Copyright

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.
Contents

Copyright ........................................................................................................................................ ii
About this Document ....................................................................................................................... v
Document Conventions .................................................................................................................. vi

Chapter 1
About the BRM Charging Driver
Overview........................................................................................................................................... 1
Overview of the BRM Charging Driver ........................................................................................... 2
BRM integration summary .................................................................................................................. 11

Chapter 2
Installing the BRM Charging Driver
Overview of installing the BRM Charging Driver ............................................................................ 13
About installing the BRM Charging Driver ....................................................................................... 14

Chapter 3
Configuring NCC for the BRM Charging Driver
About configuring NCC for the BRM Charging Driver ................................................................. 17
Summary of NCC Configuration Tasks ............................................................................................ 18
Creating the BRM Domain ................................................................................................................ 19
Configuring replication ..................................................................................................................... 20
About editing eserv.config parameters ......................................................................................... 21
Configuring bcdActionHandler ....................................................................................................... 22
Configuring bcdBillingClient ........................................................................................................... 33
Configuring calls, events, and vouchers ......................................................................................... 40
Modifying the BCD Client startup script ......................................................................................... 43
Creating balance type mappings .................................................................................................... 45

Chapter 4
Configuring BRM for the BRM Charging Driver
About configuring BRM .................................................................................................................... 49
Adding custom fields ....................................................................................................................... 50
Adding storable classes ................................................................................................................... 53
Creating header and library files for the custom classes ............................................................... 54
Generating the custom JAR file ....................................................................................................... 55
Modifying the BRM configuration files ........................................................................................... 56

Chapter 5
Creating products and deals
About creating products and deals ............................................................................................... 59
Creating a product and deal for voice calls ..................................................................................... 60
Creating a product and deal for data calls ....................................................................................... 61
Creating a product and deal for named events ................................................................................ 62
Creating BRM vouchers ................................................................................................................... 65
Creating a customer ........................................................................................................................ 66

Chapter 6
Generating statistics and reports
Overview of statistics and reports ................................................................................................. 67
About statistics and reports ............................................................................................................. 68
Chapter 7
Usage scenarios

About usage scenarios ................................................................. 71
A voice call charged against BRM ............................................... 72
An SMS call charged against BRM ............................................ 83
Recharge using BRM vouchers .................................................. 87
Recharge using VWS vouchers ................................................ 95
Data charging ........................................................................... 101
Other messages ........................................................................ 107

Appendix

Overview ..................................................................................... 111
Appendix A ................................................................................ 112
Glossary of Terms ..................................................................... 119
Index .......................................................................................... 123
About this Document

Scope
The scope of this document is to describe the process of installing, configuring and administering the NCC BRM Charging Driver. In some cases, this document refers you to existing NCC 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 Oracle's Network Charging and Control system and Oracle's Billing and Revenue Management application and it 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 Oracle's Billing and Revenue Management 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:

- NCC NCC Installation Guide
- NCC Service Management System User's Guide
- NCC Service Management System Technical Guide
- NCC Service Logic Execution Environment Technical Guide
- NCC 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

The following terms and typographical conventions are used in NCC documentation.

<table>
<thead>
<tr>
<th>Formatting convention</th>
<th>Type of information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Special Bold</strong></td>
<td>Items you must select, such as names of tabs. Names of database tables and fields.</td>
</tr>
<tr>
<td><em>Italics</em></td>
<td>Name of a document, chapter, topic or other publication. Emphasis within text.</td>
</tr>
<tr>
<td><strong>Button</strong></td>
<td>The name of a button to click or a key to press. <strong>Example:</strong> To close the window, either click <strong>Close</strong>, or press <strong>Esc</strong>.</td>
</tr>
<tr>
<td><strong>Key+Key</strong></td>
<td>Key combinations for which the user must press and hold down one key and then press another. <strong>Example:</strong> <strong>Ctrl+P</strong>, or <strong>Alt+F4</strong>.</td>
</tr>
<tr>
<td><strong>Monospace</strong></td>
<td>Text that you must type and examples of code or standard output.</td>
</tr>
<tr>
<td><strong>variable</strong></td>
<td>Used to indicate variables or text that should be replaced.</td>
</tr>
<tr>
<td><strong>menu option &gt; menu option &gt;</strong></td>
<td>Used to indicate the cascading menu option to be selected, or the location path of a file. <strong>Example:</strong> <strong>Operator Functions &gt; Report Functions</strong> <strong>Example:</strong> /IN/html/SMS/HelpText/</td>
</tr>
<tr>
<td><strong>hypertext link</strong></td>
<td>Used to indicate a hypertext link on an HTML page.</td>
</tr>
</tbody>
</table>

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

## Icons

The following icons are used in NCC guides as visual cues to draw attention to important information.

- **Note:** Indicates useful and complementary information, such as an explanation, comment, or short expansion of the text object that is intended to catch your attention.

- **Tip:** Indicates practical but non-essential information that makes the solution easier to use or operate (for example, a keyboard shortcut, or alternative way to perform a step in a procedure).

- **Warning:** Indicates a caution. If this information is ignored, it could possibly cause irreversible damage to the equipment, data, or software.
Chapter 1

About the BRM Charging Driver

Overview

Introduction

This chapter describes the architecture and the main features of the BRM Charging Driver, which integrates the Oracle Communications Network Charging and Control (NCC) application with the Oracle Communications Billing and Revenue Management (BRM) system to provide a complete network charging and account settlement system.

For architectural overviews and descriptions of Oracle Communications Network Charging and Control (NCC) and Billing and Revenue Management (BRM) systems, see the documentation sets for each of these products. For an architectural overview of NCC, see *NCC 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 ................................................. 2
BRM Integration summary .............................................................. 11
Overview of the BRM Charging Driver

About BRM
Oracle Communications Billing and Revenue Management (BRM) system is an end-to-end 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
- Providing rating by measuring events and calculating charges
- Creating bills
- Managing payments and accounts receivable

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

About integrating NCC and BRM
Integrating BRM with NCC provides customers with a charging solution from prepaid to advanced online and offline charging and account settlement.

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

BRM stores the wallet and subscriber data and you can choose to store vouchers either on the NCC Voucher and Wallet Server (VWS) or in the BRM database. The CCS software communicates with BRM through the Portal Communication Module (PCM) API.

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

- NCC 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 NCC.
- 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.

The BRM Charging Driver does not support the following NCC capabilities on BRM:

- Named events with a negative number of events
- Direct time charge in credit mode

BRM and NCC components
The following diagram illustrates the main components of an NCC system that is integrated with BRM. The new 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 NCC and BRM.

Continued on next page
The following sections describe the main components of an NCC system that is integrated with BRM:

Continued on next page
Overview of the BRM Charging Driver, Continued

The BRM Connection Manager

NCC connects to BRM through the BRM Connection Manager, which runs as a daemon on a dedicated BRM Connection Manager machine. When NCC requests a connection, a parent Connection Manager process spawns a child process to handle the connection. After that, all communication flows from NCC 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 NCC 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 doesn't 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 NCC to utilize the BRM Connection Manager, see Configuring Connections to the BRM Connection Manager.

The Portal Communications Module API

All NCC access to BRM and BRM data is done through BRM's 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 opcodes (operation codes) to BRM to perform various operations. An opcode associates an integer 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 create an object in BRM, the PCM_OP_CREATE_OBJECT opcode requires an input flist that includes all the fields that an object of that class requires.

So to request a particular operation from BRM, NCC invokes a PCM macro that passes BRM the necessary opcode and BRM then calls the corresponding function to perform the operation. BRM responds to NCC operation requests by returning an output flist.

The 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 Communication Module (PCM) API to set up connections to the BRM connection managers and to send and receive Portal Communication Process (PCP) messages. 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 startup, 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 SLC database. 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.

Continued on next page
Overview of the BRM Charging Driver, Continued

<table>
<thead>
<tr>
<th>The BCD Client (continued)</th>
<th>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.</th>
</tr>
</thead>
</table>
| The 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 NCC's 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 BCD 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 introduces new reports to report on events either initiated or observed by the BCD Client. Reporting is done through the standard NCC SMS reporting mechanism.  
For information on generating BRM Charging Driver statistics and reports, see *Generating Statistics and Reports*. |

*Continued on next page*
Overview of the BRM Charging Driver, Continued

The following NCC components are also relevant in the integration of NCC and BRM:

- **SLEE and slee_acs**
  The SLEE is the Service Logic Execution Environment (SLEE), which 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 NCC SLEE API to send and receive SLEE events - thus, the name slee_acs. For more information on the slee_acs process, see the Advanced Control Services Technical Guide.

- **The Billing Engine (BE) Client**
  The Billing Engine Client provides the interface that processes requests from the slee_acs process to the NCC Voucher and Wallet Server.

- **Voucher and Wallet Services (VWS)**
  Voucher and Wallet Services (VWS) is NCC's real-time charging and subscriber account management component. The VWS solution maintains voucher, wallet and reservation details in a database on the Voucher & Wallet Server. The role of VWS is to manage all the billing and charging information associated with call processing. In essence it is a precursor and more limited alternative to BRM for NCC charging.

- **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 identical functions to the BCD Client and the BCD actions shared library, except that SLEE FOX events replace SLEE BCD events, FOX messages replace PCP 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 Intelligent Network Application Part (INAP) language. In doing so, they translate network protocols such as SIP, MAP, or Diameter into INAP so that a common NCC service logic is possible, independent of the network protocol.

<table>
<thead>
<tr>
<th>BRM Charging Driver features</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session Charging</td>
<td>The ability to charge for voice and data sessions using the NCC Universal Attempt Termination with Billing (UATB) feature node and BRM</td>
</tr>
<tr>
<td>Time Units</td>
<td>The ability to charge for sessions using the time unit type</td>
</tr>
</tbody>
</table>

Continued on next page
## Overview of the BRM Charging Driver, Continued

<table>
<thead>
<tr>
<th>BRM Charging Driver features (continued)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Units</td>
<td>The ability to charge for sessions using the data unit type</td>
</tr>
<tr>
<td>Funds Exhaustion</td>
<td>Rejection of further quota requests (duration or volume) when BRM indicates that funds are exhausted</td>
</tr>
<tr>
<td>Network Ends Session</td>
<td>The UATB feature node will report used units (data or time) to BRM when the network indicates that the session terminated</td>
</tr>
<tr>
<td>Cost of Session / Remaining Balance</td>
<td>The ability to report remaining balance and cost of session at the end of a session</td>
</tr>
<tr>
<td>Voice Call Cost</td>
<td>The ability to use the Voice Call Cost feature node to request play a message that states the cost of a session</td>
</tr>
<tr>
<td>SMS Call Info</td>
<td>The ability to use the SMS Call Info feature node to request BRM to send the cost of a session</td>
</tr>
<tr>
<td>Cumulative Info</td>
<td>The ability to use the Cumulative Info feature node to play a message that states the balances of an account on BRM</td>
</tr>
<tr>
<td>SMS Account Balances</td>
<td>The ability to use the SMS Account Balances feature node to send the balances of an account that resides on BRM</td>
</tr>
<tr>
<td>Account State Branch</td>
<td>The ability to use the Account State Branch feature node to branch on the state of an account that resides on BRM</td>
</tr>
<tr>
<td>Account Status</td>
<td>The ability in the control plan to play the status and balance of an account on BRM</td>
</tr>
<tr>
<td>Billing Failure Treatment</td>
<td>The ability to specify in the control plan the treatment to use when the UATB feature node cannot contact BRM</td>
</tr>
<tr>
<td>Billing Failure Treatment EDRs</td>
<td>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</td>
</tr>
<tr>
<td>Direct Named Event</td>
<td>The ability to use the Billable Event feature node to create a Direct Named Event against BRM</td>
</tr>
<tr>
<td>Named Event Reservation</td>
<td>The ability to use the Billable Event feature node to create a Named Event Reservation against BRM</td>
</tr>
<tr>
<td>Confirm Named Event</td>
<td>The ability to use the Billable Event feature node to confirm a Named Event Reservation against BRM</td>
</tr>
<tr>
<td>Revoke Named Event</td>
<td>The ability to use the Billable Event feature node to revoke a Named Event Reservation against BRM</td>
</tr>
<tr>
<td>Recharge of BRM Vouchers</td>
<td>The ability to use the Voucher Redeem and Voucher Recharge feature nodes to recharge BRM vouchers against BRM accounts</td>
</tr>
<tr>
<td>IVR Redemption and Recharge of BRM Vouchers</td>
<td>Using Interactive Voice Response (IVR) as the input method, the ability to use the Voucher Redeem and Voucher Recharge feature nodes to redeem and recharge BRM vouchers against BRM accounts. NCC instructs the IVR to play announcements and user responses.</td>
</tr>
<tr>
<td>IVR Playing of Redeemed Accounts</td>
<td>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</td>
</tr>
<tr>
<td>SMS Redemption and Recharge of BRM Vouchers</td>
<td>Using SMS as the input method, the ability to use the Voucher Redeem and Voucher Recharge feature nodes to redeem and recharge BRM vouchers against BRM accounts</td>
</tr>
</tbody>
</table>

Continued on next page
### Overview of the BRM Charging Driver, Continued

<table>
<thead>
<tr>
<th>BRM Charging Driver features (continued)</th>
<th>Overview</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>USSD Redemption and Recharge of BRM Vouchers</strong></td>
<td>Using Unstructured Supplementary Service Data (USSD) as the input method, the ability to use the Voucher Redeem and Voucher Recharge feature nodes to recharge BRM vouchers against BRM accounts. This feature does not include providing information on recharged amounts by way of USSD. A USSD message is part of an interactive text dialog between a subscriber and the server.</td>
</tr>
<tr>
<td><strong>Web Services Redemption and Recharge of BRM Vouchers</strong></td>
<td>Triggered through web services, the ability to use the Voucher Redeem and Voucher Recharge feature nodes to redeem and recharge BRM vouchers against BRM accounts.</td>
</tr>
<tr>
<td><strong>Redemption and Recharge of VWS Vouchers</strong></td>
<td>The ability to use the Voucher Redeem and Voucher Recharge feature nodes to redeem and recharge VWS vouchers against BRM accounts.</td>
</tr>
<tr>
<td><strong>IVR Redemption and Recharge of VWS Vouchers</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>IVR Playing of Redeemed Amounts for VWS Vouchers</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>SMS Recharge of VWS Vouchers</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>USSD Recharge of VWS Vouchers</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Web Services Recharge of VWS Vouchers</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Voucher Redeem Failure Records</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Reservationless Charging or Refund</strong></td>
<td>The ability to use the DUCR feature node to debit based on a usage amount (duration or volume) without reservation of quota.</td>
</tr>
<tr>
<td><strong>Post Call Billing</strong></td>
<td>The ability to use the DUCR feature node to charge for voice calls against BRM accounts after the calls have finished.</td>
</tr>
<tr>
<td><strong>Credit Transfer</strong></td>
<td>The ability to use the Credit Wallet Transfer feature node to perform a credit transfer between two BRM accounts.</td>
</tr>
</tbody>
</table>
Overview of the BRM Charging Driver, Continued

BRM Charging Driver features (continued)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCC Defined Credit Transfer</td>
<td>The ability to use the Credit Wallet Transfer feature node to perform a credit transfer between two BRM accounts using a billable event that is mapped to a deal in BRM and a voucher type defined in the NCC system.</td>
</tr>
<tr>
<td>Rating Guidance</td>
<td>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.</td>
</tr>
<tr>
<td>Usage Tracking</td>
<td>The ability to track service usage for BRM accounts using NCC tracker wallets.</td>
</tr>
<tr>
<td>Connection Manager Details</td>
<td>The ability to specify details of how NCC connects to BRM connection managers.</td>
</tr>
<tr>
<td>Connection Manager Addresses</td>
<td>The ability to specify the IP address of each BRM connection manager to which NCC connects.</td>
</tr>
<tr>
<td>Connection Manager Maximum Connections</td>
<td>The ability to specify the maximum number of connections from each NCC process to BRM.</td>
</tr>
<tr>
<td>PCM API</td>
<td>The ability to use the PCM API to communicate with BRM connection managers.</td>
</tr>
<tr>
<td>BRM EDRs</td>
<td>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.</td>
</tr>
</tbody>
</table>

Unavailable BRM Charging Driver features

When BRM is integrated with NCC through the BRM Charging Driver, you cannot use the following NCC 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

Continued on next page
Overview of the BRM Charging Driver, Continued

Unavailable BRM Charging Driver features (continued)

The following Prepaid Charging features, or capabilities, are not available in a combined NCC 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 the CCS Feature Nodes User’s Guide.
BRM integration summary

The process of integrating NCC 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*.

2. Configuring NCC for the BRM Charging Driver.
   For information on configuring NCC, see *Configuring NCC for the BRM Charging Driver*.

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*.

4. Creating Products and Deals.
   For information on creating products and deals, see *Creating Products and Deals*.

5. Generating statistics and reports.
   For information on generating statistics and reports, see *Generating Statistics and Reports*.

BRM Charging Driver
Technical Guide
# Chapter 2

## Installing the BRM Charging Driver

### Overview of installing the BRM Charging Driver

<table>
<thead>
<tr>
<th>Introduction</th>
<th>This chapter explains how to install the BRM Charging Driver components.</th>
</tr>
</thead>
<tbody>
<tr>
<td>In this chapter</td>
<td>This chapter contains the following topics.</td>
</tr>
<tr>
<td></td>
<td>About installing the BRM Charging Driver.................................14</td>
</tr>
</tbody>
</table>
About installing the BRM Charging Driver

The general platform requirements for the BRM Charging Driver components are the same as they are for NCC. For information on the general platform requirements for NCC, see Network Charging and Control Installation Guide.

