [0] 0
0 PIN_FLD_CLOSE_WHEN_T TSTAMP [0] (0) <null>
0 PIN_FLD_MOD_T        TSTAMP [0] (0) <null>
0 PIN_FLD_LAST_STATUS_T TSTAMP [0] (0) <null>
```

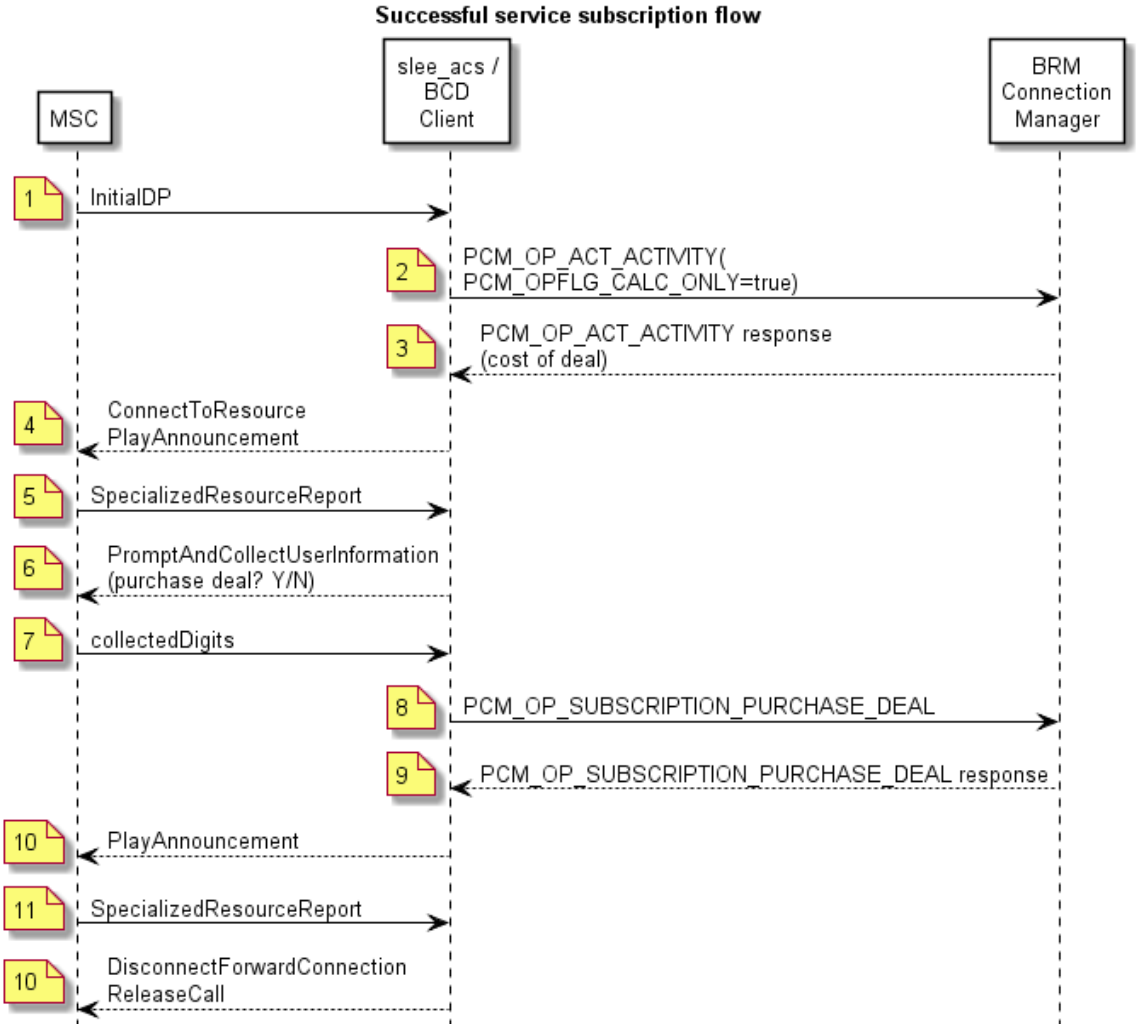
Result: Received for operation PCM_OP_READ_FLDS (4)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 135267 8
0 PIN_FLD_CURRENCY        INT [0] 978
0 PIN_FLD_STATUS         ENUM [0] 10100
0 PIN_FLD_CLOSE_WHEN_T TSTAMP [0] (0) <null>
0 PIN_FLD_MOD_T        TSTAMP [0] (1331700907) Wed Mar 14 04:55:07 2012
0 PIN_FLD_LAST_STATUS_T TSTAMP [0] (1331675046) Tue Mar 13 21:44:06 2012
```

Successful service subscription

Successful service subscription flow

Here is an example message flow for a successful service subscription flow.



Successful service subscription scenario

This scenario describes the sequence of messages that occurs when the caller calls the toll free number to subscribe to a service.

Step	Action
1	• MSC sends InitialDP to the slee_acs process, with serviceKey set to a special value indicating “subscribe to service”. • The subscriber has dialed the “subscribe to service” toll free number. • The CCS service loader looks up the subscriber and wallet in the SCP database on the SLC and determines that the wallet information is on the BRM domain. It also determines the control plan to run, using the serviceKey to select the “subscribe to service” control plan. • The slee_acs process runs the control plan. • The slee_acs process reaches a Named Event feature node, specifying named event rates for a non-reservable event.

Step	Action
2	• The BCD actions library constructs a FlistSleeEvent containing a PCM_OP_ACT_ACTIVITY operation and sends it to BCD Client. • The BCD actions library constructs an event(FlistSleeEvent) that contains the PCM_OP_ACT_ACTIVITY operation and sends it to the BCD Client. • The BCD Client invokes the PCM_OP_ACT_ACTIVITY operation and sets a timer to the configured value for this type of operation.
3	• BRM responds to the operation, giving the cost of the deal. • The BCD Client receives the operation output flist and sends it in an event (BcdSleeEvent) to the slee_acs process. It also marks the BRM connection as free for reuse and cancels the operation timer. • The BCD actions library translates the output flist into the response of the NamedEventRates action. • The Named Event feature node stores the cost in call context, takes the success exit and reaches a Play Variable Part Announcement node.
4	• The slee_acs process sends ConnectToResource,PlayAnnouncement to the MSC, with a variable part indicating the cost of the deal. • MSC plays the announcement to the caller.
5	• The announcement finishes and MSC sends a SpecializedResourceReport message to the slee_acs process. • The Play Variable Part Announcement feature node takes the success exit. • The slee_acs process reaches a Selection Dependent Routing feature node.
6	• The Selection Dependent Routing node sends PromptAndCollectUserInformation to the MSC, instructing the MSC to prompt for a single digit from a menu. • MSC plays the specified announcement to the caller and collects the menu choice.
7	• MSC sends PromptAndCollectUserInformation result, containing the menu choice digit, to the slee_acs process. • The SDR feature node takes an exit to a Named Event feature node that specifies a direct named event for the same type of non-reservable event.
8	• The BCD actions library constructs an event (FlistSleeEvent) that contains a PCM_OP_SUBSCRIPTION_PURCHASE_DEAL operation and sends it to BCD Client. Sometimes the BCD actions library will send PCM_OP_CUST_POL_GET_DEALS before sending PCM_OP_SUBSCRIPTION_PURCHASE_DEAL if it has no knowledge of the requested deal or if the cached information about the deal is too old. • The BCD Client invokes the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL operation and sets a timer to the configured value for this type of operation.
9	• BRM responds to the operation indicating that the deal has been purchased. • The BCD Client receives the operation output flist and sends it in an event (BcdSleeEvent) to the slee_acs process. It also marks the BRM connection as free for reuse and cancels the operation timer. • The BCD actions library translates the output flist into the response of the DirectNamedEvent action. • The Named Event feature node takes the success exit and reaches a Play Announcement feature node for a success announcement.
10	• A Play Announcement feature node sends PlayAnnouncement to the MSC. • MSC plays an announcement to the caller, stating that the transaction was successful.