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

To install the BRM Charging Driver, you must install the following components:

- The **bcdSms** package installs the BRM Charging Driver software on an SMS machine. This package has the following contents:
  - Definitions for BRM Charging Driver statistics
  - BRM Charging Driver statistics report scripts as runnable SMS reports
  - Alarm definitions for alarms produced by the BRM Charging Driver processes
  - Profile tag definitions for new tags defined by the BRM Charging Driver
  - A new feature node set containing all the CCS nodes in a combined NCC-BRM system
  - A CCS domain type called BCD

- **Portal Development Toolkit 7.5** must be installed in `/opt/portal/7.5/PortalDevKit/lib` as a prerequisite to installing **bcdScp** to avoid **bcdScp** install failing.

- The **bcdScp** package installs the BRM Charging Driver binaries and shared libraries on an SLC machine and adds a BCD section to the `eserv.config` file.

The **bcdSms** and **bcdScp** packages are part of the NCC 4.4.1.1.0 Patchset which can be found on My Oracle Support at https://support.oracle.com/CSP/main/article?cmd=show&type=NOT&id=1370042.1

The patchset .zip file will contain separate .zip files for each module being patched as well as a .zip file for the **bcdSms** and **bcdScp** packages.

*Continued on next page*
About installing the BRM Charging Driver, Continued

Installing the `bcdSms` package requires 1 megabyte of disk space. Copy and extract the `bcdSms` package to the SMS machine where you want to install it.

To install the `bcdSms` package:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | From a command line, run the following command, which assumes that you are in the same location (.) where you extracted the files:  
   `pkgadd -d . bcdSms`  
   The script verifies system information and disk space requirements and displays the following message:  
   The package contains scripts which will be executed with super-user permission during the process of installing this package.  
   Do you want to continue with the installation of bcdSms [y, n?]  
   Answer `y` to continue.  
   The script installs `bcdSms` in the following directory:  
   `/IN/service_packages/BCD`  
   As a post-processing step, the script gives you the option of updating the SMF database with data for the domain, statistics, alarms, privileges, and so on. The script displays the following messages:  
   The database install only needs to be done if this is the primary node of a cluster, or a non-clustered machine.  
   *Install the database? [y,n?]*  
   Answer `y` if you are installing `bcdSms` on the primary node in a cluster or on a machine that is not part of a cluster and you want to update the SMF database.  
   If you answer `y` to install the database, the script displays the following prompt to ask you to enter the password for the SMF user:  
   *Please enter the password for the SMF user on the SMF instance.*  
   Enter the password for the SMF user to continue.  
   When the script completes successfully, it displays the following messages:  
   `bcdSms` install is complete  
   Installation of `bcdSms` was successful.  
   The output of the installation process is written to the `bcdSms` installation log file, which is `/IN/service_packages/BCD/bcdSms/bcdSms.install.log`. |

Continued on next page
About installing the BRM Charging Driver, Continued

Installing the `bcdScp` package requires 25 megabytes of disk space. Copy and extract the `bcdScp` package to the SLC machine where you want to install it.

To install the `bcdScp` package:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Install the Portal Development Toolkit 7.5 in <code>/opt/portal/7.5/PortalDevKit/lib</code></td>
</tr>
</tbody>
</table>
| 2    | From a command line, run the following command on the SLC machine, which assumes that you are in the same location (.) where you extracted the files:  
  `pkgadd -d . bcdScp`  
  The script verifies the system information and disk space requirements and displays the following message:  
  The package contains scripts which will be executed with super-user permission during the process of installing this package.  
  Do you want to continue with the installation of bcdScp [y, n]? |
| 3    | Answer y to continue.  
  The script installs `bcdScp` in the following directory:  
  `/IN/service_packages/BC`  
  When the script completes successfully, it displays the following messages:  
  `bcdScp install is complete`  
  `Installation of bcdScp was successful.` |

The output of the installation process is written to the `bcdScp` installation log file, which is `/IN/service_packages/BCD/bcdScp.install.log`: 
Chapter 3

Configuring NCC for the BRM Charging Driver

About configuring NCC for the BRM Charging Driver

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

This chapter contains the following topics:

- Summary of NCC Configuration Tasks .......................................................... 18
- Creating the BRM Domain ........................................................................... 19
- Configuring replication ............................................................................... 20
- About editing eserv.config parameters .................................................... 21
- Configuring bcdActionHandler ................................................................. 22
- Configuring bcdBillingClient ..................................................................... 33
- Configuring calls, events, and vouchers .................................................... 40
- Modifying the BCD Client startup script ................................................... 43
- Creating balance type mappings ............................................................... 45
## Summary of NCC Configuration Tasks

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

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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 just Wallets, in which case NCC will use NCC vouchers to recharge BRM accounts.</td>
</tr>
</tbody>
</table>
| 2    | Configure replication for the 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. Setting up multiple BCD Client startup scripts (`bcdBeClient.sh`) if you have a large system and you want to run more processes. |
| 5    | Start the SLEE. |
| 6    | Create balance types in the NCC Service Management System to match BRM resource IDs and create balance type mappings between NCC balance types and BRM resource IDs, which are effectively BRM balance types. For more information, see *Creating Balance Type Mappings*. |
### Creating the BRM Domain

Follow these steps to create a BRM domain.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log in to the Service Management System (SMS).</td>
</tr>
<tr>
<td>2</td>
<td>From the Services menu, select Prepaid Charging, then Service Management. For more information on Prepaid Charging, see CCS User's Guide.</td>
</tr>
<tr>
<td>3</td>
<td>Select the Domain tab to view the current list of NCC domains.</td>
</tr>
<tr>
<td>4</td>
<td>Click New to create a new domain.</td>
</tr>
</tbody>
</table>
| 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

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 check the replication settings, select the Operator Functions menu in the NCC 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

The eserv.config file is a shared configuration file from which many NCC 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 the bcdBillingClient.

The eserv.config file resides in the following directory on each of the SLC servers:

/IN/service_packages/.

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 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 NCC configuration parameters for the BRM Charging Driver. It is provided as a reference for operators when they 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

The bcdActionHandler structure resides in the BCD section of the eserv.config file and defines the ServiceProfileTagMapping array, the BrmToNccCurrencyMapping array, and the NccToBrmFieldMapping array, as well as additional individual parameters.

The ServiceProfileTagMapping array allows you to map ServiceKey and BearerID combinations, which identify the type of NCC 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 fall back 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 (equates to 1Mb per second)
    BRMField = "BYTES_UPLINK"
    BRMReqMode = "VOLUME"
    BrmServicePoid = "/service/telco/gsm/data"
    BrmObjectType = "gsm/ncc"
    UsedUnitsCumulative = false
    DefaultUnitType = "UP_BYTES"
  }
]
```

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

NCC always calls a function called InitialTimeReservation() even if it wants to reserve an amount of data. If it wants 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.

However, BRM does not understand bytes so NCC turns seconds into bytes using a scaling factor when sending PCM_OP_TCF_AAA_AUTHORIZE to BRM. It also sets the ReqMode parameter in PCM_OP_TCF_AAA_AUTHORIZE to VOLUME to let BRM know it is trying to reserve volume.

The fields in the ServiceProfileTagMapping array items have the following significance:

Continued on next page
Configuring bcdActionHandler, Continued

Mapping NCC sessions to BRM services (continued)

ServiceKey

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>See Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>The service key from the InitialDP that triggered this session. The BRM Charging Driver looks up the relevant 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 Message Manager product (for SMS) and the DCA product (for data) translate their protocols into INAP and send InitialDPs also.</td>
</tr>
<tr>
<td>Type:</td>
<td>Integer</td>
</tr>
<tr>
<td>Optionality:</td>
<td>Required</td>
</tr>
<tr>
<td>Allowed:</td>
<td>0 to 2147483647</td>
</tr>
<tr>
<td>Default:</td>
<td>-1</td>
</tr>
<tr>
<td>Example:</td>
<td>ServiceKey = 1</td>
</tr>
</tbody>
</table>

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

BearerID

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>See Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>The ITC of the bearer capability provided in the InitialDP and used by NCC to determine the nature of the session - for example, voice vs data. Use with ServiceKey as a key for ServiceProfileTagMapping.</td>
</tr>
<tr>
<td>Type:</td>
<td>Integer</td>
</tr>
<tr>
<td>Optionality:</td>
<td>Required</td>
</tr>
<tr>
<td>Allowed:</td>
<td>0 to 31</td>
</tr>
<tr>
<td>Default:</td>
<td>-1</td>
</tr>
<tr>
<td>Example:</td>
<td>BearerID = 17</td>
</tr>
</tbody>
</table>

**Note:** NCC cannot receive a BearerID with a value of -1. It is a special value meaning 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.

Continued on next page
Configuring bcdActionHandler, Continued

Mapping NCC sessions to BRM services (continued)

**ScalingFactor**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>See example.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>Determines the conversion rate between BRM's defined unit for QUANTITY, such as bytes, for example, and deciseconds.</td>
</tr>
<tr>
<td>Type:</td>
<td>Integer</td>
</tr>
<tr>
<td>Optionality:</td>
<td>Required</td>
</tr>
<tr>
<td>Allowed:</td>
<td>Any positive integer</td>
</tr>
<tr>
<td>Default:</td>
<td>None.</td>
</tr>
<tr>
<td>Example:</td>
<td>ScalingFactor = 1</td>
</tr>
</tbody>
</table>

**BRMField**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>See Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>This field is required and must be set to QUANTITY, BYTES_UPLINK or BYTES_DOWNLINK. In the 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.</td>
</tr>
<tr>
<td>Type:</td>
<td>String</td>
</tr>
<tr>
<td>Optionality:</td>
<td>Required</td>
</tr>
<tr>
<td>Allowed:</td>
<td>QUANTITY, BYTES_UPLINK, BYTES_DOWNLINK</td>
</tr>
<tr>
<td>Default:</td>
<td>None.</td>
</tr>
<tr>
<td>Example:</td>
<td>BRMField = &quot;QUANTITY&quot;</td>
</tr>
</tbody>
</table>

**BRMReqMode**

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>See Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>The unit for which we are charging.</td>
</tr>
<tr>
<td>Type:</td>
<td>String</td>
</tr>
<tr>
<td>Optionality:</td>
<td>Required</td>
</tr>
<tr>
<td>Allowed:</td>
<td>DURATION or VOLUME</td>
</tr>
<tr>
<td>Default:</td>
<td>None</td>
</tr>
<tr>
<td>Example:</td>
<td>BRMReqMode = &quot;DURATION&quot;</td>
</tr>
</tbody>
</table>
Configuring bcdActionHandler, Continued

Mapping NCC sessions to BRM services (continued)

**BrmServicePoid**

Syntax: See Example
Description: This parameter is used in the PIN_FLD_POID parameter of the PCM_OP_TCF_AAA_AUTHORIZE operation.
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 of a PCM_OP_TCF_AAA_AUTHORIZE operation.
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 used bytes 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

Continued on next page
Configuring bcdActionHandler, Continued

Mapping NCC sessions to BRM services (continued)

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"

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

The following example illustrates format of items in the BrmToNccCurrencyMapping array:

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

For more information on currencies and synchronizing monetary transactions between NCC and BRM, see Creating Balance Type Mappings.

Mapping NCC currency codes to BRM values

Mapping NCC information to BRM fields

The NccToBrmFieldMapping array maps NCC event detail record (EDR) information items, as well as other NCC items, to BRM fields and their associated types. Each field that is defined can be sent in an NCC information structure within opcodes that are sent to BRM. You trigger such operations 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 these items and perform the related BRM compilations before attempting to use these custom fields. For information on creating BRM custom fields for NCC information items, see About Adding Custom Fields.

Continued on next page
Mapping NCC information to BRM fields (continued)

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

- **NCCItem** - The name of the NCC 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.

```plaintext
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 }
]
```

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.

```plaintext
LocationNumberMapping = {
    { Primary = "PT_CC_LOCATION_NUMBER", Secondary = "NONE" },
    { Primary = "NONE", Secondary = "PT_CC_LOCATION_INFO_LOCATION_NUMBER" },
    { Primary = "NONE", Secondary = "NONE" }
}
```

Continued on next page
Configuring bcdActionHandler, Continued

Mapping the location number (continued)

```plaintext
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
```

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 was 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`

Continued on next page
Configuring bcdActionHandler, Continued

Configuring additional bcdActionHandler Parameters (continued)

**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 billable 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"

---

Continued on next page
Configuring bcdActionHandler, Continued

Configuring additional bcdActionHandler Parameters (continued)

**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

Continued on next page
Configuring bcdActionHandler, Continued

Configuring additional bcdActionHandler Parameters (continued)

NccInfoFieldNumber

Syntax: See Example
Description: Specifies the BRM field ID of the flist substruct field under which all the NCC 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 is "/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

Continued on next page
Configuring **bcdActionHandler**, Continued

**Configuring additional bcdActionHandler Parameters (continued)**

**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 startup scripts, `serviceDomainInterfaceName` specifies the prefix of the interface name. For example, for the following `INTERFACE` parameters in `SLEE.cfg`, which are abbreviated here:

```plaintext
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*.

**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

The BCD section of the eserv.config file also contains the bcdBillingClient structure, which defines opcode mappings, the OperationTimeouts array, and the ConnectionManager structure, as well as additional individual parameters.

Mapping opcodes
The OpCodeMapping section of the bcdBillingClient structure specifies the following opcode mapping definitions to associate opcode integers with operation macros that NCC passes to BRM to perform authorization and accounting operations.

Note: You should not need to change this section:

<table>
<thead>
<tr>
<th>OpCodeMapping</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>operation = &quot;PCM_OP_BAL_GET_BALANCE&quot;, opCode = 3701</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_CUST MODIFY_CUSTOMER&quot;, opCode = 64</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_PYMT TOPUP&quot;, opCode = 3726</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_SEARCH&quot;, opCode = 7</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_SUBSCRIPTION_PURCHASE_DEAL&quot;, opCode = 108</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_TCF AAA AUTHORIZE&quot;, opCode = 4002</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_TCF AAA CANCEL AUTHORIZATION&quot;, opCode = 4004</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_TCF AAA QUERY_BALANCE&quot;, opCode = 4104</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_TCF AAA STOP ACCOUNTING&quot;, opCode = 4007</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_TCF AAA STOP ACCOUNTING PREP_INPUT&quot;, opCode = 4031</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_TCF AAA UPDATE AND REAUTHORIZE&quot;, opCode = 4026</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_TRANS_ABORT&quot;, opCode = 13</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_TRANS_OPEN&quot;, opCode = 12</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_WRITE_FLDS&quot;, opCode = 5</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_READ_FLDS&quot;, opCode = 4</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_ACT_ACTIVITY&quot;, opCode = 151</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_CUST POL GET DEALS&quot;, opCode = 278</td>
<td></td>
</tr>
<tr>
<td>operation = &quot;PCM_OP_BILL DEBIT&quot;, opCode = 105</td>
<td></td>
</tr>
</tbody>
</table>

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
```
Specifying operation timeouts (continued)

```c
defaultOperationTimeout = 600;

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_TCF_AAA_AUTHORIZE", timeoutMilliseconds = 100},
  {operation = "PCM_OP_TCF_AAA_QUERY_BALANCE", timeoutMilliseconds = 20},
  {operation = "PCM_OP_TCF_AAA_STOP_ACCOUNTING", timeoutMilliseconds = 300},
  {operation = "PCM_OP_BAL_STOP_ACCOUNTING_PREP_INPUT", timeoutMilliseconds = 250},
  {operation = "PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE", timeoutMilliseconds = 250},
  {operation = "PCM_OP_TRANS_ABORT", timeoutMilliseconds = 250},
  {operation = "PCM_OP_TRANS_OPEN", timeoutMilliseconds = 250},
  {operation = "PCM_OP_WRITE_FLDS", timeoutMilliseconds = 250}
]
```

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 seconds that is equal to the value specified by the `contextOpenTimeout` 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:

```c
ConnectionManager = {
  database = 1
  service = "service/pcm_client"
}
```
Configuring bcdBillingClient, Continued

Configuring BRM connections (continued)

```java
cmPointers {
  poolSize = 250
  "ip 192.168.111.111 12010"
}
{
  poolSize = 500
  "ip 192.168.111.112 12010"
}
{
  poolSize = 250
  "ip 192.168.111.111 12011"
}
{
  poolSize = 500
  "ip 192.168.111.112 12011"
}
}
```

The **database** and **service** values should not be changed. NCC passes these values to the BRM Connection Manager when they connect.

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.

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

contextOpenTimeout

**Syntax:** See Example

**Description:** Specifies the length of time in seconds 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).

**Example:**

```
contextOpenTimeout =
```

Continued on next page
Configuring additional bcdBillingClient parameters (continued)

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.

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
Configuring bcdBillingClient, Continued

Configuring additional bcdBillingClient parameters (continued)

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 PCP messages before checking for SLEE events. This guide recommends that you leave setting at the default value of 50. Increasing the value would improve latency but 50 ms is not that high.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: See Description
Default: 50
Example: maxSelectMicroseconds = 50

Continued on next page
Configuring additional `bcdBillingClient` parameters (continued)

**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 or false  
Default: false  
Example: `recordCMIPAddressInStats = false`

**recordOpcodeInStats**

Syntax: See Example  
Description: Specifies whether to include opcodes in statistics.  
Type: Boolean  
Optionality: Optional (default used if not set).  
Allowed: true or 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 or false  
Default: false  
Example: `recordPortInStats = false`

Continued on next page
Configuring bcdBillingClient, Continued

Configuring additional bcdBillingClient parameters (continued)

recoverCmPtrSeconds

<table>
<thead>
<tr>
<th>Syntax:</th>
<th>See Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>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.</td>
</tr>
<tr>
<td>Type:</td>
<td>Integer</td>
</tr>
<tr>
<td>Optionality:</td>
<td>Optional (default used if not set).</td>
</tr>
<tr>
<td>Allowed:</td>
<td>Any positive integer</td>
</tr>
<tr>
<td>Default:</td>
<td>5</td>
</tr>
<tr>
<td>Example:</td>
<td>recoverCmPtrSeconds = 5</td>
</tr>
</tbody>
</table>
Configuring calls, events, and vouchers

Typically, voice calls are controlled in NCC 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:

```java
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 NCC Sessions to BRM Services.