Step	Action
11	• The announcement finishes and MSC sends a SpecializedResourceReport message to the slee_acs process.
12	• The control plan reaches an end node and ACS sends DisconnectForwardConnection and ReleaseCall actions to MSC and clears the call context. • The caller is disconnected.

Messages: successful service subscription

The following messages include operations sent to BRM and results returned by BRM for a successful service subscription. The general message format is: nesting level (0; 1, or 2); field; data type; value.

Operation: send is PCM_OP_BAL_GET_ECE_BALANCE (1281)

```
Flags = 0
0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony -1 0
0 PIN_FLD_LOGIN           STR [0] "0049100701"
0 PIN_FLD_MODE           ENUM [0] 1
0 PIN_FLD_START_T      TSTAMP [0] (1486008379) Thu Feb 02 04:06:19 2017
```

Result: received for operation PCM_OP_BAL_GET_ECE_BALANCE (1281)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony -1 0
0 PIN_FLD_MODE           ENUM [0] 1
0 PIN_FLD_BAL_IMPACTS   ARRAY [840] allocated 20, used 6
1     PIN_FLD_CREDIT_LIMIT DECIMAL [0] 2000
1     PIN_FLD_CURRENT_BAL  DECIMAL [0] 10.50
1     PIN_FLD_CREDIT_PROFILE    INT [0] 6
1     PIN_FLD_CREDIT_FLOOR DECIMAL [0] -300000
1     PIN_FLD_SUB_BAL_IMPACTS  ARRAY [840] allocated 20, used 4
2         3205                 DECIMAL [0] 0
2         PIN_FLD_AMOUNT       DECIMAL [0] -3.00
2         PIN_FLD_VALID_TO     TSTAMP [0] (1503184917) Sat Aug 19 23:21:57 2017
2         PIN_FLD_VALID_FROM   TSTAMP [0] (0) <null>
1     PIN_FLD_SUB_BAL_IMPACTS  ARRAY [1] allocated 20, used 3
2         3205                 DECIMAL [0] 0
2         PIN_FLD_AMOUNT       DECIMAL [0] 13.50
2         PIN_FLD_VALID_FROM   TSTAMP [0] (1485826686) Tue Jan 31 01:38:06 2017
```

Operation: send is PCM_OP_ACT_ACTIVITY (151)

```
Flags = 128
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
0 PIN_FLD_PROGRAM_NAME    STR [0] "NCC_BCD_Client"
0 PIN_FLD_OBJ_TYPE        STR [0] "/gsm/ncc"
```

Result: received for operation PCM_OP_ACT_ACTIVITY (151)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
0 PIN_FLD_RESULTS       ARRAY [0] allocated 27, used 27
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/activity/gsm/ncc -1 0
1     PIN_FLD_NAME            STR [0] "Activity Tracking Event Log"
1     PIN_FLD_USERID         POID [0] 0.0.0.1 /service/pcm_client 1 34764
1     PIN_FLD_SESSION_OBJ    POID [0] 0.0.0.1 /event/session 272643699316532386 0
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
1     PIN_FLD_PROGRAM_NAME    STR [0] "NCC_BCD_Client"
1     PIN_FLD_START_T      TSTAMP [0] (1339108525) Thu Jun 07 22:35:25 2012
1     PIN_FLD_END_T        TSTAMP [0] (1339108525) Thu Jun 07 22:35:25 2012
1     PIN_FLD_FLAGS           INT [0] 128
1     PIN_FLD_SYS_DESCR       STR [0] "Activity: /gsm/ncc"
1     PIN_FLD_BILLINFO_OBJ   POID [0] 0.0.0.1 /billinfo 129203 0
1     PIN_FLD_RUM_NAME        STR [0] "Occurrence"
1     PIN_FLD_UNIT           ENUM [0] 0
1     PIN_FLD_TOD_MODE       ENUM [0] 0
1     PIN_FLD_NET_QUANTITY DECIMAL [0] 1
1     PIN_FLD_MIN_QUANTITY DECIMAL [0] 1
1     PIN_FLD_INCR_QUANTITY DECIMAL [0] 1
```

```
1     PIN_FLD_MIN_UNIT       ENUM [0] 0
1     PIN_FLD_INCR_UNIT      ENUM [0] 0
1     PIN_FLD_ROUNDING_MODE   ENUM [0] 0
1     PIN_FLD_TIMEZONE_MODE   ENUM [0] 1
1     PIN_FLD_RATED_TIMEZONE_ID    STR [0] "US/Pacific"
1     PIN_FLD_BAL_IMPACTS   ARRAY [0] allocated 20, used 18
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 338
2         PIN_FLD_PRODUCT_OBJ    POID [0] 0.0.0.1 /product 80619 9
2         PIN_FLD_TAX_CODE        STR [0] ""
2         PIN_FLD_RATE_OBJ       POID [0] 0.0.0.1 /rate 79259 1
2         PIN_FLD_RATE_TAG        STR [0] "Rate 1"
2         PIN_FLD_IMPACT_CATEGORY    STR [0] "default"
2         PIN_FLD_OFFERING_OBJ   POID [0] 0.0.0.1 /purchased_product 129779 176
2         PIN_FLD_LINEAGE         STR [0] NULL str ptr
2         PIN_FLD_GL_ID           INT [0] 0
2         PIN_FLD_QUANTITY     DECIMAL [0] 1.00000000
2         PIN_FLD_IMPACT_TYPE    ENUM [0] 1
2         PIN_FLD_DISCOUNT     DECIMAL [0] 0
2         PIN_FLD_PERCENT      DECIMAL [0] 1
2         PIN_FLD_AMOUNT       DECIMAL [0] 0.200
2         PIN_FLD_RESOURCE_ID     INT [0] 978
2         PIN_FLD_AMOUNT_DEFERRED DECIMAL [0] 0
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 127667 6798
2         PIN_FLD_BILLINFO_OBJ   POID [0] 0.0.0.1 /billinfo 129203 0
1     PIN_FLD_TOTAL         ARRAY [978] allocated 20, used 1
2         PIN_FLD_AMOUNT       DECIMAL [0] 0.200
1     PIN_FLD_UNRATED_QUANTITY DECIMAL [0] 0
1     PIN_FLD_RATING_STATUS   ENUM [0] 0
1     PIN_FLD_SUB_BAL_IMPACTS  ARRAY [0] allocated 20, used 3
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 127667 6798
2         PIN_FLD_RESOURCE_ID     INT [0] 978
2         PIN_FLD_SUB_BALANCES  ARRAY [0] allocated 20, used 3
3             PIN_FLD_AMOUNT       DECIMAL [0] 0.200
3             PIN_FLD_VALID_FROM   TSTAMP [0] (0) <null>
3             PIN_FLD_VALID_TO     TSTAMP [0] (0) <null>
```