Typically, data calls are controlled in NCC 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:

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

Continued on next page
Configuring calls, events, and vouchers, Continued

Configuring NCC for data calls (continued)

For descriptions of the ServiceProfileTagMapping parameters in the eserv.config file, see Mapping NCC Sessions to BRM Services.

Configuring NCC for named events

Examples of named events include SMS, MMS, and email.

Named events are controlled by the Billable Event feature node in the NCC control plan. To charge a BRM account, do one of the following:

- Select the BRM domain (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

Configuring NCC for named event reservations and reservable direct named events

You make named event reservations using a Billable Event feature node with Billable Event Feature Selection set to Reserve, followed by another node set to Confirm or Revoke.

To create a direct named event, set Billable Event Feature Selection to Direct Event. Set the Number of Events field to a positive value.

You must handle named event refunds separately. For information on configuring named event refunds, see Configuring Named Event Refunds.

Configuring NCC for BRM vouchers

The Voucher Recharge feature node in the NCC Control Plan Editor typically controls voucher refunds. You can refund BRM accounts using either BRM or NCC vouchers.

To use BRM vouchers, the Set Active Domain feature node must have both Voucher and Wallet enabled for the BCD domain. For more information, see Creating a BCD domain (on page 19).

If you require EDRs (event detail records) to be sent to BRM when an attempt to redeem a voucher fails see Tracking BRM Voucher Redemptions with Bad PINs.

Tracking BRM voucher redemptions with bad PINs

Each time a subscriber enters a bad PIN when trying to redeem a voucher on BRM, NCC can create a new event detail record (EDR) on BRM by sending a PCM_OP_ACTIVITY operation to track possible fraud.

To enable sending of these EDRs to BRM, set the bcdActionHandler.BrmEdrObjectType parameter in the NCC eserv.config file to match the subclass name of the /event/activity class. For example, if you will be sending EDRs to BRM when a subscriber attempts to redeem a voucher using a bad PIN, set the parameter to /voucher. Likewise, set BrmBadPinResourceId to the BRM resource ID that is used to count bad PIN attempts (the default ID is 1000011) and set BrmBadPinEdrActive to true. The following example illustrates these settings:

```java
bcdActionHandler = {
...  
    BrmBadPinResourceId = 1000011
    BrmEdrObjectType = "/voucher"
...  
}
```

Continued on next page
Configuring calls, events, and vouchers, Continued

Tracking BRM voucher redemptions with bad PINs (continued)

```csharp
BrmBadPinEdrActive = True
...
}
```

For information on creating BRM vouchers, see Creating BRM Vouchers.
Modifying the BCD Client startup script

When you install the BRM Charging Driver packages, the BCD Client startup script (bcdBeClient.sh) is added in the following location:

```
/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 bcdBeClientn.sh file to specify the correct names of the startup script and its log file. For example, if you change the name of the startup 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:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Login to the SMS server as acs_oper.</td>
</tr>
<tr>
<td>2</td>
<td>Change directories to: /IN/service_packages/BCD/bin</td>
</tr>
<tr>
<td>3</td>
<td>Copy the existing startup script and give the copy a new name. For example, bcdBeClient2.sh.</td>
</tr>
</tbody>
</table>
| 4    | Using a text editor such as vi or vim, open the new startup 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 startup script and its corresponding log file to match the name you gave to the new startup 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 startup script that you want to create. |

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

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
```

This line tells the SLEE to run the BCD Client startup script. If you create multiple startup 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
```

Continued on next page
Modifying the BCD Client startup script, Continued

Configuring the SLEE to run the scripts (continued)

```
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 startup script.

**Note:** If you create multiple startup 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 startup 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

You must create balance types to match the names of the BRM resource IDs for all the resources that are used by both NCC and BRM.

NCC has two concepts with regard to account balances: currency and balance type. For example, Euros would be a currency and General Cash might be a balance type. BRM, however, uses only one concept, which is a resource ID. For example, let's say that on BRM we've made the number 978 the resource ID for Euros, while 1000076 is the resource ID for a balance type of free time.

When NCC queries BRM for a balance, it expects both the currency and the balance type.

Note: On NCC a balance of $50 means the subscriber has $50 to spend and a balance of -$10 means the subscriber owes $10. On BRM, a balance of -$50 means the subscriber has $50 to spend and a balance of $10 means the subscriber owes $10.

The BRM Charging Driver changes the sign when it presents balances to NCC users. For example, a BRM balance of -$50 will be shown in a text message as $50. However, the balance that appears in an flist for an operation is shown as it exists on BRM because the BRM Charging Driver calls a BRM print function to display it.

The BRM Charging Driver determines the currency by looking up the NCC currency code in the SMS UI. You can access the Currency Code tab in the Service Management window by logging in to SMS and selecting Prepaid Charging from the Services menu, and then Service Management to display the following window:

![Currency Code Tab in Service Management Window](image)

Continued on next page
Creating balance type mappings, Continued

About Creating Balance Type Mappings (continued)

NCC then obtains the BRM resource ID for that currency by looking up the currency code in the `BrmToNccCurrencyMapping` array in the `eserv.config` file, a sample of which is given here, to:

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

For the balance type, NCC relies on data that has been entered through the SMS Wallet Management screens, which you can access by selecting Prepaid Charging from the Services menu, and then Wallet Management. On the Balance Types tab, you must define each NCC balance type and each BRM resource ID. The following figure illustrates these entries:

![Wallet Management screenshot](image)

The relevant NCC balance types shown are Free Time and General Cash while the relevant BRM resource IDs in this example are 978 and 1000076. Note that for BRM, you specify the resource ID as the balance type name.

Continued on next page
Creating balance type mappings, Continued

About Creating Balance Type Mappings (continued)

To create balance types, login to the Service Management System, select Prepaid Charging from the Services menu, and then select Wallet Management. For more information on creating balance types, see the Charging Control Services User’s Guide.

Next, you must create balance type mappings to associate the NCC balance types with the BRM resource IDs and specify any scaling that's required between the two types.

Create balance type mappings in the Service Management window of the SMS UI. To access the Service Management window, login to the Service Management System, select Prepaid Charging from the Services menu, and select the Service Management menu item and then the Balance Type Mapping tab on the Service Management screen. The following figure illustrates the balance type mapping entries.

<table>
<thead>
<tr>
<th>NCC Balance Type</th>
<th>BRM Balance Type</th>
<th>BRM Balance Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free Time</td>
<td>1000076</td>
<td>6000</td>
</tr>
<tr>
<td>General Cash</td>
<td>978</td>
<td>100</td>
</tr>
</tbody>
</table>

The first line maps the NCC balance type Free Time to the BRM 1000076 resource ID, which is called Free Time in BRM. All time balances in NCC are in hundredths of seconds while BRM minutes are actually in minutes. Because there are 6000 hundredths of a second in a minute, the scaling factor is 6000.

The second line maps the NCC General Cash balance type to the BRM resource ID 978, which is Euros. NCC balances are always in small units, which in this case is Euro cents, so the scaling factor is 100.

The third and fourth lines indicate that Free Time and General Cash are balance types that can exist on a third party billing engine.

The BRM resource IDs for currency codes are defined in the BRM Pricing Center’s Resource Editor.

For the specific steps to create balance type mappings in the SMS UI, see NCC’s Charging Control Services User’s Guide.
Chapter 4

Configuring BRM for the BRM Charging Driver

About configuring BRM

This chapter describes the BRM configuration tasks that you must perform to integrate BRM with NCC.

To integrate BRM with NCC, 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 NCC. 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 NCC, 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 ................................................................. 50
- Adding storable classes .............................................................. 53
- Creating header and library files for the custom classes .................. 54
- Generating the custom JAR file .................................................... 55
- Modifying the BRM configuration files .......................................... 56
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 NCC 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 NCC information items and the corresponding BRM custom fields by adding the appropriate values to the bcdActionHandler.NccToBrmFieldMapping list in the NCC eserv.config file. For more information on creating the mapping from NCC to BRM fields, see Mapping NCC Information to BRM Fields.

You must 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 NCC fields in when NCC sends operations to BRM. The PIN_FLD_NCC_FIELD is required because the PIN_FLD_NCC_INFO container must not be empty when NCC 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 NCC, so you cannot assume that ID 10000 will be free.

For examples of NCC operations (opcodes) that include the custom fields you define, see the sections containing messages in Usage Scenarios.

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: NCC substruct containing custom fields
- Default Field ID: 10000

Note: You must set bcdActionHandler.NccInfoFieldNumber in the NCC 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.
Adding custom fields, Continued

About adding custom fields (continued)

<table>
<thead>
<tr>
<th>PIN_FLD_NCC_FIELD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Default Field ID</td>
</tr>
<tr>
<td>Description</td>
</tr>
</tbody>
</table>

**Note:** You must set the corresponding bcdActionHandler.NccInfoFieldDummyEntry value in the NCC 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.

<table>
<thead>
<tr>
<th>PIN_FLD_LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Type</td>
</tr>
<tr>
<td>Default Field ID</td>
</tr>
</tbody>
</table>

GSM networks pass NCC 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 NCC 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>PIN_FLD_LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>PIN_FLDT_STR</td>
</tr>
<tr>
<td>Default Field ID</td>
<td>1251</td>
</tr>
</tbody>
</table>

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 NCC and BRM fields by adding the appropriate values to the bcdActionHandler.NccToBrmFieldMapping list in the NCC eserv.config file. Each entry should contain the following parameters:

- NCCItem - The text name of the NCC 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.
Adding custom fields, Continued

Creating additional custom fields (continued)

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

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 NCC 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 NCC, 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 NCC subclass for each of the following classes:

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

The following example illustrates the appropriate subclasses:

/event/activity/telco/gsm
/active_session/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've 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 the 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 NCC custom fields to them.
Creating header and library files for the custom classes

Follow these steps to create the BRM header and library files required to make your NCC custom fields available to BRM application screens.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Use the BRM Developer Center to generate a header file, for example <em>ncc_flds.h</em>, for the custom NCC fields that you added.</td>
</tr>
<tr>
<td>2</td>
<td>Copy the header file to the appropriate location.</td>
</tr>
<tr>
<td>3</td>
<td>Create a library file from the header file.</td>
</tr>
<tr>
<td>4</td>
<td>Copy the library file to the appropriate location and make it available to your applications.</td>
</tr>
<tr>
<td>5</td>
<td>Restart processes.</td>
</tr>
</tbody>
</table>

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

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 the 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 the 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 the BRM Developer's Guide. These are the general steps:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Compile the .java files.</td>
</tr>
<tr>
<td>2</td>
<td>Move the compiled .class files to a directory.</td>
</tr>
<tr>
<td>3</td>
<td>Create a Java archive (.jar) file.</td>
</tr>
<tr>
<td>4</td>
<td>Copy the .jar file into a directory that client applications can access</td>
</tr>
<tr>
<td>5</td>
<td>Edit the Developer Center startup script, which is platform specific, to include the custom .jar file.</td>
</tr>
<tr>
<td>6</td>
<td>Add the new custom fields generated by the Developer Center into the Infranet.properties file of the relevant BRM client applications. Infranet.properties is a configuration file used by BRM client applications like Pricing Center, Developer Center and Customer Center. It must be changed on every computer on which the client applications run.</td>
</tr>
</tbody>
</table>
Modifying the BRM configuration files

To integrate BRM with NCC, 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 NCC.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | If they are not already present, add the BRM events that you will use with NCC 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:  
| 2    | Load the `pin_event_map` file, following instructions for the `load_pin_event` utility in *BRM Setting Up Pricing and Rating* |
| 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/session/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*. |

Continued on next page
Modifying the BRM configuration files, Continued

Steps to modify BRM configuration files (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>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 NCC eserv.config file on the SLC. For more information on the scaling factor, see Mapping NCC Sessions to BRM Services. For example, modify the default pin_config_reservation_aaa_prefs_gsm_data file to match the scaling factor:</td>
</tr>
<tr>
<td></td>
<td>0 PIN_FLD_RESERVATION_INFO ARRAY [0]</td>
</tr>
<tr>
<td></td>
<td>1 PIN_FLD_QUANTITY DECIMAL [0] &lt;50 x scaling factor&gt;</td>
</tr>
<tr>
<td></td>
<td>1 PIN_FLD_MIN_QUANTITY DECIMAL [0] 0</td>
</tr>
<tr>
<td></td>
<td>1 PIN_FLD_INCREMENT QUANTITY DECIMAL [0] &lt;50 x scaling factor&gt;</td>
</tr>
<tr>
<td></td>
<td>1 PIN_FLD_RUM_NAME STR [0] &quot;Size&quot;</td>
</tr>
<tr>
<td></td>
<td>1 PIN_FLD_REQ_MODE ENUM [0] 4</td>
</tr>
<tr>
<td></td>
<td>1 PIN_FLD_UNIT ENUM [0] 0</td>
</tr>
<tr>
<td></td>
<td>0 PIN_FLD_RESERVATION_INFO ARRAY [1]</td>
</tr>
<tr>
<td></td>
<td>1 PIN_FLD_QUANTITY DECIMAL [0] 50</td>
</tr>
<tr>
<td></td>
<td>1 PIN_FLD_MIN_QUANTITY DECIMAL [0] 0</td>
</tr>
<tr>
<td></td>
<td>1 PIN_FLD_INCREMENT QUANTITY DECIMAL [0] 50</td>
</tr>
<tr>
<td></td>
<td>1 PIN_FLD_RUM_NAME STR [0] &quot;Amount&quot;</td>
</tr>
<tr>
<td></td>
<td>1 PIN_FLD_REQ_MODE ENUM [0] 1</td>
</tr>
<tr>
<td></td>
<td>1 PIN_FLD_UNIT ENUM [0] 0</td>
</tr>
<tr>
<td>6</td>
<td>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.</td>
</tr>
<tr>
<td>7</td>
<td>Restart the BRM Connection Manager.</td>
</tr>
</tbody>
</table>

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 NCC custom fields. You must also match the corresponding NCC parameters to the product values as follows:

- The NCC BrmServicePoid parameter must match the product value "Applies to".
- The NCC BrmObjectType parameter must match the product value "Event".
- The NCC 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 ............................................................60
Creating a product and deal for data calls ............................................................61
Creating a product and deal for named events ..................................................62
Creating BRM vouchers .................................................................................65
Creating a customer .........................................................................................66
Creating a 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 NCC, this service is defined in the BrmServicePoid parameter for the voice call entry in the `eserv.config` file. For example:
  \[\text{BrmServicePoid} = \text{/service/telco/gsm/telephony}\]

- **Event set to** `/event/session/telco/gsm`
  On NCC, 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 = \text{"gsm"}`.

- **Measured by:** `Duration`
  On NCC, 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

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 NCC, 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 NCC, 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 NCC, BrmReqMode parameter must be set to "VOLUME"

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

NCC uses a concept called *named event* that allows you to name events such as the sending of a short message that costs fifty cents or an event for purchasing the Friends and Family service for thirty dollars. You can define these using the SMS UI by selecting **Prepaid Charging** from the **Services** menu, then **Rating Management**, and then the **Billable Event** tab.

You can define several sets of events and several events within each set. Each set and each event have names. When you create an event you also specify an event type: Direct Event, Reserve Event, or Any. In our example, assume that the short message event is of type *Any* and the purchase of the Friends and Family service is of type Direct Event.

To use named events, use the Billable Event feature node in a control plan and select the class name, the event name, and one of the following billable event types: **Direct Event**, **Reserve Event**, **Confirm Event**, Revoke Event, or **Cost of Event**.

To charge for a short message, you can use the Billable Event feature node to send a Reserve Event message when NCC attempts to send the message and a Confirm Event when the message has been delivered. When used with the BRM Charging Driver, the Reserve Event type sends a PCM_OP_TCF_AAA_AUTHORIZE operation and the Confirm Event type sends a PCM_OP_TCF_AAA_STOP_ACCOUNTING operation.

Alternatively, if you want to charge immediately for the message, without knowing whether it will be delivered, you can choose a Direct Event billable event, which is allowed for named event types of Direct Event or Any. In the example, because the named event type is *Any*, this is a reservable direct named event. In this case, the Billable Event feature node sends a PCM_OP_TCF_AAA_AUTHORIZE operation immediately followed by a PCM_OP_TCF_AAA_STOP_ACCOUNTING operation.

To purchase the Friends and Family service, set the billable event type to **Direct Event** and select the **Friends and Family** event. Because the event type is Direct Event, the Billable Event feature node sends a PCM_OP_PURCHASE_DEAL operation.

For examples of the messages sent for direct named events, see *Messages: Direct Named Event Reservation* and *Messages: Direct Named Event for Non-Reservable Event*.
Creating a product and deal for named events, Continued

About creating products and deals for direct named events (continued)

Configuring BRM for named event reservations and reservable direct named events
On BRM, create a product with the following properties:

- **Applies to: /service/telco/gsm/sms**
  
  On NCC, the prefix of this service, which is common to all named events, is defined in the bcdActionHandler.poidPrefix parameter as "/service/telco/". The NCC billable event set, or billable class, must match the suffix of the BRM service that follows the prefix, which in this case is **gsm/sms**.

- **Event: /event/session/telco/gsm/ncc**
  
  The NCC billable event name must match the suffix of the BRM event that follows the /event/session/telco prefix. In this example, the billable event name is **gsm/ncc**. In this example, BrmObjectType = "gsm/ncc".

- **Measured by: Number of Events**
  
  On NCC, the BRMReqMode parameter must be set to "NUMBER OF EVENTS".

The rate plan should be positive.

You can rate a named event without charging the subscriber's account by using the Billable Event feature node with Billable Event Feature Selection set to Cost of Event. To support this feature, the BRM product must include the following additional event:

- **Event: /event/activity/gsm/ncc**
- **Measured by: Occurrence**.