Operation: send is PCM_OP_CUST_POL_GET_DEALS (278)

```
Flags = 0
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
```

Result: received for operation PCM_OP_CUST_POL_GET_DEALS (278)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
0 PIN_FLD_DEALS         ARRAY [10] allocated 20, used 14
1     PIN_FLD_POID           POID [0] 0.0.0.1 /deal 90056 4
1     PIN_FLD_CREATED_T    TSTAMP [0] (1327455797) Wed Jan 25 01:43:17 2012
1     PIN_FLD_MOD_T        TSTAMP [0] (1327455952) Wed Jan 25 01:45:52 2012
1     PIN_FLD_READ_ACCESS     STR [0] "B"
1     PIN_FLD_WRITE_ACCESS    STR [0] "S"
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 0
1     PIN_FLD_CODE            STR [0] "50 Euro Topup Deal"
1     PIN_FLD_DESCR           STR [0] ""
1     PIN_FLD_END_T        TSTAMP [0] (0) <null>
1     PIN_FLD_FLAGS           INT [0] 0
1     PIN_FLD_NAME            STR [0] "50 Euro Topup Deal"
1     PIN_FLD_PERMITTED       STR [0] "/account"
1     PIN_FLD_START_T      TSTAMP [0] (0) <null>
1     PIN_FLD_PRODUCTS      ARRAY [0] allocated 20, used 19
2         PIN_FLD_CYCLE_DISCOUNT DECIMAL [0] 0
2         PIN_FLD_CYCLE_END_DETAILS    INT [0] 2
2         PIN_FLD_CYCLE_END_T  TSTAMP [0] (0) <null>
2         PIN_FLD_CYCLE_START_DETAILS    INT [0] 1
2         PIN_FLD_CYCLE_START_T TSTAMP [0] (0) <null>
2         PIN_FLD_PRODUCT_OBJ    POID [0] 0.0.0.1 /product 82729 6
2         PIN_FLD_PURCHASE_DISCOUNT DECIMAL [0] 0
2         PIN_FLD_PURCHASE_END_DETAILS    INT [0] 2
2         PIN_FLD_PURCHASE_END_T TSTAMP [0] (0) <null>
2         PIN_FLD_PURCHASE_START_DETAILS    INT [0] 1
2         PIN_FLD_PURCHASE_START_T TSTAMP [0] (0) <null>
2         PIN_FLD_QUANTITY     DECIMAL [0] 1
```

```
2         PIN_FLD_STATUS         ENUM [0] 1
2         PIN_FLD_STATUS_FLAGS    INT [0] 0
2         PIN_FLD_USAGE_DISCOUNT DECIMAL [0] 0
2         PIN_FLD_USAGE_END_DETAILS    INT [0] 2
2         PIN_FLD_USAGE_END_T  TSTAMP [0] (0) <null>
2         PIN_FLD_USAGE_START_DETAILS    INT [0] 1
2         PIN_FLD_USAGE_START_T TSTAMP [0] (0) <null>
0 PIN_FLD_DEALS         ARRAY [9] allocated 20, used 14
1     PIN_FLD_POID           POID [0] 0.0.0.1 /deal 88008 4
1     PIN_FLD_CREATED_T    TSTAMP [0] (1327455797) Wed Jan 25 01:43:17 2012
1     PIN_FLD_MOD_T        TSTAMP [0] (1327455952) Wed Jan 25 01:45:52 2012
1     PIN_FLD_READ_ACCESS     STR [0] "B"
1     PIN_FLD_WRITE_ACCESS    STR [0] "S"
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 1 0
1     PIN_FLD_CODE            STR [0] "DirectEventDeal"
1     PIN_FLD_DESCR           STR [0] ""
1     PIN_FLD_END_T        TSTAMP [0] (0) <null>
1     PIN_FLD_FLAGS           INT [0] 0
1     PIN_FLD_NAME            STR [0] "DirectEventDeal"
1     PIN_FLD_PERMITTED       STR [0] "/account"
1     PIN_FLD_START_T      TSTAMP [0] (0) <null>
1     PIN_FLD_PRODUCTS      ARRAY [0] allocated 20, used 19
2         PIN_FLD_CYCLE_DISCOUNT DECIMAL [0] 0
2         PIN_FLD_CYCLE_END_DETAILS    INT [0] 2
2         PIN_FLD_CYCLE_END_T  TSTAMP [0] (0) <null>
2         PIN_FLD_CYCLE_START_DETAILS    INT [0] 1
2         PIN_FLD_CYCLE_START_T TSTAMP [0] (0) <null>
2         PIN_FLD_PRODUCT_OBJ    POID [0] 0.0.0.1 /product 84328 7
2         PIN_FLD_PURCHASE_DISCOUNT DECIMAL [0] 0
2         PIN_FLD_PURCHASE_END_DETAILS    INT [0] 2
2         PIN_FLD_PURCHASE_END_T TSTAMP [0] (0) <null>
2         PIN_FLD_PURCHASE_START_DETAILS    INT [0] 1
2         PIN_FLD_PURCHASE_START_T TSTAMP [0] (0) <null>
2         PIN_FLD_QUANTITY     DECIMAL [0] 1
2         PIN_FLD_STATUS         ENUM [0] 1
2         PIN_FLD_STATUS_FLAGS    INT [0] 0
2         PIN_FLD_USAGE_DISCOUNT DECIMAL [0] 0
2         PIN_FLD_USAGE_END_DETAILS    INT [0] 2
2         PIN_FLD_USAGE_END_T  TSTAMP [0] (0) <null>
2         PIN_FLD_USAGE_START_DETAILS    INT [0] 1
2         PIN_FLD_USAGE_START_T TSTAMP [0] (0) <null>
```

Operation: send is PCM_OP_SUBSCRIPTION_PURCHASE_DEAL (108)

```
Flags = 0
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
0 PIN_FLD_PROGRAM_NAME    STR [0] "NCC_BCD_Client"
0 PIN_FLD_DEAL_INFO    SUBSTRUCT [0] allocated 20, used 1
1     PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.1 /deal 88008 4
```

Result: received for operation PCM_OP_SUBSCRIPTION_PURCHASE_DEAL (108)

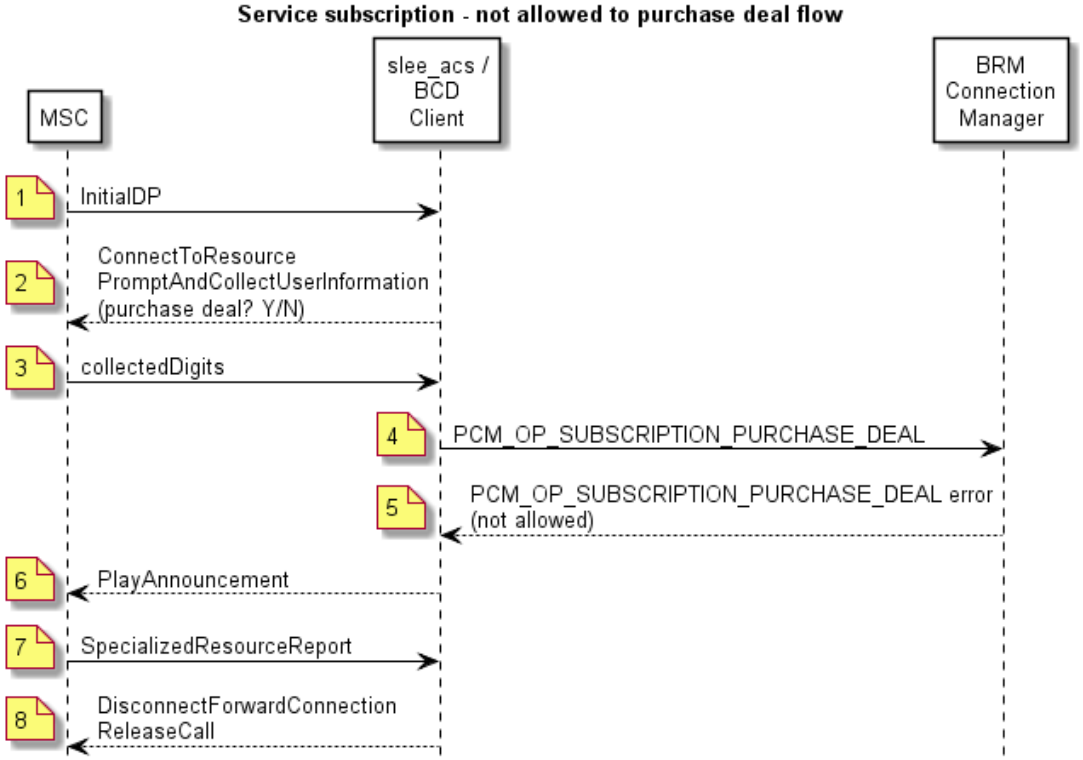
```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
0 PIN_FLD_RESULTS       ARRAY [0] allocated 20, used 9
1     PIN_FLD_BAL_IMPACTS   ARRAY [0] allocated 20, used 18
2         PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 338
2         PIN_FLD_PRODUCT_OBJ    POID [0] 0.0.0.1 /product 84328 7
2         PIN_FLD_TAX_CODE        STR [0] ""
2         PIN_FLD_RATE_OBJ       POID [0] 0.0.0.1 /rate 79451 1
2         PIN_FLD_RATE_TAG        STR [0] "Rate 1"
2         PIN_FLD_IMPACT_CATEGORY    STR [0] "default"
2         PIN_FLD_OFFERING_OBJ   POID [0] NULL poid pointer
2         PIN_FLD_LINEAGE         STR [0] NULL str ptr
2         PIN_FLD_GL_ID           INT [0] 0
2         PIN_FLD_QUANTITY     DECIMAL [0] 1.00000000
2         PIN_FLD_IMPACT_TYPE    ENUM [0] 1
2         PIN_FLD_DISCOUNT     DECIMAL [0] 0
2         PIN_FLD_PERCENT      DECIMAL [0] 1
2         PIN_FLD_AMOUNT       DECIMAL [0] 0.300
2         PIN_FLD_RESOURCE_ID     INT [0] 978
2         PIN_FLD_AMOUNT_DEFERRED DECIMAL [0] 0
2         PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 127667 6798
```

```
2          PIN_FLD_ITEM_OBJ       POID [0] 0.0.0.1 /item/misc 129715 0
1     PIN_FLD_SUB_BAL_IMPACTS  ARRAY [0] allocated 20, used 3
2          PIN_FLD_BAL_GRP_OBJ    POID [0] 0.0.0.1 /balance_group 127667 6799
2          PIN_FLD_RESOURCE_ID     INT [0] 978
2          PIN_FLD_SUB_BALANCES  ARRAY [0] allocated 20, used 3
3              PIN_FLD_AMOUNT       DECIMAL [0] 0.300
3              PIN_FLD_VALID_FROM   TSTAMP [0] (0) <null>
3              PIN_FLD_VALID_TO     TSTAMP [0] (0) <null>
1     PIN_FLD_UNRATED_QUANTITY DECIMAL [0] 0
1     PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.0  0 0
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
1     PIN_FLD_RATING_STATUS   ENUM [0] 0
1     PIN_FLD_NET_QUANTITY DECIMAL [0] 1
1     PIN_FLD_RUM_NAME        STR [0] "Occurrence"
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/billing/product/fee/purchase
272643699316538418 0
0 PIN_FLD_RESULTS       ARRAY [1] allocated 20, used 3
1     PIN_FLD_SERVICE_OBJ    POID [0] 0.0.0.0  0 0
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/billing/product/action/purchase
17592186221106 0
0 PIN_FLD_PRODUCTS      ARRAY [0] allocated 20, used 2
1     PIN_FLD_PRODUCT_OBJ    POID [0] 0.0.0.1 /product 84328 7
1     PIN_FLD_PACKAGE_ID      INT [0] 260
0 PIN_FLD_RESULTS       ARRAY [2] allocated 20, used 2
1     PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
1     PIN_FLD_POID           POID [0] 0.0.0.1 /event/billing/deal/purchase 17592186223154 0
```

Service subscription - not allowed to purchase deal

Service subscription - not allowed to purchase deal flow

Here is an example message flow for the subscriber not being allowed to purchase a deal.



Service subscription - not allowed to purchase deal scenario

This scenario describes the sequence of messages that occurs when the subscriber is not allowed to purchase a deal.

Step	Action
1	• MSC sends InitialDP to the slee_acs process, with serviceKey set to a special value indicating “subscribe to service”. • The subscriber has dialed the “subscribe to service” toll free number. • The CCS service loader looks up the subscriber and wallet in the SCP database on the SLC and determines that the wallet information is on the BRM domain. It also determines the control plan to run, using the serviceKey to select the “subscribe to service” control plan. • The slee_acs process runs the control plan. • The slee_acs process reaches a Selection Dependent Routing feature node.
2	• The Selection Dependent Routing feature node sends ConnectToResource, PromptAndCollectUserInformation to the MSC, instructing the MSC to prompt for a single digit from a menu. • MSC plays the specified announcement to the caller and collects the menu choice.
3	• MSC sends PromptAndCollectUserInformation result, containing the menu choice digit, to the slee_acs process. • The SDR feature node takes an exit to a Named Event feature node and specifies a direct named event for a non-reservable event.
4	• The BCD actions library constructs an event (FlistSleeEvent) that contains a PCM_OP_SUBSCRIPTION_PURCHASE_DEAL operation and sends it to the BCD Client. • The BCD Client invokes the PCM_OP_SUBSCRIPTION_PURCHASE_DEAL operation and sets a timer to the configured value for this type of operation.
5	• BRM responds to the operation with an error. • The BCD Client receives the operation error output flist and sends it in an event (BcdSleeEvent) to the slee_acs process. It also marks the BRM connection as free for reuse and cancels the operation timer. • The BCD actions library translates the error output flist into the error response of the DirectNamedEvent action. • The Named Event feature node takes an error exit and reaches a Play Announcement node for an error announcement.
6	• A Play Announcement feature node sends PlayAnnouncement to the MSC. • MSC plays an announcement to the caller, stating that the transaction was not successful.
7	• The announcement finishes and MSC sends SpecializedResourceReport to the slee_acs process.
8	• The control plan reaches an end node and ACS sends DisconnectForwardConnection and ReleaseCall actions to MSC and clears the call context. • The caller is disconnected.

Messages: service subscription - not allowed to purchase deal

The following messages include operations sent to BRM and results returned by BRM for service subscription when the subscriber is not allowed to purchase a deal. The general message format is: nesting level (0; 1, or 2); field; data type; value.

Operation: send is PCM_OP_ACT_FIND (159)