The rate plan amount must match the amount assigned to the session event described above.

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

Continued on next page
Creating a product and deal for named events, Continued

About creating products and deals for direct named events (continued)

Configuring named event refunds
Named event refunds are named events that are controlled by a Billable Event feature node whose Number of Events field is set to a negative value.

Create a BRM product with the following properties:

- Applies to: /service/telco/gsm/fax
  
  **Note**: The service must be different than the service used by ordinary named events. You can also configure a custom class, such as /service/telco/gsm/sms/refund.

  The prefix of this service, which is common to all named events, is defined in the bcdActionHandler.poidPrefix parameter in the NCC eserv.config file. In this example, the value of poidPrefix is /service/telco.

  The NCC billable event set, or event class, must match the suffix of the BRM service following the prefix. For example, if you use the fax service, the billable event set must be gsm/fax.

- Event: /event/session/telco/gsm/ncc
- Measured by: Number of Events

  The NCC billable event name must match the suffix of the BRM event following the /event/session/telco/ prefix. In this example, the billable event name must be gsm/ncc.

  When you create the NCC billable event, set Event Type to Reservable Event, which allows only named event reservations, or Any, which allows both named event reservations and reservable direct named events.

The rate plan amount must be negative.

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

Configuring non-reservable direct named events
Non-reservable direct named events are events that are controlled by a Billable Event feature node with Billable Feature Selection set to Direct Event. When creating these events in the feature node, set Event Type to Direct Event.

Create a BRM product with the following properties:

- Applies to: Account
- Event: Purchase Fee Event

  The NCC billable event name must match the name of the BRM deal to be purchased.

- Measured by: Occurrence

Create a deal for this product, but do not add the deal to the plan in the same manner as other types of named events. This deal is purchased directly by the named event chassis action.
Creating BRM vouchers

The Voucher Recharge feature node in the NCC Control Plan Editor typically controls voucher refunds. You can refund BRM accounts using either BRM or NCC vouchers.

To use BRM vouchers, the NCC Set Active Domain feature node must have both Voucher and Wallet enabled for the BCD domain. For more information, see *Creating a BRM domain* (on page 19).

If you require EDRs (event detail records) to be sent to BRM when an attempt to redeem a voucher fails see *Tracking BRM Voucher Redemptions with Bad PINs*.

The following deal is required only for BRM vouchers.

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

- **Applies to: /service/telco/gsm/account**
  
  On NCC, this service is defined in the BrmServicePoid parameter for the voucher entry. In this case, for example:
  
  BrmServicePoid = "/service/telco/gsm/account"

- **Event set to /event/session/telco/purchase fee event**
  
  On NCC, 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 = "purchase fee event"

- **Measured by: Occurrence**
  
  On NCC, BrmReqMode parameter must be set to "OCCURRENCE".

The rate plan amount must be negative.

Create a deal for this product. Generate BRM vouchers in the BRM Voucher Administration Center using the selected deal.
Creating a customer

For any NCC 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 complete the following prerequisites on NCC:

- Create an NCC subscriber whose subscriber ID matches the number assigned to the BRM services.
- Select the BRM Charging Driver domain (for example, BCD) when you create the subscriber's wallet.

For information on creating NCC subscribers, see NCC Charging and 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:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select the plan containing the deals that you defined in Creating Products and Deals.</td>
</tr>
<tr>
<td>2</td>
<td>Assign a number and SIM to each service that is included in the plan.</td>
</tr>
</tbody>
</table>
| 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 NCC 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 ................................................................. 68
About statistics and reports

The BCD Client collects several statistics that it reports to the SMS database by using the SMS statistics collection mechanism. The BCD Client reports statistics according to the reporting times that are defined in the SMF_STATISTICS_DEFN table in the SMS screens. The BCD Client records the following statistics:

- A count of the number of SLEE events received; that is, the number of requests by the slee_acs process to send an operation to BRM.
- The number of successful operation responses, by opcode and by connection IP address, that it receives from the BRM Connection Managers.
- The number of timeouts that have occurred while waiting for a response to an operation. This statistic is reported by operation code and by connection IP address.
- The number of successful attempts to establish a connection to a BRM Connection Manager, per IP address and port pair.
- The number of unsuccessful attempts to establish a connection to a BRM Connection Manager, per IP address and port pair.
- The number of successful attempts at sending an operation to a BRM Connection Manager, per operation code and per Connection Manager.
- The number of unsuccessful attempts at sending an operation to a BRM Connection Manager, per operation code and per Connection Manager.
- The number of successful attempts at receiving an operation from a Connection Manager, per operation code and per Connection Manager.
- The number of unsuccessful attempts to receive an operation from a Connection Manager, per operation code and per connection manager.

Note: Statistics will be recorded per operation code and per connection manager only if those options are turned on in the eserv.config file. The parameters related to those options are: recordCMIPAddressInStats and recordOpcodeInStats.

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 NCC for the BRM Charging Driver. For more information, see Configuring Replication for the BRM Charging Driver Tables.

For information on the SMS statistics mechanism and on configuring replication for SMF database tables, see the Service Management System Technical Guide and the 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
```

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

By default, NCC collects all BRM Charging Driver statistics every five minutes. You can set the frequency through the Service Management System UI. Select the Operator Functions menu, then Statistics Management, and then Statistics.
About statistics and reports, Continued

To generate reports on BRM Charging Driver statistics, go to the Operator Functions menu in the NCC Service Management System 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 Network Charging and Control Service Management System User's Guide.
Chapter 7

Usage scenarios

About usage scenarios

<table>
<thead>
<tr>
<th>Introduction</th>
<th>This chapter presents some common usage scenarios that describe the interactions of key components when NCC integrates BRM into the charging and account settlement process.</th>
</tr>
</thead>
<tbody>
<tr>
<td>In this chapter</td>
<td>This chapter contains the following topics.</td>
</tr>
<tr>
<td>A voice call charged against BRM</td>
<td>.............................................................. 72</td>
</tr>
<tr>
<td>An SMS call charged against BRM</td>
<td>.............................................................. 83</td>
</tr>
<tr>
<td>Recharge using BRM vouchers</td>
<td>.............................................................. 87</td>
</tr>
<tr>
<td>Recharge using VWS vouchers</td>
<td>.............................................................. 95</td>
</tr>
<tr>
<td>Data charging</td>
<td>........................................................................ 101</td>
</tr>
<tr>
<td>Other messages</td>
<td>........................................................................ 107</td>
</tr>
</tbody>
</table>
A voice call charged against BRM

The scenarios in this section describe a voice call that is charged against BRM. These scenarios assume the following preconditions:

- The subscriber's MSISDN is provisioned on the NCC system.
- The subscriber is provisioned to have voice calls run a control plan with a UATB feature node.
- The subscriber's MSISDN is provisioned on BRM with prepaid GSM (voice call) service enabled.
- The subscriber has an account balance that is sufficient for 10 minutes of talk time.
- The configured grace period, specified by the `eserv.config` parameter `CCS.ccsMacroNodes.BFTGracePeriodLength` is set to 30 seconds.
- The NCC `eserv.config` parameter `bcdActionHandler.lowCreditBufferTime` is set to 30 seconds.
- BRM has been configured to charge different rates for different GSM locations.
- The appropriate entry for a voice call has been created in the `ServiceProfileTagMapping` array in the `bcdActionHandler` structure in the BCD section of the `eserv.config` file. For more information, see Configuring NCC for Voice Calls.
- A product and deal for voice calls has been set up. For more information, see Creating a Product and Deal for Voice Calls.

Continued on next page
A voice call charged against BRM, Continued

Here is an example message flow covering the caller hangs up scenario.

Caller hangs up flow

<table>
<thead>
<tr>
<th>Action 1</th>
<th>Action 2</th>
<th>Action 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>InitialDP</td>
<td>PCM_OP_TCF_AAA_AUTHORIZE</td>
<td>PCM_OP_TCF_AAA_AUTHORIZE response (5 minutes)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>RequestReportBCSMEvent ApplyCharging(4.5 minutes) Connect</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caller talks for 4.5 minutes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action 6</th>
<th>Action 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE(4.5 minutes)</td>
<td>PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE response (5 more minutes)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ApplyCharging (5 minutes)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caller hangs up after 2 minutes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCM_OP_TCF_AAA_STOP_AC COUNTING (2 minutes)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCM_OP_TCF_AAA_STOP_ACCOUNTING response</td>
</tr>
</tbody>
</table>

Continued on next page
A voice call charged against BRM, Continued

This scenario describes the actions that are taken when a subscriber makes a voice call on a CAMEL network. The voice call is handled by NCC and charged against BRM. In this sequence, the call is terminated when the caller hangs up.

See Preconditions for Voice Call Scenarios for information about the preconditions for this scenario.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1      | - The subscriber, Party A, 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 looks up the subscriber and wallet in the SLC database 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 a Universal Attempt Terminate with Billing (UATB) feature node in the control plan and invokes an InitialTimeReservation action on the BCD actions library.  
        | - The BCD actions library creates an event (BcdSleeEvent) to invoke a PCM_OP_TCF_AAA_AUTHORIZE operation and sends the event to the BCD Client that is the least busy.  
        | - The BCD Client finds a free connection on the BRM Connection Manager that has the lowest proportion of its connections currently in use.  
        | - The BCD Client invokes the PCM_OP_TCF_AAA_AUTHORIZE opcode and starts a timer for the configured value for this type of operation. |
| 3      | - BRM responds to the operation and authorizes 5 minutes of time to be used.  
        | - The BCD Client receives the BRM output flist from PCM_OP_TCF_AAA_AUTHORIZE and packages it in a BcdSleeEvent and sends it to slee_acs. It also marks the BRM connection as available and cancels the operation timer.  
        | - The BCD actions library takes the output flist and translates it into the response to the InitialTimerReservation action. |
| 4      | - The slee_acs process sends a RequestReportBCSMEvent operation, a Connect or a Continue operation, and an ApplyCharging operation. The ApplyCharging operation allows 4.5 minutes of call time (5 minutes minus the 30 second buffer previously defined).  
        | - The MSC connects the call to party B.  
        | - Party B answers the call. |
| 5      | After 4.5 minutes, the MSC sends ApplyChargingReport(callActive=true, 4.5 minutes) to slee_acs. |

Continued on next page
A voice call charged against BRM, Continued

Caller hangs up scenario (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
</table>
| 6      | The slee_acs invokes an ExtendTimeReservation action on the BCD actions library.  
   The BCD actions library constructs an event (BcdSleeEvent) and invokes a PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE opcode and sends the event to the least busy BCD Client.  
   The BCD Client chooses the most recently used BRM connection.  
   The BCD Client sends the PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE opcode and sets the timer to the configured value for this type of operation.  
| 7      | BRM responds by indicating that the caller can use 5 more minutes.  
   The BCD Client sends the output flist from the operation to slee_acs in an event (BcdSleeEvent). It also marks the BRM connection as available and cancels the operation timer.  
   The BCD actions library translates the output flist to the response of the ExtendTimeReservation action.  
| 8      | The slee_acs sends an ApplyCharging(5 minutes) operation to the MSC.  
   After 2 more minutes, party A hangs up.  
| 9      | MSC sends an ApplyChargingReport(callActive=false, 6.5 minutes) and an EventReportBCSM(Disconnect) to slee_acs.  
| 10     | The slee_acs invokes a ConfirmTimeReservation action on the BCD actions library.  
   The BCD actions library constructs an event (BcdSleeEvent) and invokes a PCM_OP_TCF_AAA_STOP_ACCOUNTING operation to indicate that 6.5 minutes of total talk time were used and sends it to the least busy Billing Client.  
   The BCD Client chooses the most recently used BRM connection.  
   The BCD Client calls the PCM_OP_SEND() function to invoke a PCM_OP_TCF_AAA_STOP_ACCOUNTING operation and sets a timer to the configured value for this type of operation.  
| 11     | BRM responds by indicating that 6.5 minutes of talk time has been deducted from the account.  
   The BCD Client sends the output flist from the operation to slee_acs packaged in an event (BcdSleeEvent). It also marks the BRM connection as available and cancels the operation timer.  
   The BCD actions library translates the output flist into the response of the ConfirmTimeReservation action. It also stores the charge information in a call context for possible later use.  
| 12     | The slee_acs proceeds with the control plan. If there are no more feature nodes to run, processing ends for this call.  

Continued on next page
A voice call charged against BRM, Continued

The following messages include operations sent to BRM and results returned by BRM for a voice call that is completed when the caller hangs up. The general message format is: nesting level (0; 1, or 2); field; data type; value.

**Operation:** send PCM_OP_TCF_AAA_AUTHORIZE (4002)

Flags = 0

<table>
<thead>
<tr>
<th>Field</th>
<th>Nested level</th>
<th>Data type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 PIN_FLD_POID</td>
<td>0</td>
<td>POID</td>
<td>/service/telco/gsm/telephony -1 0</td>
</tr>
<tr>
<td>0 PIN_FLD_PROGRAM_NAME</td>
<td>0</td>
<td>STR</td>
<td>&quot;NCC_BCD_Client&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_AUTHORIZATION_ID</td>
<td>0</td>
<td>STR</td>
<td>&quot;brmClient-cleejoh_session_6001_0&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_OBJ_TYPE</td>
<td>0</td>
<td>STR</td>
<td>&quot;gsm/ncc&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_CALLING_NUMBER</td>
<td>0</td>
<td>STR</td>
<td>&quot;004085752158&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_CALLED_NUMBER</td>
<td>0</td>
<td>STR</td>
<td>&quot;55587390000&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_EXTENDED_INFO</td>
<td>0</td>
<td>SUBSTRUCT</td>
<td>allocated 20, used 2</td>
</tr>
<tr>
<td>1 PIN_FLD_GSM_INFO</td>
<td>0</td>
<td>SUBSTRUCT</td>
<td>allocated 20, used 3</td>
</tr>
<tr>
<td>2 PIN_FLD_DIRECTION</td>
<td>0</td>
<td>ENUM</td>
<td>0</td>
</tr>
<tr>
<td>2 PIN_FLD_CELL_ID</td>
<td>0</td>
<td>STR</td>
<td>&quot;000c&quot;</td>
</tr>
<tr>
<td>2 PIN_FLD_LOC_AREA_CODE</td>
<td>0</td>
<td>STR</td>
<td>&quot;064001000f&quot;</td>
</tr>
<tr>
<td>1 10000</td>
<td>0</td>
<td>SUBSTRUCT</td>
<td>allocated 20, used 2</td>
</tr>
<tr>
<td>2 10001</td>
<td>0</td>
<td>STR</td>
<td>&quot;Present&quot;</td>
</tr>
<tr>
<td>2 PIN_FLD_LOCATION</td>
<td>0</td>
<td>STR</td>
<td>&quot;7390002&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_MSID</td>
<td>0</td>
<td>STR</td>
<td>&quot;004085752158&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_REQ_MODE</td>
<td>0</td>
<td>ENUM</td>
<td>2</td>
</tr>
<tr>
<td>0 PIN_FLD_RESULT</td>
<td>0</td>
<td>ENUM</td>
<td>1</td>
</tr>
<tr>
<td>0 PIN_FLD_RATING_STATUS</td>
<td>0</td>
<td>ENUM</td>
<td>0</td>
</tr>
<tr>
<td>0 PIN_FLD_AUTHORIZATION_ID</td>
<td>0</td>
<td>STR</td>
<td>&quot;brmClient-cleejoh_session_6001_0&quot;</td>
</tr>
</tbody>
</table>

**Note:** REQ_MODE 2 = DURATION according to the following definitions:

- PIN_TCF_AAA_REQ_MODE_AMOUNT = 1
- PIN_TCF_AAA_REQ_MODE_DURATION = 2
- PIN_TCF_AAA_REQ_MODE_VOLUME = 4
- PIN_TCF_AAA_REQ_MODE_ACTIVITY = 8

**Result:** received for operation PCM_OP_TCF_AAA_AUTHORIZE (4002)

<table>
<thead>
<tr>
<th>Field</th>
<th>Nested level</th>
<th>Data type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 PIN_FLD_POID</td>
<td>0</td>
<td>POID</td>
<td>/active_session/telco/gsm/ncc 134439 0</td>
</tr>
<tr>
<td>0 PIN_FLD_EXPIRATION_T</td>
<td>0</td>
<td>TSTAMP</td>
<td>(1331825648) Thu Mar 15 15:34:08 2012</td>
</tr>
<tr>
<td>0 PIN_FLD_QUANTITY</td>
<td>0</td>
<td>DECIMAL</td>
<td>50</td>
</tr>
<tr>
<td>0 PIN_FLD_SERVICE_OBJ</td>
<td>0</td>
<td>POID</td>
<td>/reservation 131879 0</td>
</tr>
<tr>
<td>0 PIN_FLD_BALGRP_OBJ</td>
<td>0</td>
<td>POID</td>
<td>/balance_group 136803 98</td>
</tr>
<tr>
<td>1 PIN_FLD_AMOUNT</td>
<td>0</td>
<td>DECIMAL</td>
<td>0.50</td>
</tr>
<tr>
<td>1 PIN_FLD_AVAILABLE_RESOURCE_LIMIT</td>
<td>0</td>
<td>DECIMAL</td>
<td>9.500</td>
</tr>
<tr>
<td>0 PIN_FLD_RESULT</td>
<td>0</td>
<td>ENUM</td>
<td>1</td>
</tr>
<tr>
<td>0 PIN_FLD_RATING_STATUS</td>
<td>0</td>
<td>ENUM</td>
<td>0</td>
</tr>
<tr>
<td>0 PIN_FLD_AUTHORIZATION_ID</td>
<td>0</td>
<td>STR</td>
<td>&quot;brmClient-cleejoh_session_6001_0&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_ACCOUNT_OBJ</td>
<td>0</td>
<td>POID</td>
<td>/account 135267 0</td>
</tr>
</tbody>
</table>

**Note:** PIN_FLD_QUANTITY = 50. This means 50 seconds.

We are charging one Euro cent per second and, on BRM, a positive balance means that the subscriber owes money.

This is a limited credit account with the credit limit set to 10 Euros.

**Operation:** Send PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE (4026)

Flags = 0

<table>
<thead>
<tr>
<th>Field</th>
<th>Nested level</th>
<th>Data type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 PIN_FLD_POID</td>
<td>0</td>
<td>POID</td>
<td>/service/telco/gsm/telephony -1 0</td>
</tr>
<tr>
<td>0 PIN_FLD_PROGRAM_NAME</td>
<td>0</td>
<td>STR</td>
<td>&quot;NCC_BCD_Client&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_AUTHORIZATION_ID</td>
<td>0</td>
<td>STR</td>
<td>&quot;brmClient-cleejoh_session_6001_0&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_OBJ_TYPE</td>
<td>0</td>
<td>STR</td>
<td>&quot;gsm/ncc&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_CALLING_NUMBER</td>
<td>0</td>
<td>STR</td>
<td>&quot;004085752158&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_CALLED_NUMBER</td>
<td>0</td>
<td>STR</td>
<td>&quot;55587390000&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_EXTENDED_INFO</td>
<td>0</td>
<td>SUBSTRUCT</td>
<td>allocated 20, used 2</td>
</tr>
<tr>
<td>1 PIN_FLD_GSM_INFO</td>
<td>0</td>
<td>SUBSTRUCT</td>
<td>allocated 20, used 3</td>
</tr>
<tr>
<td>2 PIN_FLD_DIRECTION</td>
<td>0</td>
<td>ENUM</td>
<td>0</td>
</tr>
</tbody>
</table>

Continued on next page
A voice call charged against BRM, Continued

Messages: caller hangs up (continued)

2 PIN_FLD_CELL_ID STR [0] "000c"
2 PIN_FLD_LOC_AREA_CODE STR [0] "064001000f"
1 1000 SUBSTRUCT [0] allocated 20, used 2
2 1001 STR [0] "Present"
2 PIN_FLD_LOCATION STR [0] "7390002"
0 PIN_FLD_MSID STR {0} "04085752158"
0 PIN_FLD_REQ_MODE ENUM [0] 2
0 PIN_FLD_QUANTITY DECIMAL [0] 45.000000000000000

Note: PIN_FLD_QUANTITY=45.0 means the subscriber has used 45 seconds.

Field ID 10000 is NCC_INFO and 10001 is NCC_FIELD. The names do not come out because they are not standard BRM fields. On a production system, the IDs may be different because 10000 and 10001 may already be used.

Result: received for operation

PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE (4026)

0 PIN_FLD_POID POID [0] 0.0.0.1 /active_session/telco/gsm/ncc
134439 1
0 PIN_FLD_EXPIRATION_T TSTAMP [0] (1331826564) Thu Mar 15 15:34:14 2012
0 PIN_FLD_QUANTITY DECIMAL [0] 50
0 PIN_FLD_RESERVED_OBJ POID [0] 0.0.0.1 /reservation 131879 0
0 PIN_FLD_BAL_GRP_OBJ POID [0] 0.0.0.1 /balance_group 136803 99
0 PIN_FLD_BALANCES ARRAY [978] allocated 20, used 2
1 PIN_FLD_AMOUNT DECIMAL [0] 0.500
1 PIN_FLD_AVAIL_RESOURCE_LIMIT DECIMAL [0] 9.000
0 PIN_FLD_RESULT ENUM [0] 1
0 PIN_FLD_RATING_STATUS ENUM [0] 0
0 PIN_FLD_AUTHORIZATION_ID STR [0] "brmClient-cleejoh_session_6001_0"

Operation: Send PCM_OP_TCF_AAA_STOP_ACCOUNTING (4007)

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_AUTHORIZATION_ID STR [0] "brmClient-cleejoh_session_6001_0"
0 PIN_FLD_OBJ_TYPE STR [0] "gsm/ncc"
0 PIN_FLD_CALLING_NUMBER STR [0] "004085752158"
0 PIN_FLD_CALLED_NUMBER STR [0] "55587390000"
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] "7390002"
2 10007 STR [0] "55587390000"
0 PIN_FLD_MSID STR {0} "04085752158"
0 PIN_FLD_REQ_MODE ENUM [0] 2
0 PIN_FLD_QUANTITY DECIMAL [0] 18.000000000000000

Note: The subscriber has used 18 more seconds and that's the end of the call.

Times reported to BRM are not cumulative.

Continued on next page
Here is an example message flow covering the funds expire scenario.
This scenario describes the actions that are taken when a voice call is made by a subscriber on a CAMEL network. The voice call is handled by NCC and charged against BRM. In this sequence, the call terminates when funds expire.

See Preconditions for Voice Call Scenarios for information about the preconditions for this scenario.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 7</td>
<td>See Caller hangs up scenario (on page 74).</td>
</tr>
<tr>
<td>8</td>
<td>Slee_acs sends an ApplyCharging(5 minutes) operation to the MSC.</td>
</tr>
<tr>
<td>9</td>
<td>After 5 minutes, the MSC sends ApplyChargingReport(callActive=true,9.5 minutes) to slee_acs.</td>
</tr>
<tr>
<td>10</td>
<td>The slee_acs invokes an ExtendTimeReservation action on the BCD actions library.</td>
</tr>
<tr>
<td></td>
<td>The BCD actions library constructs an event (BcdSleeEvent) to invoke a PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE operation and sends it to the least busy BCD Client.</td>
</tr>
<tr>
<td></td>
<td>The BCD Client chooses the most recently used BRM connection.</td>
</tr>
<tr>
<td></td>
<td>The BCD Client calls the PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE opcode and sets a timer to the configured value for this type of operation.</td>
</tr>
<tr>
<td>11</td>
<td>BRM determines that there are no funds left on this account and sends a response indicating that no more time can be used.</td>
</tr>
<tr>
<td></td>
<td>The BCD Client calls 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.</td>
</tr>
<tr>
<td></td>
<td>The BCD actions library translates the output flist to the failure response of the ExtendTimeReservation action.</td>
</tr>
<tr>
<td>12</td>
<td>The slee_acs process sends to the MSC an ApplyCharging operation for 30 seconds (held from the original buffer), with both tone and release-if-duration-exceeded set.</td>
</tr>
<tr>
<td></td>
<td>MSC plays a tone to the caller.</td>
</tr>
<tr>
<td></td>
<td>30 seconds elapse.</td>
</tr>
<tr>
<td>13</td>
<td>MSC disconnects the call and sends TC_END(ApplyChargingReport(callActive=false, 10 minutes)) to slee_acs.</td>
</tr>
<tr>
<td>14</td>
<td>The slee_acs invokes a ConfirmTimeReservation action on the BCD actions library.</td>
</tr>
<tr>
<td></td>
<td>The BCD actions library constructs an event (BcdSleeEvent) for a PCM_OP_TCF_AAA_STOP_ACCOUNTING operation, indicating that 10 minutes of total talk time have been used, and sends it to the least busy BCD Client.</td>
</tr>
<tr>
<td></td>
<td>The BCD Client chooses the most recently used BRM connection.</td>
</tr>
<tr>
<td></td>
<td>The BCD Client calls the PCM_OP_TCF_AAA_STOP_ACCOUNTING opcode, and sets a timer to the configured value for this type of operation.</td>
</tr>
</tbody>
</table>

Continued on next page
A voice call charged against BRM, Continued

A Voice Call When Funds Expire (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
</table>
| 15     | • BRM responds to the operation, indicating that 10 minutes of talk time has been deducted from the caller's account.  
        • The BCD Client 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.  
        • The BCD actions library translates the output flist to the response of the ConfirmTimeReservation action. It also stores the charge information in call context for possible later use. |
| 16     | The slee_acs proceeds with the control plan. If there are no more feature nodes, processing for the call ends. |

Messages: funds expire

The following messages include operations sent to BRM and results returned by BRM for a voice call that is terminated when the caller's funds expire. The general message format is: nesting level (0, 1, or 2); field; data type; value.

Result received for operation

PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE (4026)

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN_FLD_POID</td>
<td>POID [0] 0.0.0.1 /active_session/telco/gsm/ncc</td>
</tr>
<tr>
<td>PIN_FLD_ACTIVE_SESSION_ID</td>
<td>STR [0] &quot;brmClient-cleejoh_session_15004_0&quot;</td>
</tr>
<tr>
<td>PIN_FLD_RESULT</td>
<td>ENUM [0] 0</td>
</tr>
<tr>
<td>PIN_FLD_REASON</td>
<td>ENUM [0]</td>
</tr>
<tr>
<td>PIN_FLD_RATING_STATUS</td>
<td>ENUM [0] 26</td>
</tr>
</tbody>
</table>

Note: RATING_STATUS non-zero is bad. The following definitions apply:

typedef enum rating_result {
SUCCESSFUL_RATING = 0,
ZERO_QUANTITY = 1,
NO_SCALE_TO_RATE = 10,
NO_CANDIDATE_RUMS = 11,
NO_INITIAL_PRODUCTS = 12,
CALC_MAX_IN_MULTI_RUM = 13,
NO_MATCHING_RUM = 14,
NO_QUALIFIED_PRODUCTS = 15,
NO_RUM = 16,
STATUS_MISMATCH = 17,
PRODUCT_NOT_IN_DB = 18,
NO_PRODUCT_IN_AUDIT_DB = 19,
NO_RATE_PLAN = 20,
NO_MATCHING_SELECTOR_DATA = 21,
NO_RATING_CURRENCY = 22,
NO_VALID_RATE_SPAN = 23,
NO_VALID_RATE = 24,
NO_MATCHING_IMPACT_CATEGORY = 25,
CREDIT_LIMIT_EXCEEDED = 26
} rating_result_t;

Continued on next page
Here is an example message flow covering the subscriber not found on BRM scenario.

The scenario describes the sequence of events that occur when the caller has been provisioned on NCC but not on BRM. In this case, the operator wants to give the caller 10 minutes of talk time and produce a Billing Failure Treatment (BFT) event data record (EDR). Billing Failure Treatment refers to use of a service without real-time billing.

See Preconditions for Voice Call Scenarios for information about the preconditions for this scenario.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1      | • The subscriber attempts a 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 looks up the subscriber and wallet in the SLC database 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 a Universal Attempt Terminate with Billing feature node in the control plan and invokes an InitialTimeReservation action on the BCD actions library.  
        • The BCD actions library constructs an event (BcdSleeEvent) to invoke the PCM_OP_TCF_AAA_AUTHORIZE opcode and sends the event to the least busy BCD Client.  
        • The BCD Client finds a free connection on the Connection Manager with the least number of connections currently in use.  
        • The BCD Client calls the PCM_OP_TCF_AAA_AUTHORIZE opcode and sets a timer to the configured value for this type of operation. |
### A voice call charged against BRM, Continued

**Subscriber not found on BRM scenario (continued)**

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
</table>
| 3      | - BRM responds to the operation with an error code of 4, PIN_ERR_BAD_ARG.  
- The BCD Client sends the output flist from the operation in an event (BcdSleeEvent) to slee_acs. It also marks the BRM connection as available and cancels the timer.  |
| 4      | - The UATB feature node takes the Billing Failure Treatment (BFT) exit. This condition is regarded as a billing failure because it is a provisioning error.  
- The control plan runs an Attempt Terminate with Duration feature node that grants the caller 10 minutes of time.  
- The slee_acs process sends an ApplyCharging operation to MSC.  |
| 5      | - 10 minutes elapse.  
- The slee_acs writes a BFT event detail record and releases the call context. |
An SMS call charged against BRM

The scenarios in this section describe an SMS (short message) call from a GSM (voice) phone. These scenarios assume the following preconditions:

- The subscriber's MSISDN is provisioned on the NCC system.
- The subscriber is provisioned so that a short message runs a control plan with a Billable Event feature node (direct named event) and attempt delivery to pending node.
- The subscriber's MSISDN is provisioned on BRM with GSM service enabled.
- The subscriber's account balance is sufficient to send one SMS.
- The destination MSISDN is valid.

The scenarios in this section assume that Message Manager (XMS) have been configured to trigger ACS. For information on configuring Message Manager, see the Messaging Manager Technical Guide.

The scenarios in this section also assume that you have set up a product and deal for named events. For information on creating a product and deal for named events see Creating a Product and Deal for Named Events.

Here is an example message flow covering the SMS call delivered successfully scenario.
An SMS call charged against BRM, Continued

This scenario describes a short message (SMS) that a subscriber sends from a GSM phone. The message is handled by NCC and charged against BRM and is delivered successfully to the destination.

See Preconditions for SMS Call Scenarios for information about the preconditions for this scenario.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The subscriber sends a short message from a GSM (voice) phone. MSC sends MOForwardSM to xmsTrigger</td>
</tr>
<tr>
<td>2</td>
<td>xmsTrigger sends InitialDP to slee_acs</td>
</tr>
<tr>
<td>3</td>
<td>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 SLC database and determines the control plan to run. The slee_acs process runs the control plan. The slee_acs process reaches a Billable Event feature node specifying Direct Event and name of the event as SMS. The Billable Event feature node invokes the DirectNamedEvent action in the BCD actions library. The BCD actions library constructs an event (FlistSleeEvent) containing a PCM_OP_TCF_AAA_STOP_ACCOUNTING operation with the quantity set to 1. The BCD Client finds a free connection on the connection manager with the lowest proportion of its connections currently in use. The BCD Client calls the PCM_OP_TCF_AAA_STOP_ACCOUNTING opcode and sets a timer to the configured value for this type of operation.</td>
</tr>
<tr>
<td>4</td>
<td>BRM responds to the operation by indicating that the SMS message has been successfully charged. The BCD Client 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 timer for the operation. The BRM Charging Driver actions library translates the output flist to the response of the DirectNamedEvent action.</td>
</tr>
<tr>
<td>5</td>
<td>The control plan reaches an Attempt Delivery to Pending (ADP) feature node, which is the main MMX node for delivering short messages. The ADP feature node sends RequestReportBCSMEvent and INAP Continue operations to xmsTrigger.</td>
</tr>
<tr>
<td>6</td>
<td>xmsTrigger sends MTFowardSM to the MSC serving the called subscriber.</td>
</tr>
<tr>
<td>7</td>
<td>Destination MSC successfully delivers SMS to called subscriber and sends success result of MTFowardSM to xmsTrigger.</td>
</tr>
</tbody>
</table>

Continued on next page
An SMS call charged against BRM, Continued

SMS call delivered successfully scenario (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 8    | - xmsTrigger sends EventReportBCSM(0Disconnect) to slee_acs.  
     | - The slee_acs process reaches an end node and clears the call context. |
| 9    | xmsTrigger sends a success result of MOForwardSM to the originating MSC. |

SMS call that fails permanently scenario

Here is an example message flow covering the SMS call that fails permanently scenario.

This scenario describes a short message (SMS) that a subscriber sends from a GSM phone. The message is handled by NCC and charged against BRM but delivery fails permanently.

See Preconditions for SMS Call Scenarios for information about the preconditions for this scenario.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 5</td>
<td>See steps for SMS Call Delivered Successfully Scenario (on page 83).</td>
</tr>
<tr>
<td>6</td>
<td>xmsTrigger sends SendRoutingInfoForSMS to HLR to find the location of the destination handset.</td>
</tr>
</tbody>
</table>

Continued on next page
## An SMS call charged against BRM, Continued

### SMS call that fails permanently scenario (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>The destination number is invalid so HLR sends SendRoutingInfoForSMS error to xmsTrigger.</td>
</tr>
<tr>
<td>8</td>
<td>xmsTrigger sends EventReportBCSM(RouteSelectFailure) to the slee_acs process.</td>
</tr>
<tr>
<td>9</td>
<td>The slee_acs process sends ReleaseCall to xmsTrigger.</td>
</tr>
<tr>
<td>10</td>
<td>xmsTrigger sends a MTForwardSM error to the originating MSC.</td>
</tr>
</tbody>
</table>
| 11   | • The slee_acs process takes the Delivery Failure branch of the ADP feature node and reaches a billable event feature node.  
      • The billable event feature node is configured to send -1 event of type SMS (that is, refund an SMS). It invokes a DirectNamedEvent action on the BCD actions library.  
      • The BCD actions library constructs an FlistSleeEvent containing PCM_OP_TCF_AAA_REFUND, because the number of events is negative, with quantity set to 1.  
      • The BCD Client has a free BRM connection and calls the PCM_OP_TCF_AAA_REFUND opcode and sets the timer for the operation to the configured value for this type of operation. |
| 12   | • BRM responds to the operation indicating that the cost of the SMS has been refunded successfully.  
      • The BCD Client sends the output flist for the operation in an event (BcdSleeEvent) to the slee_acs process. It also marks the BRM connection and free and cancels the operation timer.  
      • The BCD actions library takes the output flist and translates it to the response of the DirectNamedEvent action. |
| 13   | The slee_acs process reaches an end node and clears the call context. |
Recharge using BRM vouchers

Preconditions for recharge using BRM vouchers

The scenarios in this section describe a subscriber who has a BRM voucher and uses it through the NCC platform to recharge the amount on a BRM account, either by way of IVR, or SMS. These scenarios assume the following preconditions:

- The subscriber's MSISDN is provisioned on the NCC system.
- The subscriber is provisioned so that voice calls run a control plan with a UATB feature node.
- The subscriber's MSISDN is provisioned on BRM with prepaid GSM (voice) service enabled.
- The voucher is provisioned on BRM.

For information on creating vouchers, see Creating Vouchers.

IVR BRM voucher recharge flow

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

IVR BRM voucher recharge scenario

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

Continued on next page
Recharge using BRM vouchers, Continued

IVR BRM voucher recharge scenario (continued)

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

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | - The subscriber dials the voucher recharge number, which is toll free.  
      | - MSC sends InitialDP to the slee_acs process with ServiceKey set to a special value that indicates voucher recharge. |
| 2    | - The CCS service loader looks up the subscriber and wallet in the SLC database and determines the control plan to run, using 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 slee_acs. |
| 4    | - The Voucher Recharge feature node invokes the VoucherRedeem action in the BCD actions library.  
      | - The BCD actions library constructs an event (FlistSleeEvent) that contains PCM_OP_PYMT_TOPUP and sends it to the BCD Client.  
      | - The BCD Client calls the PCM_OP_PYMT_TOPUP opcode and sets a timer to the configured value for this type of operation. |
| 5    | - BRM responds to the operation indicating that the account has been successfully recharged.  
      | - The BCD Client 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.  
      | - The BCD actions library translates the output flist to the response of the VoucherRedeem action.  
      | - The Voucher Recharge feature node sends Play Announcement to the MSC. |
| 6    | MSC plays an announcement to the caller, stating that the recharge was successful. |
| 7    | The announcement finishes and MSC sends Specialized ResourceReport to slee_acs. |

Continued on next page
Recharge using BRM vouchers, Continued

IVR BRM voucher recharge scenario (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 8    | • The control plan reaches a Play Voucher Redeem Balances feature node that invokes a VoucherInfo action on the BCD actions library.  
  • The BCD actions library returns the balance update information returned in the output flist for the PCM_OP_PYMT_TOPUP operation.  
  • The Play Voucher Redeem Balances feature node sends PlayAnnouncement to the MSC. |
| 9    | • MSC plays the balance update information to the caller in an announcement.  
  • The announcement ends and the MSC sends SpecializedResourceReport to slee_acs. |
| 10   | • The control plan reaches an end node and ACS sends DisconnectForwardConnection, ReleaseCall to the MSC and clears the call context.  
  • The caller is disconnected. |

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

**Operation:** Send PCM_OP_TCF_AAA_QUERY_BALANCE (4104)