```
Flags = 0
 0 PIN_FLD_POID           POID [0] 0.0.0.1 /service/telco/gsm/telephony -1 0
 0 PIN_FLD_PROGRAM_NAME    STR [0] "NCC_BCD_Client"
 0 PIN_FLD_OBJ_TYPE        STR [0] "gsm/ncc"
 0 PIN_FLD_EXTENDED_INFO SUBSTRUCT [0] allocated 20, used 2
 1     PIN_FLD_GSM_INFO     SUBSTRUCT [0] allocated 20, used 3
 2         PIN_FLD_DIRECTION      ENUM [0] 0
 2         PIN_FLD_CELL_ID         STR [0] "000c"
 2         PIN_FLD_LOC_AREA_CODE    STR [0] "064001000f"
 1     10000                SUBSTRUCT [0] allocated 20, used 3
 2         10001                    STR [0] "Present"
 2         PIN_FLD_LOCATION         STR [0] "004085752160"
 2         10002                    INT [0] 1
 0 PIN_FLD_MSID            STR [0] "004085752160"
 0 PIN_FLD_FLAGS           INT [0] 4
```

Result: received for operation PCM_OP_ACT_FIND (159)

```
 0 PIN_FLD_POID           POID [0] 0.0.0.1 /balance_group 127667 6801
 0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 128819 0
 0 PIN_FLD_BILLINFO_OBJ   POID [0] 0.0.0.1 /billinfo 129203 0
 0 PIN_FLD_EFFECTIVE_T  TSTAMP [0] (1337557354) Sun May 20 23:42:34 2012
 0 PIN_FLD_BALANCES      ARRAY [978] allocated 20, used 10
 1     PIN_FLD_RESERVED_AMOUNT DECIMAL [0] 0
 1     PIN_FLD_NEXT_BAL     DECIMAL [0] 0
 1     PIN_FLD_CONSUMPTION_RULE   ENUM [0] 0
 1     PIN_FLD_CURRENT_BAL  DECIMAL [0] 0
 1     PIN_FLD_SUB_BALANCES  ARRAY [0] allocated 20, used 12
 2         PIN_FLD_CONTRIBUTOR_STR    STR [0] ""
 2         PIN_FLD_VALID_TO     TSTAMP [0] (0) <null>
 2         PIN_FLD_VALID_TO_DETAILS    INT [0] 0
 2         PIN_FLD_VALID_FROM   TSTAMP [0] (1337497200) Sun May 20 07:00:00 2012
 2         PIN_FLD_VALID_FROM_DETAILS    INT [0] 0
 2         PIN_FLD_CURRENT_BAL  DECIMAL [0] 0
 2         PIN_FLD_NEXT_BAL     DECIMAL [0] 0
 2         PIN_FLD_DELAYED_BAL  DECIMAL [0] 0
 2         PIN_FLD_ROLLOVER_DATA    INT [0] 0
 2         PIN_FLD_GRANTOR_OBJ    POID [0] 0.0.0.1 /purchased_product 130995 0
 2         PIN_FLD_STATUS         ENUM [0] 1
 2         PIN_FLD_FLAGS           INT [0] 2
 1     PIN_FLD_CURRENT_TOTAL DECIMAL [0] 0
 1     PIN_FLD_CREDIT_FLOOR DECIMAL [0] NULL
 1     PIN_FLD_CREDIT_LIMIT DECIMAL [0] 0
 1     PIN_FLD_CREDIT_THRESHOLDS    INT [0] 0
 1     PIN_FLD_CREDIT_THRESHOLDS_FIXED    STR [0] ""
 0 PIN_FLD_REALTIME_CNTR    INT [0] 4
 0 PIN_FLD_BALANCES      ARRAY [1000011] allocated 20, used 10
 1     PIN_FLD_RESERVED_AMOUNT DECIMAL [0] 0
 1     PIN_FLD_NEXT_BAL     DECIMAL [0] 0
 1     PIN_FLD_CONSUMPTION_RULE   ENUM [0] 0
 1     PIN_FLD_CURRENT_BAL  DECIMAL [0] 77
 1     PIN_FLD_SUB_BALANCES  ARRAY [4] allocated 20, used 12
 2         PIN_FLD_CONTRIBUTOR_STR    STR [0] ""
 2         PIN_FLD_VALID_TO     TSTAMP [0] (0) <null>
 2         PIN_FLD_VALID_TO_DETAILS    INT [0] 0
 2         PIN_FLD_VALID_FROM   TSTAMP [0] (1337666360) Tue May 22 05:59:20 2012
 2         PIN_FLD_VALID_FROM_DETAILS    INT [0] 0
 2         PIN_FLD_CURRENT_BAL  DECIMAL [0] 77
 2         PIN_FLD_NEXT_BAL     DECIMAL [0] 0
 2         PIN_FLD_DELAYED_BAL  DECIMAL [0] 0
 2         PIN_FLD_ROLLOVER_DATA    INT [0] 0
 2         PIN_FLD_GRANTOR_OBJ    POID [0] 0.0.0.0  0 0
 2         PIN_FLD_STATUS         ENUM [0] 1
 2         PIN_FLD_FLAGS           INT [0] 2
 1     PIN_FLD_CURRENT_TOTAL DECIMAL [0] 77
 1     PIN_FLD_CREDIT_FLOOR DECIMAL [0] NULL
 1     PIN_FLD_CREDIT_LIMIT DECIMAL [0] 0
 1     PIN_FLD_CREDIT_THRESHOLDS    INT [0] 0
 1     PIN_FLD_CREDIT_THRESHOLDS_FIXED    STR [0] ""
 0 PIN_FLD_BALANCES      ARRAY [1000076] allocated 20, used 10
 1     PIN_FLD_RESERVED_AMOUNT DECIMAL [0] 0
```

```
 1     PIN_FLD_NEXT_BAL     DECIMAL [0] 0
 1     PIN_FLD_CONSUMPTION_RULE   ENUM [0] 0
 1     PIN_FLD_CURRENT_BAL  DECIMAL [0] -125.37
 1      PIN_FLD_SUB_BALANCES  ARRAY [2] allocated 20, used 12
 2         PIN_FLD_CONTRIBUTOR_STR    STR [0] ""
 2         PIN_FLD_VALID_TO     TSTAMP [0] (0) <null>
 2         PIN_FLD_VALID_TO_DETAILS    INT [0] 0
 2         PIN_FLD_VALID_FROM   TSTAMP [0] (1337497200) Sun May 20 07:00:00 2012
 2         PIN_FLD_VALID_FROM_DETAILS    INT [0] 0
 2         PIN_FLD_CURRENT_BAL  DECIMAL [0] -125.37
 2         PIN_FLD_NEXT_BAL     DECIMAL [0] 0
 2         PIN_FLD_DELAYED_BAL  DECIMAL [0] 0
 2         PIN_FLD_ROLLOVER_DATA    INT [0] 0
 2         PIN_FLD_GRANTOR_OBJ    POID [0] 0.0.0.1 /purchased_product 130995 205
 2         PIN_FLD_STATUS         ENUM [0] 1
 2         PIN_FLD_FLAGS           INT [0] 2
 1     PIN_FLD_CURRENT_TOTAL DECIMAL [0] -125.37
 1     PIN_FLD_CREDIT_FLOOR DECIMAL [0] NULL
 1     PIN_FLD_CREDIT_LIMIT DECIMAL [0] 0
 1     PIN_FLD_CREDIT_THRESHOLDS    INT [0] 0
 1     PIN_FLD_CREDIT_THRESHOLDS_FIXED    STR [0] ""
```

Operation: send is PCM_OP_SUBSCRIPTION_PURCHASE_DEAL (108)

```
Flags = 0
 0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0
 0 PIN_FLD_PROGRAM_NAME    STR [0] "NCC_BCD_Client"
 0 PIN_FLD_DEAL_INFO    SUBSTRUCT [0] allocated 20, used 1
 1     PIN_FLD_DEAL_OBJ       POID [0] 0.0.0.1 /deal 88008 4
```

Result: received for operation PCM_OP_SUBSCRIPTION_PURCHASE_DEAL (108)

```
0 PIN_FLD_POID           POID [0] 0.0.0.1 /account 128819 0

Error specified:
 location=5
 pin_errclass=1
 pin_err=81
 field=0
 rec_id=0
 reserved=0
 facility=0
 msg_id=0
 version=0
```

Appendix A

Example BCD section of the eserv.config file

This appendix contains an example of the default `BCD` section of the Convergent Charging Controller **eserv.config** file, which you can find in the **/IN/service_packages** directory on each of the SLC machines.

```
BCD = {
    bcdActionHandler = {
        serviceDomainInterfaceName = "bcdBeClient"
        clientIDString = "client1"
        loggedNotificationPeriod = 300
        lowCreditBufferTime = 10
        roundingScheme = 3
        cacheTimeout = 60
        maxOutstandingRequests = 1000
        VoucherPinLength = 4
        poidPrefix = "/service/telco/"
        accountString = "/account"

        ServiceProfileTagMapping = [
            {
                # Default config. Basic duration measured voice call
                ServiceKey = -1 # default
                BearerID = -1  # default
                ScalingFactor = 1
                BRMField = "QUANTITY" # or BYTES_UPLINK, BYTES_DOWNLINK
                BRMReqMode = "DURATION" # or VOLUME
                BrmServicePoid = "/service/telco/gsm/telephony"
                BrmObjectType = "gsm"
                UsedUnitsCumulative = false
                DefaultUnitType = "QUANTITY" # or UP_BYTES, DOWN_BYTES
            }
            {
                # Specific configuration for data calls.
                ServiceKey = 1
                BearerID = 17
                ScalingFactor = 100000 # Bytes per deci-second (=1Mb/second)
                BRMField = "BYTES_UPLINK"
                BRMReqMode = "VOLUME"
                BrmServicePoid = "/service/telco/gsm/data"
                BrmObjectType = "gsm"
                UsedUnitsCumulative = false
                DefaultUnitType = "UP_BYTES"
            }
        ]

        BrmToNccCurrencyMapping = [
            {
                NCCCode = "NZD"
                BRMNum = 554
            }
            {
                NCCCode = "EUR"
                BRMNum = 978
            }
```

```
]

BrmBadPinEdrActive = false
BrmEdrObjectType = "/voucher"
BrmBadPinResourceId = 1000011
NccInfoFieldNumber = 10000
NccInfoFieldDummyEntry = 10001

NccToBrmFieldMapping = [
    {
        NCCItem = "TARIFF_PLAN_ID"
        BRMType = "INT"
        BRMField = 10004
    }
    {
        NCCItem = "NUMBER_OF_EVENTS_ID"
        BRMType = "INT"
        BRMField = 10002
    }
    {
        NCCItem = "EXAMPLE"
        BRMType = "STRING"
        BRMField = 10005
    }
    {
        NCCItem = "VOUCHER"
        BRMType = "STRING"
        BRMField = 10007
    }
    {
        NCCItem = "PIN"
        BRMType = "STRING"
        BRMField = 10008
    }
]

BrmFieldToEdrMapping = [
    {
        BRMField = 7450
        EDRItem = "AUTHORIZATION_ID"
    }
    {
        BRMField = 3039
        EDRItem = "SESSION_ID"
    }
]

LocationNumberMapping = {
    BRMField = 1251     # Or zero to disable sending of Location Number.
    PrimaryLocationNumberProfileBlock = 18     # Call Context
    PrimaryLocationNumberProfileTag = 327692   # PT_CC_LOCATION_NUMBER. Must
    be
                                               # non-zero if BRMField is non-
    zero.
    SecondaryLocationNumberProfileBlock = 18   # Call Context
    SecondaryLocationNumberProfileTag = 327716 #
    PT_CC_LOCATION_INFO_LOCATION_NUMBER
                                               # or zero to disable secondary
    choice
}

CustomOpCodeMapping = [
    {
        BrmOpCode = 4007
        CustomOpCode = 11007
```

```
        }
        {
            BrmOpCode = 4026
            CustomOpCode = 11026
        }
    ]

    InSessionNotificationMapping = {
        ProfileBlock = 17    # NCC profile block to populate. Default = 17
        (TEMPORARY STORAGE)

        Language = {
            Description = "Preferred Language"   # Text to match in
            PIGGYBACK_NOTIFICATIONS
            ProfileTag = 37             # NCC profile tag to use (PT_LANGUAGE)
        }

        Channel = {
            Description = "Preferred Channel"    # Text to match in
            PIGGYBACK_NOTIFICATIONS
            ProfileTag = 1312050        # NCC profile tag to use
            (PT_ISN_PREF_CHANNEL)
            InSessionTrigger = [ "Email", "SMS" ] # Which channels require in-
            session trigger
        }

        Time = {
            Description = "Preferred Time"    # Text to match in
            PIGGYBACK_NOTIFICATIONS
            ProfileTag = 1312051        # NCC profile tag to use
            (PT_ISN_PREF_CHANNEL)
        }