```plaintext
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 2
    2 10001 STR [0] "Present"
    2 PIN_FLD_LOCATION STR [0] "7390002"
0 PIN_FLD_MSID STR [0] "004085752158"
0 PIN_FLD_FLAGS INT [0] 4
```

**Result:** received for operation PCM_OP_TCF_AAA_QUERY_BALANCE (4104)

```plaintext
0 PIN_FLD_POID POID [0] 0.0.0.1 /balance_group 136803 159
0 PIN_FLD_ACCOUNT_OBJ POID [0] 0.0.0.1 /account 135267 0
0 PIN_FLD_BILLINFO_OBJ POID [0] 0.0.0.1 /billinfo 138339 0
0 PIN_FLD_EFFECTIVE_T TSTAMP [0] (1331675046) Tue Mar 13 21:44:06 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] 400
  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] (1331622000) Tue Mar 13 07:00:00 2012
    2 PIN_FLD_VALID_FROM_DETAILS INT [0] 0
  2 PIN_FLD_CURRENT_BAL DECIMAL [0] 400
  2 PIN_FLD_NEXT_BAL DECIMAL [0] 0
  2 PIN_FLD_DELAYED_BAL DECIMAL [0] 0
```

Continued on next page
Recharge using BRM vouchers, Continued

Messages: redemption of BRM voucher (continued)

```
2  PIN_FLD_ROLLOVER_DATA  INT [0] 0
2  PIN_FLD_GRANTOR_OBJ    POID [0] 0.0.0.1 /purchased_product 137443
0 2  PIN_FLD_STATUS       ENUM [0] 1
2  PIN_FLD_FLAGS         INT [0] 2
1  PIN_FLD_CURRENT_TOTAL DECIMAL [0] 400
1  PIN_FLD_CREDIT_FLOOR DECIMAL [0] NULL
1  PIN_FLD_CREDIT_LIMIT DECIMAL [0] 500
1  PIN_FLD_CREDIT_THRESHOLDS INT [0] 0
1  PIN_FLD_CREDIT_THRESHOLDS_FIXED STR [0] ""
0  PIN_FLD_REALTIME_CNTR  INT [0] 2
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] -87.5
1  PIN_FLD_ROLLOVER_DATA INT [0] 0
2  PIN_FLD_CONTRIBUTOR_STR STR [0] ""
2  PIN_FLD_VALID_TO       TSTAMP [0] (1332565514) Sat Mar 24 05:05:14 2012
2  PIN_FLD_VALID_TO_DETAILS INT [0] 0
2  PIN_FLD_VALID_FROM     TSTAMP [0] (1331622000) Tue Mar 13 07:00:00 2012
2  PIN_FLD_VALID_FROM_DETAILS INT [0] 0
2  PIN_FLD_CURRENT_TOTAL DECIMAL [0] -87.5
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 PCM_OP_PYMT_TOPUP (3726)

```
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] "gsm/ncc"
0  PIN_FLD_INHERITED_INFO SUBSTRUCT [0] allocated 20, used 1
1  PIN_FLD_VOUCHERS_INFO  ARRAY [0] allocated 20, used 4
2  PIN_FLD_DEVICE_ID       STR [0] "1000400104"
2  PIN_FLD_VOUCHER_PIN     STR [0] "0971"
2  PIN_FLD_BILLINFO_OBJ    POID [0] 0.0.0.0.1 /billinfo 138339 0
2  PIN_FLD_BAL_GRP_OBJ     POID [0] 0.0.0.0.1 /balance_group 136803 159
```

Result: received for operation PCM_OP_PYMT_TOPUP (3726)

```
0  PIN_FLD_POID           POID [0] 0.0.0.1 /account 135267 0
0  PIN_FLD_RESULTS        ARRAY [0] allocated 63, used 63
1  PIN_FLD_POID           POID [0] 0.0.0.1 /event/billing/payment/voucher 271148363502715095 0
1  PIN_FLD_CREATED_T      TSTAMP [0] (1331746228) Wed Mar 14 17:30:28 2012
1  PIN_FLD_MOD_T          TSTAMP [0] (1331746228) Wed Mar 14 17:30:28 2012
1  PIN_FLD_READ_ACCESS    STR [0] "L"
1  PIN_FLD_WRITE_ACCESS   STR [0] "L"
1  PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 135267 0
1  PIN_FLD_ARCHIVE_STATUS ENUM [0] 0
1  PIN_FLD_BATCH_ID       STR [0] ""
1  PIN_FLD_CURRENCY       INT [0] 978
1  PIN_FLD_DESCR          STR [0] ""
1  PIN_FLD_EARNED_END_T   TSTAMP [0] (0) <null>
1  PIN_FLD_EARNED_START_T TSTAMP [0] (0) <null>
```

Continued on next page
Recharge using BRM vouchers, Continued

Messages: redemption of BRM voucher (continued)

Continued on next page
Recharge using BRM vouchers, Continued

Messages: redemption of BRM voucher (continued)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN_FLD_GL_ID</td>
<td>INT [0]</td>
<td>0</td>
</tr>
<tr>
<td>PIN_FLD_IMPACT_CATEGORY</td>
<td>STR [0]</td>
<td>&quot;default&quot;</td>
</tr>
<tr>
<td>PIN_FLD_IMPACT_TYPE</td>
<td>ENUM [0]</td>
<td>2</td>
</tr>
<tr>
<td>PIN_FLD_ITEM_OBJ</td>
<td>POID [0]</td>
<td>0.0.0.1 /item/payment 134999 0</td>
</tr>
<tr>
<td>PIN_FLD_LINEAGE</td>
<td>STR [0]</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>PIN_FLD_OFFERING_OBJ</td>
<td>POID [0]</td>
<td>0.0.0.0 0 0 0</td>
</tr>
<tr>
<td>PIN_FLD_PERCENT</td>
<td>DECIMAL [0]</td>
<td>1</td>
</tr>
<tr>
<td>PIN_FLD_PRODUCT_OBJ</td>
<td>POID [0]</td>
<td>0.0.0.1 /product 82729 6</td>
</tr>
<tr>
<td>PIN_FLD_QUANTITY</td>
<td>DECIMAL [0]</td>
<td>1</td>
</tr>
<tr>
<td>PIN_FLD_RATE_OBJ</td>
<td>POID [0]</td>
<td>0.0.0.1 /rate 79195 1</td>
</tr>
<tr>
<td>PIN_FLD_RATE_TAG</td>
<td>STR [0]</td>
<td>&quot;Rate 1&quot;</td>
</tr>
<tr>
<td>PIN_FLD_RESOURCE_ID</td>
<td>INT [0]</td>
<td>978</td>
</tr>
<tr>
<td>PIN_FLD_RESOURCE_ID_ORIG</td>
<td>INT [0]</td>
<td>0</td>
</tr>
<tr>
<td>PIN_FLD_RUM_ID</td>
<td>INT [0]</td>
<td>0</td>
</tr>
<tr>
<td>PIN_FLD_TAX_CODE</td>
<td>STR [0]</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>PIN_FLD_EVENT_MISC_DETAILS</td>
<td>ARRAY [0]</td>
<td>allocated 20, used 2</td>
</tr>
<tr>
<td>PIN_FLD_REASON_ID</td>
<td>INT [0]</td>
<td>0</td>
</tr>
<tr>
<td>PIN_FLD_REASON_ID_ORIG</td>
<td>INT [0]</td>
<td>0</td>
</tr>
<tr>
<td>PIN_FLD_PAYMENT</td>
<td>SUBSTRUCT [0]</td>
<td>allocated 20, used 13</td>
</tr>
<tr>
<td>PIN_FLD_ACCOUNT_NO</td>
<td>STR [0]</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>PIN_FLD_ACH</td>
<td>INT [0]</td>
<td>0</td>
</tr>
<tr>
<td>PIN_FLD_AMOUNT</td>
<td>DECIMAL [0]</td>
<td>50</td>
</tr>
<tr>
<td>PIN_FLD_AMOUNT_ORIGINAL_PAYMENT</td>
<td>DECIMAL [0]</td>
<td>0</td>
</tr>
<tr>
<td>PIN_FLD_BILL_NO</td>
<td>STR [0]</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>PIN_FLD_CHANNEL_ID</td>
<td>INT [0]</td>
<td>0</td>
</tr>
<tr>
<td>PIN_FLD_COMMAND</td>
<td>ENUM [0]</td>
<td>0</td>
</tr>
<tr>
<td>PIN_FLD_CURRENCY</td>
<td>INT [0]</td>
<td>978</td>
</tr>
<tr>
<td>PIN_FLD_MERCHANT</td>
<td>STR [0]</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>PIN_FLD_PAY_TYPE</td>
<td>ENUM [0]</td>
<td>10016</td>
</tr>
<tr>
<td>PIN_FLD_STATUS</td>
<td>ENUM [0]</td>
<td>0</td>
</tr>
<tr>
<td>PIN_FLD_SUB_TRANS_ID</td>
<td>STR [0]</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>PIN_FLD_TRANS_ID</td>
<td>STR [0]</td>
<td>&quot;T1,20,0&quot;</td>
</tr>
<tr>
<td>PIN_FLD_VOUCHERS_INFO</td>
<td>ARRAY [0]</td>
<td>allocated 20, used 5</td>
</tr>
<tr>
<td>PIN_FLD_CARD_EXPIRATION</td>
<td>INT [0]</td>
<td>0</td>
</tr>
<tr>
<td>PIN_FLD_DEVICE_ID</td>
<td>STR [0]</td>
<td>&quot;1000400104&quot;</td>
</tr>
<tr>
<td>PIN_FLD_VALID_FROM</td>
<td>TSTAMP [0]</td>
<td>(0) (null)</td>
</tr>
<tr>
<td>PIN_FLD_VALID_TO</td>
<td>TSTAMP [0]</td>
<td>(0) (null)</td>
</tr>
<tr>
<td>PIN_FLD_VOUCHER_PIN</td>
<td>STR [0]</td>
<td>&quot;0971&quot;</td>
</tr>
<tr>
<td>PIN_FLD_RESULT</td>
<td>ENUM [0]</td>
<td>1</td>
</tr>
<tr>
<td>PIN_FLD_TYPE</td>
<td>ENUM [0]</td>
<td>0</td>
</tr>
<tr>
<td>PIN_FLD_SELECT_RESULT</td>
<td>INT [0]</td>
<td>0</td>
</tr>
<tr>
<td>PIN_FLD_SELECT_STATUS</td>
<td>INT [0]</td>
<td>4</td>
</tr>
<tr>
<td>PIN_FLD_ITEM_NO</td>
<td>STR [0]</td>
<td>&quot;P1-32&quot;</td>
</tr>
<tr>
<td>PIN_FLD_VOUCHERS_INFO</td>
<td>ARRAY [0]</td>
<td>allocated 20, used 6</td>
</tr>
<tr>
<td>PIN_FLD_BILLINFO_OBJ</td>
<td>POID [0]</td>
<td>0.0.0.1 /billinfo 138339 0</td>
</tr>
<tr>
<td>PIN_FLD_DEVICE_ID</td>
<td>STR [0]</td>
<td>&quot;1000400104&quot;</td>
</tr>
<tr>
<td>PIN_FLD_VOUCHER_PIN</td>
<td>STR [0]</td>
<td>&quot;0971&quot;</td>
</tr>
<tr>
<td>PIN_FLD_EXPIRATION_T</td>
<td>TSTAMP [0]</td>
<td>(1804935600) Sat Mar 13 11:00:00 2027</td>
</tr>
<tr>
<td>PIN_FLD_VALID_TO</td>
<td>TSTAMP [0]</td>
<td>(0) (null)</td>
</tr>
<tr>
<td>PIN_FLD_VALID_FROM</td>
<td>TSTAMP [0]</td>
<td>(0) (null)</td>
</tr>
<tr>
<td>PIN_FLD_VOUCHERS_INFO</td>
<td>ARRAY [0]</td>
<td>allocated 20, used 9</td>
</tr>
<tr>
<td>PIN_FLD_AMOUNT</td>
<td>DECIMAL [0]</td>
<td>50.000</td>
</tr>
<tr>
<td>PIN_FLD_CURRENCY</td>
<td>INT [0]</td>
<td>978</td>
</tr>
<tr>
<td>PIN_FLD_VALID_TO</td>
<td>TSTAMP [0]</td>
<td>(0) (null)</td>
</tr>
<tr>
<td>PIN_FLD_VALID_FROM</td>
<td>TSTAMP [0]</td>
<td>(0) (null)</td>
</tr>
<tr>
<td>PIN_FLD_DEAL_INFO</td>
<td>SUBSTRUCT [0]</td>
<td>allocated 20, used 9</td>
</tr>
<tr>
<td>PIN_FLD_VOUCHERS_INFO</td>
<td>ARRAY [0]</td>
<td>allocated 20, used 4</td>
</tr>
<tr>
<td>PIN_FLD_BILLINFO_OBJ</td>
<td>POID [0]</td>
<td>0.0.0.1 /billinfo 138339 0</td>
</tr>
<tr>
<td>PIN_FLD_CURRENCY</td>
<td>INT [0]</td>
<td>978</td>
</tr>
<tr>
<td>PIN_FLD_VALID_TO</td>
<td>TSTAMP [0]</td>
<td>(0) (null)</td>
</tr>
<tr>
<td>PIN_FLD_VALID_FROM</td>
<td>TSTAMP [0]</td>
<td>(0) (null)</td>
</tr>
<tr>
<td>PIN_FLD_VOUCHERS_INFO</td>
<td>ARRAY [0]</td>
<td>allocated 20, used 4</td>
</tr>
<tr>
<td>PIN_FLD_AMOUNT</td>
<td>DECIMAL [0]</td>
<td>50.000</td>
</tr>
<tr>
<td>PIN_FLD_CURRENCY</td>
<td>INT [0]</td>
<td>978</td>
</tr>
</tbody>
</table>

Continued on next page
Here is an example message flow covering the SMS, BRM voucher recharge, invalid PIN scenario.

This scenario describes a subscriber with an invalid PIN who uses SMS to attempt to use a BRM voucher to recharge the amount on a BRM account.

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

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | • Subscriber sends an SMS from a GSM phone to the SMS recharge number. The SMS contains the voucher number and PIN.  
      • MSC sends MOForwardSM to xmsTrigger. |
| 2    | xmsTrigger sends InitialDP to sleev_acs with the service key set to a special value indicating SMS recharge. |
### Recharge using BRM vouchers, Continued

#### SMS BRM voucher recharge - invalid PIN scenario (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 3    | • 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 SLC database and determines that the wallet information is on 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 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 node which extracts the voucher number and PIN from temporary storage and uses them to invoke a VoucherRedeem action on the BCD actions library.  
      • The BCD actions library constructs an FlistSleeEvent containing PCM_OP_PYMT_TOPUP and sends it to the BCD Client.  
      • The BCD Client calls the PCM_OP_PYMT_TOPUP opcode and sets a timer to the configured value for this type of operation.  |
| 4    | • BRM responds to the operation indicating that the PIN was invalid.  
      • The BCD Client sends the output flist from the operation to slee_acs in an event (BcdSleeEvent). It also marks the BRM connection as available and cancels the operation timer.  
      • The BCD actions library translates the output flist to the error response of the VoucherRedeem action.  |
| 5    | • The slee_acs process takes the Voucher Invalid branch of the Voucher Recharge feature node and reaches a Send Short Message Notification node, which specifies sending a failure text message.  
      • The SSMN node constructs an MMX GenericMessage containing the failure message and sends it to xmsTrigger.  |
| 6    | • xmsTrigger sends an MTForwardSM operation to the MSC, containing the failure message.  
      • The MSC sends an SMS containing the failure message to the caller.  |
| 7    | • The slee_acs process reaches an Accept node that sends ReleaseCall to the xmsTrigger.  
      • The slee_acs reaches an end node and clears the call context.  
      • xmsTrigger sends an MTForwardSM result to the MSC.  
      • The MSC sends a notification of successful delivery of the original SMS to the caller.  |
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's MSISDN is provisioned on the NCC system.
- The subscriber is provisioned so that voice calls run a control plan with a UATB feature node.
- The subscriber's MSISDN is provisioned on the BRM with prepaid GSM service enabled.
- The voucher is provisioned on the VWS.