        CreditThreshold = {
            Description = "Credit Threshold Breach"    # Text to match in
            PIGGYBACK_NOTIFICATIONS
            ProfileTag = 1312052        # NCC profile tag to use
            (PT_ISN_CT_BALANCE)
            BalanceTypeTag = 1312054    # NCC profile subtag for balance type ID
            (PT_ISN_CT_BAL_TYPE)
            BalanceNameTag = 1312055    # NCC profile subtag for balance type
            name (PT_ISN_CT_BAL_NAME)
            AmountTag = 1312053         # NCC profile subtag for balance amount
            (PT_ISN_CT_BAL_AMOUNT)
            CurrentBalanceTag = 1312056 # NCC profile subtag for current balance
            (PT_ISN_CT_BAL_CURRENT_BAL)
            GroupObjTag = 1312057       # NCC profile subtag for group object
            (PT_ISN_CT_BAL_GROUP_OBJ)
            PercentTag = 1312058        # NCC profile subtag for percent
            (PT_ISN_CT_BAL_PERCENT)
            SourceObjTag = 1312059      # NCC profile subtag for source object
            (PT_ISN_CT_BAL_SOURCE_OBJ)
            AlertTypeTag = 1312060      # NCC profile subtag for alert type
            (PT_ISN_CT_BAL_ALERT_TYPE)
            ReasonTag = 1312061         # NCC profile subtag for reason
            (PT_ISN_CT_BAL_REASON)
            CreditFloorTag = 1312062    # NCC profile subtag for credit floor
            (PT_ISN_CT_BAL_CREDIT_FLOOR)
            CreditLimitTag = 1312063    # NCC profile subtag for credit limit
            (PT_ISN_CT_BAL_CREDIT_LIMIT)
            CreditThresholdTag = 1312064   # NCC profile subtag for percent
            threshold (PT_ISN_CT_BAL_CREDIT_THRESH)
            CreditThresholdFixedTag = 1312065   # NCC profile subtag for fixed
            threshold (PT_ISN_CT_BAL_CREDIT_THRESH_FIXED)
```

```
            BalanceUnitNameTag = 1312078    # NCC profile subtag for balance unit
            name (PT_ISN_CT_BAL_UNIT_NAME)
        }

        SubscriptionExpiry = {
            Description = "Subscription Expired"    # Text to match in
            PIGGYBACK_NOTIFICATIONS
            ProfileTag = 1312066        # NCC profile tag to use
            (PT_ISN_SUB_EXPIRY)
            ExpiryDateTag = 1312067     # NCC profile subtag for expiry date
            (PT_ISN_SUB_EXPIRY_DATE)
        }

        StreamingThreshold = {
            Description = "Streaming Threshold reached"    # Text to match in
            PIGGYBACK_NOTIFICATIONS
            ProfileTag = 1312068        # NCC profile tag to use
            (PT_ISN_STREAM_THRESH)
            CurrentBalanceTag = 1312069 # NCC profile subtag for current balance
            (PT_ISN_STREAM_THRESH_CURRENT_BAL)
        }

        Balance = {
            ProfileTag = 1312070        # NCC profile tag to use (PT_ISN_BALANCE)
            BalanceTypeTag = 1312076    # NCC profile subtag for balance type ID
            (PT_ISN_BALANCE_TYPE)
            BalanceNameTag = 1312077    # NCC profile subtag for balance type
            name (PT_ISN_BALANCE_NAME)
            AmountTag = 1312071         # NCC profile subtag for amount
            (PT_ISN_BALANCE_AMOUNT)
            AvailLimitTag = 1312072     # NCC profile subtag for amount
            (PT_ISN_BALANCE_AVAIL_LIMIT)
            BalanceUnitNameTag = 1312079    # NCC profile subtag for balance unit
            name (PT_ISN_BALANCE_UNIT_NAME)
        }

        Status = {
            RatingStatusTag = 1312073   # NCC profile subtag for rating status
            (PT_ISN_RATING_STATUS)
            LifecycleStateTag = 1312074 # NCC profile subtag for lifecycle state
            (PT_ISN_LIFECYCLE_STATE)
            FailureReasonTag = 1312075  # NCC profile subtag for failure reason
            (PT_ISN_FAILURE_REASON)
        }
    } # End of InSessionNotificationMapping

} # End of bcdActionHandler

bcdBillingClient = {
    oracleUserAndPassword = "/@SCP"
    contextOpenTimeoutMilliseconds = 5000
    recoverCmPtrSeconds = 5
    maxPollMilliseconds = 1
    maxTries = 3
    latencyStatisticsInterval = 300
    recordCMIPAddressInStats  = false
    recordOpcodeInStats = false
    recordPortInStats = false

    opCodeMapping = [
        {operation = "PCM_OP_BAL_GET_BALANCE" , opCode = 3701 }
        {operation = "PCM_OP_CUST_MODIFY_CUSTOMER" , opCode = 64 }
        {operation = "PCM_OP_PYMT_TOPUP" , opCode = 3726 }
        {operation = "PCM_OP_SEARCH" , opCode = 7 }
        {operation = "PCM_OP_SUBSCRIPTION_PURCHASE_DEAL" , opCode = 108 }
```

```
            {operation = "PCM_OP_SUBSCRIPTION_READ_ACCT_PRODUCTS" , opCode = 81 }
            {operation = "PCM_OP_TRANS_ABORT" , opCode = 13 }
            {operation = "PCM_OP_TRANS_OPEN" , opCode = 12 }
            {operation = "PCM_OP_WRITE_FLDS" , opCode = 5 }
            {operation = "PCM_OP_READ_FLDS" , opCode = 4 }
            {operation = "PCM_OP_ACT_ACTIVITY" , opCode = 151 }
            {operation = "PCM_OP_CUST_POL_GET_DEALS" , opCode = 278 }
            {operation = "PCM_OP_BILL_DEBIT" , opCode = 105 }
        ]

        ConnectionManager = {
            database = 1
            enableTLS = 1
            cipherList = "SSL_RSA_WITH_AES_128_CBC_SHA"
            TLSWalletDirectory = "/IN/service_packages/BCD/wallet"
            service = "/service/pcm_client"
            cmPointers = [
                { cmPtr = "ip 192.168.111.111 12010", poolSize = 15}
                { cmPtr = "ip 192.168.111.112 12010", poolSize = 50}
                { cmPtr = "ip 192.168.111.111 12011", poolSize = 15}
                { cmPtr = "ip 192.168.111.112 12011", poolSize = 15}
            ]

        }

        defaultOperationTimeout = 250
        OperationTimeouts = [
            {operation = "PCM_OP_BAL_GET_BALANCE" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_CUST_MODIFY_CUSTOMER" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_PYMT_TOPUP" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_SEARCH" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_SUBSCRIPTION_PURCHASE_DEAL" , timeoutMilliseconds =
            250 }
            {operation = "PCM_OP_TRANS_ABORT" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_TRANS_OPEN" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_WRITE_FLDS" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_ACT_ACTIVITY" , timeoutMilliseconds = 250 }
            {operation = "PCM_OP_READ_FLDS" , timeoutMilliseconds = 250 }
        ]

    } # End of bcdBillingClient
} # end of BCD
```

Glossary of Terms

AAA

Authentication, Authorization, and Accounting. Specified in Diameter RFC 3588.

ACS

Advanced Control Services configuration platform.

API

Application Programming Interface

BFT

Billing Failure Treatment - the process that is applied if the system has lost all connections to a billing engine. It allows for limited continuation of call processing functions, if configured.

CAMEL

Customized Applications for Mobile network Enhanced Logic

This is a 3GPP (Third Generation Partnership Project) initiative to extend traditional IN services found in fixed networks into mobile networks. The architecture is similar to that of traditional IN, in that the control functions and switching functions are remote. Unlike the fixed IN environment, in mobile networks the subscriber may roam into another PLMN (Public Land Mobile Network), consequently the controlling function must interact with a switching function in a foreign network. CAMEL specifies the agreed information flows that may be passed between these networks.

CC

Country Code. Prefix identifying the country for a numeric international address.

CCS

1) Charging Control Services component.

2) Common Channel Signalling. A signalling system used in telephone networks that separates signalling information from user data.

CDR

Call Data Record

Note: The industry standard for CDR is EDR (Event Detail Record).

Connection

Transport level link between two peers, providing for multiple sessions.

Convergent

Also “convergent billing”. Describes the scenario where post-paid and pre-paid calls are handed by the same service platform and the same billing system. Under strict converged billing, post-paid subscribers are essentially treated as “limited credit pre-paid”.

CORBA

Common Object Request Broker Architecture. It is a framework that provides interoperability between objects built in different programming languages, running on different physical machines perhaps on different networks. It specifies an Interface Definition Language, and API that allows client / server interaction with the ORB.

CPU

Central Processing Unit

Diameter

A feature rich AAA protocol. Utilises SCTP and TCP transports.

DP

Detection Point

DTMF

Dual Tone Multi-Frequency - system used by touch tone telephones where one high and one low frequency, or tone, is assigned to each touch tone button on the phone.

ENUM

E.164 Number Mapping.

FOX

Fast OSA eXtensions. A TCP/IP billing protocol intended for use with external vendors. Based on OSA, it fills in functional gaps missing in OSA, and defines “combined” OSA operations to increase platform throughput. Uses a non-CORBA transport layer in order to provide enhanced fail-over and connection redundancy.

GPRS

General Packet Radio Service - employed to connect mobile cellular users to PDN (Public Data Network- for example the Internet).

GSM

Global System for Mobile communication.

It is a second generation cellular telecommunication system. Unlike first generation systems, GSM is digital and thus introduced greater enhancements such as security, capacity, quality and the ability to support integrated services.

HLR

The Home Location Register is a database within the HPLMN (Home Public Land Mobile Network). It provides routing information for MT calls and SMS. It is also responsible for the maintenance of user subscription information. This is distributed to the relevant VLR, or SGSN (Serving GPRS Support Node) through the attach process and mobility management procedures such as Location Area and Routing Area updates.

HPLMN

Home PLMN

IN

Intelligent Network

INAP

Intelligent Network Application Part - a protocol offering real time communication between IN elements.

IP

1) Internet Protocol

2) Intelligent Peripheral - This is a node in an Intelligent Network containing a Specialized Resource Function (SRF).

IP address

Internet Protocol Address - network address of a card on a computer.

ISDN

Integrated Services Digital Network - set of protocols for connecting ISDN stations.

ITU

International Telecommunication Union

IVR

Interactive Voice Response - systems that provide information in the form of recorded messages over telephone lines in response to user input in the form of spoken words or, more commonly, DTMF signalling.

MAP

Mobile Application Part - a protocol which enables real time communication between nodes in a mobile cellular network. A typical usage of the protocol would be for the transfer of location information from the VLR to the HLR.

Messaging Manager

The Messaging Manager service and the Short Message Service components of Oracle Communications Convergent Charging Controller product. Component acronym is MM (formerly MMX).

MIN

Mobile Identification Number, also known as an MSID.

MM

Messaging Manager. Formerly MMX, see also *XMS* (on page 117) and *Messaging Manager* (on page 113).

MO

Mobile Originated

MOC

Managed Object Class

MS

Mobile Station

MSC

Mobile Switching Centre. Also known as a switch.

MSID

Mobile Subscriber Identification, also known as an MIN.

MSISDN

Mobile Station ISDN number. Uniquely defines the mobile station as an ISDN terminal. It consists of three parts; the country code (CC), the national destination code (NDC) and the subscriber number (SN).

MT

Mobile Terminated

MTC

Mobile Terminated Call. The part of the call associated with a subscriber receiving an inbound call.

ORB

Object Request Broker. Within an Object based communication system, an ORB keeps track of the actual addresses of all defined objects and thus is used to route traffic to the correct destination. The CORBA defines the ORB in a series of standards enabling different platforms to share common information.

OSA

Open Service Access provides a standard interface through which developers can design services that may interact with functions within the network.

PI

Provisioning Interface - used for bulk database updates/configuration instead of GUI based configuration.

PIN

Personal Identification Number

PLMN

Public Land Mobile Network

SCP

Service Control Point. Also known as SLC.

SCTP

Stream Control Transmission Protocol. A transport-layer protocol analogous to the TCP or User Datagram Protocol (UDP). SCTP provides some similar services as TCP (reliable, in-sequence transport of messages with congestion control) but adds high availability.

Session

Diameter exchange relating to a particular user or subscriber access to a provided service (for example, a telephone call).

SGSN

Serving GPRS Support Node

SIM

Usually referred to as a SIM card, the Subscriber Identity Module is the user subscription to the mobile network. The SIM contains relevant information that enables access onto the subscripted operator's network.

SIP

Session Initiation Protocol - a signaling protocol for Internet conferencing, telephony, event notification and instant messaging. (IETF)

SLC

Service Logic Controller (formerly UAS).

SLEE

Service Logic Execution Environment

SMS

Depending on context, can be:

- Service Management System hardware platform
- Short Message Service
- Service Management System platform
- Convergent Charging Controller Service Management System application

SN

Service Number

SRF

Specialized Resource Function – This is a node on an IN which can connect to both the SSP and the SLC and delivers additional special resources into the call, mostly related to voice data, for example play voice announcements or collect DTMF tones from the user. Can be present on an SSP or an Intelligent Peripheral (IP).

SSL

Secure Sockets Layer protocol

SSP

Service Switching Point

STR

Session message: Session-Termination Request

System Administrator

The person(s) responsible for the overall set-up and maintenance of the IN.

TCP

Transmission Control Protocol. This is a reliable octet streaming protocol used by the majority of applications on the Internet. It provides a connection-oriented, full-duplex, point to point service between hosts.

Telco

Telecommunications Provider. This is the company that provides the telephone service to customers.

Telecommunications Provider

See Telco.

TLS

Transport Layer Security. Cryptographic protocol used to provide secure communications. Evolved from SSL.

USSD

Unstructured Supplementary Service Data - a feature in the GSM MAP protocol that can be used to provide subscriber functions such as Balance Query.

VLR

Visitor Location Register - contains all subscriber data required for call handling and mobility management for mobile subscribers currently located in the area controlled by the VLR.

Voice Call

The term “voice call” in this document is intended to denote any call controlled by CAMEL or INAP InitialDP. In practice this also includes fax calls, data-over-voice calls, and also includes 3G voice and video conference calls.

VWS

Oracle Voucher and Wallet Server (formerly UBE).

XMS

Three letter code used to designate some components and path locations used by the Oracle Communications Convergent Charging Controller *Messaging Manager* (on page 113) service and the Short Message Service. The published code is *MM* (on page 113) (formerly MMX).

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

X