Continued on next page
Recharge using VWS vouchers, Continued

Here is an example message flow covering the IVR, VWS voucher recharge scenario.
Recharge using VWS vouchers, Continued

This scenario describes subscriber who uses IVR to 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.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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. |
| 2      | The CCS service loader looks up the subscriber and wallet in the SLC database 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.  
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. |
| 7      | BRM responds to the operation, indicating that the account has been successfully recharged.  
The BCD Client sends the output flist from the operation in an event (BcdSleeEvent) to the slee_acs process. It also marks the BRM connection as available and cancels the operation timer.  
The BCD actions library takes the output flist and translates it to the response of the WalletRecharge action. |

Continued on next page
Recharge using VWS vouchers, Continued

IVR VWS voucher recharge scenario (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
</table>
| 8      | - 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. |
| 9      | - 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. |
| 10     | - The VoucherRecharge feature node sends PlayAnnouncement to MSC.  
- MSC plays an announcement to the caller, stating that the recharge was successful. |
| 11     | - 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. |
| 12     | - 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. |
| 13     | - The announcement ends and the MSC sends SpecializedResourceReport to the slee_acs process. |
| 14     | - The control plan reaches an end node and ACS sends DisconnectForwardConnection, ReleaseCall to the MSC and clears the call context.  
- The caller is disconnected. |

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

**Operation:** Send PCM_OP_BILL_DEBIT (105)

```plaintext
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] "gsm/ncc"
0 PIN_FLD_DEBIT       ARRAY [978] allocated 20, used 1
1    PIN_FLD_BAL_OPERAND DECIMAL [0] -0.500000000000000
```

**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)

```plaintext
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 [0] allocated 20, used 17
```

Continued on next page
Recharge using VWS vouchers, Continued

Messages: redemption of VWS voucher against BRM account (continued)

<table>
<thead>
<tr>
<th>Level</th>
<th>Field</th>
<th>Data Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>PIN_FLD_IMPACT_TYPE</td>
<td>ENUM [0]</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>PIN_FLD_RESOURCE_ID</td>
<td>INT [0]</td>
<td>978</td>
</tr>
<tr>
<td>2</td>
<td>PIN_FLD_RATE_TAG</td>
<td>STR [0]</td>
<td>&quot;&quot;</td>
</tr>
<tr>
<td>2</td>
<td>PIN_FLD_RATE_OBJ</td>
<td>POID [0]</td>
<td>0.0.0.1 /account 135267 0</td>
</tr>
<tr>
<td>2</td>
<td>PIN_FLD_RATE_OBJ</td>
<td>POID [0]</td>
<td>0.0.0.0 0 0 0 0</td>
</tr>
<tr>
<td>2</td>
<td>PIN_FLD_DISCOUNT</td>
<td>DECIMAL [0]</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>PIN_FLD_PERCENT</td>
<td>DECIMAL [0]</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>PIN_FLD_QUANTITY</td>
<td>DECIMAL [0]</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>PIN_FLD_AMOUNT_DEFERRED</td>
<td>DECIMAL [0]</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>PIN_FLD_AMOUNT</td>
<td>DECIMAL [0]</td>
<td>-0.500</td>
</tr>
<tr>
<td>2</td>
<td>PIN_FLD_AMOUNT_ORIG</td>
<td>DECIMAL [0]</td>
<td>NULL pin_decimal_t ptr</td>
</tr>
<tr>
<td>2</td>
<td>PIN_FLD_BAL_IMPACTS</td>
<td>ARRAY [0]</td>
<td>allocated 20, used 3</td>
</tr>
<tr>
<td>2</td>
<td>PIN_FLD_BAL_GRP_OBJ</td>
<td>POID [0]</td>
<td>0.0.0.1 /balance_group 136803 0</td>
</tr>
<tr>
<td>2</td>
<td>PIN_FLD_GL_ID</td>
<td>INT [0]</td>
<td>978</td>
</tr>
<tr>
<td>2</td>
<td>PIN_FLD_ITEM_OBJ</td>
<td>POID [0]</td>
<td>0.0.0.1 /item/misc 138851 36</td>
</tr>
<tr>
<td>2</td>
<td>PIN_FLD_LINEAGE</td>
<td>STR [0]</td>
<td>NULL str ptr</td>
</tr>
<tr>
<td>3</td>
<td>PIN_FLD_AMOUNT</td>
<td>DECIMAL [0]</td>
<td>-0.500</td>
</tr>
<tr>
<td>3</td>
<td>PIN_FLD_VALID_FROM</td>
<td>TSTAMP [0]</td>
<td>(0) &lt;null&gt;</td>
</tr>
<tr>
<td>3</td>
<td>PIN_FLD_VALID_TO</td>
<td>TSTAMP [0]</td>
<td>(0) &lt;null&gt;</td>
</tr>
</tbody>
</table>

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 or PIN. 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
   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_BAL_IMPACTS   ARRAY [1000011] allocated 20, used 8
   1    PIN_FLD_RATE_OBJ   POID [0] 0.0.0.0.1 /account 135267 0
   1    PIN_FLD_RATE_OBJ   POID [0] 0.0.0.0.0 0 0
   1    PIN_FLD_GL_ID     INT [0] 1000011
   1    PIN_FLD_ITEM_OBJ  POID [0] 0.0.0.1 /item/misc 139539 0
   1    PIN_FLD_BAL_GRP_OBJ POID [0] 0.0.0.0.1 /balance_group 136803 247

Note: Fields 10007 and 10008 have been defined as "VOUCHER" and "PIN" respectively in the NccToBrmFieldMapping section of eserv.config. They have also been defined in BRM using Developer Center. BRM can be configured to produce an EDR showing the voucher number and PIN.

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_RATE_OBJ   POID [0] 0.0.0.1 /account 135267 0
   2    PIN_FLD_RATE_OBJ   POID [0] 0.0.0.0.0 0 0
   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] 0.0.0.0.1 /balance_group 136803 247

Continued on next page
Recharge using VWS vouchers, Continued

Messages: redemption of VWS voucher using bad PIN (continued)

```
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
1     PIN_FLD_SUB_BALANCES  ARRAY [4] allocated 20, used 8
1     PIN_FLD_ACCOUNT_OBJ   POID [0] 0.0.0.1 /account 135267 0
2     PIN_FLD_POID          POID [0] 0.0.0.1 /event/activity/voucher
```

271148363502725395 0
Data charging

The scenario in this section describes a subscriber who initiates and later closes a data session using a mobile device. The GGSN communicates with the NCC software on the SLC by way of Diameter credit control operations. NCC charges the data session against an account on BRM. For information on creating a BRM product and deal for data calls, see Creating a Product and Deal for Data Calls.

This scenario assumes the following preconditions:
- The subscriber's MSISDN is provisioned on the NCC system.
- The subscriber is provisioned so that data calls run a control plan with a UATB feature node.
- The subscriber's MSISDN is provisioned on BRM with prepaid GSM service enabled.
- The subscriber has sufficient money on his account for the data session.

Here is an example message flow covering the data session charging scenario.

This scenario describes a data session that is initiated and later closed by a subscriber using a mobile device.

Continued on next page
Data charging, Continued

Charging for a Data Session (continued)

See Preconditions for Data Session Charging on BRM for information about the preconditions for this scenario.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1      | - The subscriber starts a data session.  
        | - The GGSN sends Diameter Credit Control Request (CCR), INITIAL_REQUEST to DCA |
| 2      | The DCA sends an InitialDP operation to the slee_acs. |
| 3      | - 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 SLC database and determines that the wallet information is on the BRM domain. It also determines the control plan to run.  
        | - The slee_process runs the control plan.  
        | - The slee_acs process reaches a Universal Attempt Terminate with Billing (UATB) node in the control plan and invokes the InitialTimeReservation action on the BCD actions library.  
        | - The BCD actions library constructs an event (BcdSleeEvent) to invoke a PCM_OP_TCF_AAA_AUTHORIZE operation and sends it to the least busy BCD Client.  
        | - The BCD Client finds a free connection on the Connection Manager with the lowest proportion of its connections currently in use.  
        | - The BCD Client calls the PCM_OP_TCF_AAA_AUTHORIZE opcode and sets a timer to the configured value for this type of operation. |
| 4      | BRM responds to the operation by indicating that 5 megabytes of data can be used.  
        | - The BCD Client sends the output flist for this operation in an event (BcdSleeEvent) to slee_acs. It also marks the BRM connection as available and cancels the operation timer.  
        | - The BCD actions library translates the output flist into the response to the InitialTimeReservation action. It also copies the value of 5 megabytes into the tag used for this purpose by the DCA software. |
| 5      | The slee_acs process sends RequestreportBCSMEvent, ApplyCharging, and Connect operations to DCA. The 5 megabytes of available data is held in an extension in ApplyCharging. |
| 6      | DCA sends Credit Control Answer (CCA) to the GGSN to specify that Granted Units = 5 megabytes. |
| 7      | After some time, the GGSN sends CCR(UPDATE_REQUEST) to DCA, indicating that 5 megabytes have been used. |
| 8      | DCA sends ApplyChargingReport(callActive=true) to the slee_acs process. |

Continued on next page
Data charging, Continued

Charging for a Data Session (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
</table>
| 9      | • The slee_acs process invokes an ExtendTimeReservation action on the BCD actions library.  
• The BCD actions library constructs an event (BcdSleeEvent) for the `PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE` opcode for 5 megabytes and sends it to the least busy BCD Client.  
• The BCD Client has free BRM connections and so chooses the most recently used connection.  
• The BCD Client calls the `PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE` opcode and sets a timer to the configured value for this type of operation. |
| 10     | • BRM responds to the operation indicating that an additional 5 megabytes may be used.  
• The BCD Client sends the output flist from the operation in an event (BcdSleeEvent) to the slee_acs process. It also marks the BRM connection as available and cancels the operation timer.  
• The BCD actions library translates the output flist into the response to the ExtendTimeReservation action. It also places the 5 megabytes of allowed usage in the standard profile tag used by DCA. |
| 11     | The slee_acs process sends an ApplyCharging operation to the DCA.  
| 12     | DCA sends CCA to the GGSN to specify Granted Units = 5 megabytes.  
| 13     | • After using an additional 2 megabytes, the user terminates the session.  
• GGSN sends CCR(TERMINATION_REQUEST) to DCA indicating that 2 additional megabytes have been used. |
| 14     | DCA sends ApplyChargingReport(callActive=false), EventReportBCSM(oDisconnect) to the slee_acs process. The value of 7 megabytes is passed in an extension in ApplyChargingReport.  
| 15     | • The slee_acs process invokes a ConfirmTimeReservation action on the BCD actions library. It stores the value of 7 megabytes in context.  
• The BCD actions library constructs an event (BcdSleeEvent) for invoking the `PCM_OP_TCF_AAA_STOP_ACCOUNTING` opcode, indicating that 7 megabytes have been used and sends it to the least busy BCD Client.  
• The BCD Client has free connections and chooses the most recently used connection.  
• The BCD Client calls the `PCM_OP_TCF_AAA_STOP_ACCOUNTING` opcode and sets a timer to the configured value for this type of operation. |

Continued on next page
### Data charging, Continued

#### Charging for a Data Session (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
</table>
| 16     | - BRM responds to the operation indicating that the cost of 7 megabytes of data has been permanently deducted from the account.  
- The BCD Client sends the output flist from the operation in an event (BcdSleeEvent) to the slee_acs process. It also marks the connection as available and cancels the operation timer.  
- The BCD actions library translates the output flist into the response to the ConfirmTimeReservation action. It also stores the charge information in call context for possible use in the future. |
| 17     | - The slee_process proceeds with the control plan. If there are no more feature nodes, processing for this section ends. |

#### Messages: data session

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

**Operation:** Send PCM_OP_TCF_AAA_AUTHORIZE (4002)

Flags = 0

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 PIN_FLD_POID</td>
<td>POID [0] 0.0.0.1 /service/telco/gsm/data -1 0</td>
</tr>
<tr>
<td>0 PIN_FLD_PROGRAM_NAME</td>
<td>STR [0] &quot;NCC_BCD_Client&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_AUTHORIZATION_ID</td>
<td>STR [0] &quot;brmClient-cleejoh_session_126042_0&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_OBJ_TYPE</td>
<td>STR [0] &quot;gsm/ncc&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_CALLING_NUMBER</td>
<td>STR [0] &quot;004085752158&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_CALLED_NUMBER</td>
<td>STR [0] &quot;55587390000&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_EXTENDED_INFO</td>
<td>SUBSTRUCT [0] allocated 20, used 2</td>
</tr>
<tr>
<td>1 PIN_FLD_GSM_INFO</td>
<td>SUBSTRUCT [0] allocated 20, used 3</td>
</tr>
<tr>
<td>2 PIN_FLD_DIRECTION</td>
<td>ENUM [0] 0</td>
</tr>
<tr>
<td>2 PIN_FLD_CELL_ID</td>
<td>STR [0] &quot;000c&quot;</td>
</tr>
<tr>
<td>2 PIN_FLD_LOC_AREA_CODE</td>
<td>STR [0] &quot;064001000f&quot;</td>
</tr>
<tr>
<td>1 10000</td>
<td>SUBSTRUCT [0] allocated 20, used 2</td>
</tr>
<tr>
<td>2 10001</td>
<td>STR [0] &quot;Present&quot;</td>
</tr>
<tr>
<td>2 PIN_FLD_LOCATION</td>
<td>STR [0] &quot;7390002&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_MSID</td>
<td>STR [0] &quot;004085752158&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_REQ_MODE</td>
<td>ENUM [0] 4</td>
</tr>
</tbody>
</table>

**Note:** REQ_MODE now indicates VOLUME

**Result:** received for operation PCM_OP_TCF_AAA_AUTHORIZE (4002)

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 PIN_FLD_POID</td>
<td>POID [0] 0.0.0.1 /active_session/telco/gsm/ncc 138601 0</td>
</tr>
<tr>
<td>0 PIN_FLD_QUANTITY</td>
<td>DECIMAL [0] 5000000</td>
</tr>
<tr>
<td>0 PIN_FLD_SERVICE_OBJ</td>
<td>POID [0] 0.0.0.1 /reservation 136041 0</td>
</tr>
<tr>
<td>0 PIN_FLD_BAL_GRP_OBJ</td>
<td>POID [0] 0.0.0.1 /balance_group 136043 203</td>
</tr>
<tr>
<td>1 PIN_FLD_BALANCES</td>
<td>ARRAY [978] allocated 20, used 2</td>
</tr>
<tr>
<td>1 PIN_FLD_AVAILABLE_RESOURCE_LIMIT</td>
<td>DECIMAL [0] 9.952</td>
</tr>
<tr>
<td>1 PIN_FLD_RESULT</td>
<td>ENUM [0] 1</td>
</tr>
<tr>
<td>0 PIN_FLD_RATEID</td>
<td>ENUM [0] 0</td>
</tr>
<tr>
<td>0 PIN_FLD_AUTHORIZATION_ID</td>
<td>STR [0] &quot;brmClient-cleejoh_session_126042_0&quot;</td>
</tr>
<tr>
<td>0 PIN_FLD_ACCOUNT_OBJ</td>
<td>POID [0] 0.0.0.1 /account 135267 0</td>
</tr>
</tbody>
</table>

**Note:** PIN_FLD_QUANTITY says 5 million bytes.

**Operation:** Send PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE (4026)

Flags = 0

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 PIN_FLD_POID</td>
<td>POID [0] 0.0.0.1 /service/telco/gsm/data -1 0</td>
</tr>
</tbody>
</table>

*Continued on next page*
Data charging, Continued

Messages: data session (continued)

```
0 PIN_FLD_PROGRAM_NAME STR [0] "NCC_BCD_Client"
0 PIN_FLD_AUTHORIZATION_ID STR [0] "brmClient-cleejoh_session_126042_0"
0 PIN_FLD_OBJ_TYPE STR [0] "gsm/ncc"
0 PIN_FLD_CALLING_NUMBER STR [0] "004085752158"
0 PIN_FLD_CALLED_NUMBER STR [0] "55587390000"
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 2
  1 10001 STR [0] "Present"
  2 PIN_FLD_LOCATION STR [0] "7390002"
0 PIN_FLD_MSID STR [0] "004085752158"
0 PIN_FLD_REQ_MODE ENUM [0] 4
0 PIN_FLD_BYTES_UPLINK DECIMAL [0] 5000000.000000000000000
Note: subscriber has uploaded 5 million bytes.
Result: Received for operation PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE (4026)

0 PIN_FLD_POID POID [0] 0.0.0.1 /active_session/telco/gsm/ncc 138601 2
0 PIN_FLD_EXPIRATION_T TSTAMP [0] (1331837317) Thu Mar 15 18:48:37 2012
0 PIN_FLD_QUANTITY DECIMAL [0] 5000000
0 PIN_FLD_RESERVATION_OBJ POID [0] 0.0.0.1/reservation 136041 1
0 PIN_FLD_BAL_GRP_OBJ POID [0] 0.0.0.1/balance_group 136803 205
0 PIN_FLD_BALANCES ARRAY [978] allocated 20, used 2
  1 PIN_FLD_AMOUNT DECIMAL [0] 0.048
  1 PIN_FLD_AVAILABLE_RESOURCE_LIMIT DECIMAL [0] 9.857
0 PIN_FLD_RESULT ENUM [0] 1
0 PIN_FLD_RATING_STATUS ENUM [0] 0
0 PIN_FLD_AUTHORIZATION_ID STR [0] "brmClient-cleejoh_session_126042_0"
Note: The subscriber is now allowed to upload 5 million more bytes
Operation: Send PCM_OP_TCF_AAA_STOP_ACCOUNTING (4007)

Flags = 0
0 PIN_FLD_POID POID [0] 0.0.0.1 /service/telco/gsm/data -1 0
0 PIN_FLD_PROGRAM_NAME STR [0] "NCC_BCD_Client"
0 PIN_FLD_AUTHORIZATION_ID STR [0] "brmClient-cleejoh_session_126042_0"
0 PIN_FLD_OBJ_TYPE STR [0] "gsm/ncc"
0 PIN_FLD_CALLING_NUMBER STR [0] "004085752158"
0 PIN_FLD_CALLED_NUMBER STR [0] "55587390000"
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 2
  1 10001 STR [0] "Present"
  2 PIN_FLD_LOCATION STR [0] "7390002"
  2 10007 STR [0] "55587390000"
0 PIN_FLD_MSID STR [0] "004085752158"
0 PIN_FLD_REQ_MODE ENUM [0] 4
0 PIN_FLD.Bytes_UPLINK DECIMAL [0] 1300000.000000000000000
Note: the subscriber has uploaded 1.3 million more bytes and ended the session.
Result: received for operation PCM_OP_TCF_AAA_STOP_ACCOUNTING (4007)

0 PIN_FLD_POID POID [0] 0.0.0.1 /event/session/telco/gsm/ncc 27114836502721926 0
0 PIN_FLD_AUTHORIZATION_ID STR [0] "brmClient-cleejoh_session_126042_0"
0 PIN_FLD_ACCOUNT_OBJ POID [0] 0.0.0.1 /account 135267 0
0 PIN_FLD_SERVICE_OBJ POID [0] 0.0.0.1 /service/telco/gsm/data 135651 10
0 PIN_FLD_RATING_STATUS ENUM [0] 0
0 PIN_FLD_BALANCES ARRAY [978] allocated 20, used 2
```

Continued on next page
Data charging, Continued

Messages: data session (continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Field Name</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PIN_FLD_AMOUNT</td>
<td>DECIMAL [0]</td>
<td>0.108</td>
</tr>
<tr>
<td>1</td>
<td>PIN_FLDAVAILABLERESOURCELIMIT</td>
<td>DECIMAL [0]</td>
<td>9.892</td>
</tr>
</tbody>
</table>
Other messages

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)

```plaintext
Error specified:
location=5
pin_errclass=4
pin_err=3
field=8393688
rec_id=0
reserved=0
facility=0
msg_id=0
version=0
```

The error code of 3 is defined like this:

```c
#define PIN_ERR_NOT_FOUND                    3
```

**Operation:** Send PCM_OP_READ_FLDS (4)

```plaintext
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)

```plaintext
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_LAST_STATUS_T TSTAMP [0] (1331675046) Tue Mar 13 21:44:06 2012
```

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

**Operation:** Send PCM_OP_TCF_AAA_AUTHORIZE (4002)

```plaintext
Flags = 0
0 PIN_FLD_POID       POID [0] 0.0.0.1/service/telco/gsm/sms -1 0
0 PIN_FLD_PROGRAM_NAME STR [0] "NCC_BCD_Client"
0 PIN_FLD_AUTHORIZATION_ID STR [0] "brmClient-cleejoh_ne_96032_0"
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] "7390002"
    2 100G2 INT [0] 8
0 PIN_FLD_MSID        STR [0] "004085752158"
0 PIN_FLD_REQ_MODE   ENUM [0] 2
```

**Note:** The POID and the OBJ_TYPE are derived from the NCC event class name and the NCC event name respectively.

Field 10002 has been defined (using BRM Developer Center) as "NUMBER OF EVENTS" and BRM is set to rate on this field for this POID and object type. It is 8 because the billable event node has specified that the number of events = 8.

Continued on next page
Other messages, Continued

Messages: direct named event reservation (continued)

Operation: Send PCM_OP_TCF_AAA_STOP_ACCOUNTING (4007)

Flags = 0
pin_flist_t* bcd::BcdSleeEvent::produceFlist() const

About to un-serialize flist

# number of field entries allocated 20, used 7
0 PINFldPOID POID [0] 0.0.0.1 /service/telco/gsm/sms -1 0
0 PINFldProgramName STR [0] "NCC_BCD_Client"
0 PINFldAuthorizationID STR [0] "brmClient-cleejoh_ne_96032_0"
0 PINFldOBJTYPE STR [0] "gsm/ncc"
0 PINFldEXTENDED_INFO SUBSTRUCT [0] allocated 20, used 2
1 PINFldGSMINFO SUBSTRUCT [0] allocated 20, used 3
2 PINFldDIRECTION ENUM [0] 0
2 PINFldCELLID STR [0] "000c"
2 PINFldLOC_AREA_CODE STR [0] "064001000f"
1 10000 SUBSTRUCT [0] allocated 20, used 3
2 10001 STR [0] "Present"
2 PINFldLOCATION STR [0] "7390002"
2 10002 INT [0] 8
0 PINFldMSID STR [0] "004085752158"
0 PINFldREQMODE ENUM [0] 2

The following messages include operations sent to BRM and results returned by BRM for a direct named event for a non-reservable event, in which case a purchase deal is used. The general message format is: nesting level (0; 1, or 2); field; data type; value.

Operation: Send PCM_OP_CUST_POL_GET_DEALS (278)

Flags = 0
# number of field entries allocated 20, used 1
0 PINFldPOID POID [0] 0.0.0.1 /account 135267 0

Result: Received for operation PCM_OP_CUST_POL_GET_DEALS (278)

0 PINFldPOID POID [0] 0.0.0.1 /account 135267 0
0 PINFldDEALS ARRAY [10] allocated 20, used 14
1 PINFldPOID POID [0] 0.0.0.1 /deal 90056 4
1 PINFldCREATED T TSTAMP [0] (1327455797) Wed Jan 25 01:43:17 2012
1 PINFldMOD T TSTAMP [0] (1327455952) Wed Jan 25 01:45:52 2012
1 PINFldREADACCESS STR [0] "R"
1 PINFldWRITEACCESS STR [0] "W"
1 PINFldACCOUNTOBJ POID [0] 0.0.0.1 /account 1 0
1 PINFldCODER STR [0] "50 Euro Topup Deal"
1 PINFldDESCRIPTOR STR [0] ""
1 PINFldEND T TSTAMP [0] (0) <null>
1 PINFldFLAGS INT [0] 0
1 PINFldNAME STR [0] "50 Euro Topup Deal"
1 PINFldPERMITTED STR [0] "/account"
1 PINFldSTART T TSTAMP [0] (0) <null>
1 PINFldPRODUCTS ARRAY [0] allocated 20, used 19
2 PINFldCYCLEDISCOUNT DECIMAL [0] 0
2 PINFldCYCLEENDEDETAILS INT [0] 2
2 PINFldCYCLEEND T TSTAMP [0] (0) <null>
2 PINFldCYCLESTARTDETAILS INT [0] 1
2 PINFldPRODUCTOBJ POID [0] 0.0.0.1 /product 82729 6
2 PINFldPURCHASEDISCOUNT DECIMAL [0] 0
2 PINFldPURCHASEENDEDETAILS INT [0] 2
2 PINFldPURCHASEEND T TSTAMP [0] (0) <null>
2 PINFldPURCHASESTARTDETAILS INT [0] 1
2 PINFldPURCHASESTART T TSTAMP [0] (0) <null>
2 PINFldQUANTITY DECIMAL [0] 1
2 PINFldSTATUS ENUM [0] 1

Continued on next page
Other messages, Continued

Messages: direct named event for non-reservable event (continued)

```
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
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_DEAL_INFO SUBSTRUCT [0] allocated 20, used 1
1 PIN_FLD_DEAL_OBJ  POID [0] 0.0.0.1 /deal 88008 4
```

**Operation:** Send PCM_OP_SUBSCRIPTION_PURCHASE_DEAL (108)

```
Flags = 0
pin_flist_t* bcd::BcdSleeEvent::produceFlist() const
About to un-serialise flist
# number of field entries allocated 20, used 3
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.DEAL_INFO SUBSTRUCT [0] allocated 20, used 1
1 PIN_FLD.DEAL_OBJ  POID [0] 0.0.0.1 /deal 88008 4
```

**Note:** We select deal 88008: DirectEventDeal, because the NCC event name is DirectEventDeal.

**Result:** Received for operation PCM_OP_SUBSCRIPTION_PURCHASE_DEAL (108)

```
0 PIN_FLD_POID  POID [0] 0.0.0.1 /account 135267 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 135267 13
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"
```

Continued on next page
Other messages, Continued

Messages: direct named event for non-reservable event (continued)

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_FLDRESOURCE_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 136803
260
2    PIN_FLD_ITEM_OBJ    POID [0] 0.0.0.1 /item/misc 138851 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 136803
261
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 135267 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 27114836352715087 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 135267 0
1    PIN_FLD.POID    POID [0] 0.0.0.1
/event/billing/product/action/purchase 17592186177743 0
0    PIN_FLD_PRODUCTS    ARRAY [0] allocated 20, used 2
1    PIN_FLD_PACKAGE_ID    INT [0] 150
0    PIN_FLD_RESULTS    ARRAY [2] allocated 20, used 2
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/deal/purchase
17592186176719 0
## Appendix Overview

This appendix contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix A</td>
<td>112</td>
</tr>
<tr>
<td>Glossary of Terms</td>
<td>119</td>
</tr>
<tr>
<td>Index</td>
<td>123</td>
</tr>
</tbody>
</table>
Appendix A

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

Example eserv.config file

```
BCD = {
  bcdActionHandler = {
    # The interface name of the BRM billing client (in SLEE.cfg)
    serviceDomainInterfaceName = "bcdBeClient"
    
    # Client ID string is a unique string so that messages from each client going to a BRM server will not have the same authentication IDs, and hence won't get confusing # duplicate errors. Defaults to the hostname if absent from this file
    clientIDString = "client1"
    
    # How often to announce the number of recent message parse errors.
    loggedNotificationPeriod = 300
    
    # Seconds to hold back as the low credit buffer
    lowCreditBufferTime = 10
    
    # When rounding sub-second durations into seconds specifies the rounding scheme
    # 1 == floor
    # 2 == ceiling
    # 3 == nearest
    roundingScheme = 3
    
    # Maximum age of cached billable event and balance type map data. # The data will be re-read if it is more than cacheTimeout seconds old when needed.
    cacheTimeout = 60
    
    # Events will not be sent to brmBeClients clients with more than this # number of events outstanding.
    maxOutstandingRequests = 1000
    
    # The length of the voucher Pin
    VoucherPinLength = 4
    
    # First part of the service POID string to be used by named events. The event # class (the name of the billable event set used in the control plan) is # appended onto it, forming the complete POID name.
```

Continued on next page
Appendix A, Continued

Example eserv.config file (continued)

```plaintext
# e.g. poidPrefix = "/service/telco/"
#      eventClass = "gsm/sms"
# ->  POID = "/service/telco/gsm/sms"

poidPrefix = "/service/telco/"

# These array entries allow users to map service key / bearer type combinations
# to specific BRM services. If service key or bearer type
# cannot be found, then -1 is
# used. Hence the fallback config is the one with ServiceKey = -1 & BearerID = -1

ServiceProfileTagMapping = [
    # Default fall back config. Basic duration measured
    # voice call
    ServiceKey = -1 # default
    BearerID = -1 # default
    ScalingFactor = 1
    BRMField = "QUANTITY" # or BYTES_UPLINK,
    BRMReqMode = "DURATION" # or VOLUME
    BrmServicePoid = "/service/telco/gsm/telephony"
    BrmObjectType = "gsm"
    UsedUnitsCumulative = false
    DefaultUnitType = "QUANTITY" # or UP_BYTES,

    Bytes_DOWNLINK
    ServiceKey = 1
    BearerID = 17
    ScalingFactor = 100000 # Bytes per deci-second
    (equates to 1Mb per second)
    BRMField = "BYTES_UPLINK"
    BRMReqMode = "VOLUME"
    BrmServicePoid = "/service/telco/gsm/data"
    BrmObjectType = "gsm"
    UsedUnitsCumulative = false
    DefaultUnitType = "UP_BYTES"

    DOWN(Bytes)

    # Specific configuration for data calls.
    ServiceKey = 1
    BearerID = 17
    ScalingFactor = 100000 # Bytes per deci-second
    BRMField = "BYTES_UPLINK"
    BRMReqMode = "VOLUME"
    BrmServicePoid = "/service/telco/gsm/data"
    BrmObjectType = "gsm"
    UsedUnitsCumulative = false
    DefaultUnitType = "UP_BYTES"

] # These entries map BRM currency numbers to NCC currency codes. BRM numbers are
# defined in pin_currency.h, but only those that are used
# need to be put here.

BrmToNccCurrencyMapping = [
    # NZD
    NCCCode = "NZD"
    BRMNum = 554

    # EUR
    NCCCode = "EUR"
    BRMNum = 978
]
```

Continued on next page
Example eserv.config file (continued)

```plaintext
# Should the BadPIN action trigger the sending of an EDR to BRM?
BrmBadPinEdrActive = false

# This is the BRM object type to use when creating an EDR in the CreateEdr action
BrmEdrObjectType = "/voucher"

# This is the BRM resource ID used to count bad pin attempts.  # If this is zero, then the CreateEDR action will merely send the voucher number  # and pin number used for the failed voucher redeem attempt
BrmBadPinResourceId = 1000011

# This is the BRM field ID of the flist substruct field under which all the NCC specific # fields get added. Normally this will be the numeric value associated with  # PIN_FLD_NCC_INFO when the BRM instance was customised.
NccInfoFieldNumber = 10000

# This is the BRM field ID of a string field configured to be present in the  # PIN_FLD_NCC_INFO substruct. The field should not be important for rating, and will # be set to "Present" by the action handler. It is required as the PIN_FLD_NCC_INFO must  # be present in the flist passed to opcodes, but it is not valid to have an empty  # substruct. Hence the dummy value.
NccInfoFieldDummyEntry = 10001

# These array entries map EDR extra information items, or other NCC concepts to BRM  # fields, and their associated types.  # It is important that the BRM custom fields have been created, and that the necessary  # re-compilation of BRM code has taken place before attempting to use these custom  # fields. If an EDR item is not listed, it will not be copied into the BRM flist.
NccToBrmFieldMapping = {
    NCCItem = "TARIFF_PLAN_ID" # Text name of EDR Extra Information item, or other # NCC concept
    BRMType = "INT" # BRM data type (STRING, DECIMAL, INT, ENUM,  # TIMESTAMP)
    BRMField = 10004 # Numeric ID of BRM custom field (found in  # pin_flds.h)
}

Continued on next page
```
Appendix A, Continued

Example eserv.config file (continued)

```plaintext
{
    NCCItem = "NUMBER_OF_EVENTS_ID" # Text name of EDR Extra Information item, or other NCC concept
    BRMType = "INT" # BRM data type (STRING, DECIMAL, INT, ENUM, # TIMESTAMP)
    BRMField = 10002 # Numeric ID of BRM custom field (found in # pin_flds.h)
}
{
    NCCItem = "EXAMPLE" # Text name of EDR Extra Information item, or other NCC concept
    BRMType = "STRING" # BRM data type (STRING,
    DECIMAL, INT, ENUM,
    # TIMESTAMP)
    BRMField = 10005 # Numeric ID of BRM custom field (found in
    # pin_flds.h)
}
{
    NCCItem = "VOUCHER" # Text name of EDR item, or other NCC concept
    BRMType = "STRING" # BRM data type (STRING,
    DECIMAL, INT, ENUM,
    # TIMESTAMP)
    BRMField = 10007 # Numeric ID of BRM custom field (found in
    # pin_flds.h)
}
{
    NCCItem = "PIN" # Text name of EDR item, or other NCC concept
    BRMType = "STRING" # BRM data type (STRING,
    DECIMAL, INT, ENUM,
    # TIMESTAMP)
    BRMField = 10008 # Numeric ID of BRM custom field (found in
    # pin_flds.h)
}
```

# These entries control how the location number is retrieved from ACS, and sent to BRM
# The BRMField refers to the BRM field ID of a string field in the GSM_INFO substruct.
# If this is zero, then 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

Continued on next page
Example eserv.config file (continued)

```plaintext
# 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 when
# retrieving data from the profile block, no data is found.
# If either of the secondary location parameters are zero,
the profile block will not be
# searched.

LocationNumberMapping = {
  BRMField = 1251                  # Or zero to disable
  PrimaryLocationNumberProfileBlock = 18     # Call
  PrimaryLocationNumberProfileTag = 327692   #
  PT_CC_LOCATION_NUMBER. Must be non-zero
  SecondaryLocationNumberProfileBlock = 18   # Call
  SecondaryLocationNumberProfileTag = 327716 #
  PT_CC_LOCATION_INFO_LOCATION_NUMBER.        # or zero to
disable second choice for
  # location number.
}
```

```plaintext
) # End of bcdActionHandler

bcdBillingClient = {
  # PCM Context Open timeout in seconds
  # The time to wait for a BRM PCM Context to open
  contextOpenTimeout = 5

  # Seconds to turn off marking any ip address/port
  # combination as not working, such that it can be tried again for reuse
  recoverCmPtrSeconds = 5

  # maximum number of milliseconds to select before checking
  # SLEE events.
  # Default = 50
  maxPollMilliseconds = 1

  # maximum number of tries, including the first, that will
determine retry behaviour
  # when trying to send an Op to BRM.
  maxTries = 3

  # Interval between updates of the latency statistic
  # recording in seconds.
  # default 300
```
Example eserv.config file (continued)

```plaintext
latencyStatisticsInterval = 300

# Whether to include the destination Communications Manager IP Address in stats details.
recordCMIPAddressInStats = false

# Whether to include the Operation Code in stats details.
recordOpcodeInStats = false

# Whether to include the destination Communications Manager port in stats details.
recordPortInStats = false

# Opcode mapping
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_TCF AAA_AUTHORIZE", opCode = 4002 } 
{operation = "PCM_OP_TCF AAA CANCEL AUTHORIZATION", opCode = 4004 } 
{operation = "PCM_OP_TCF AAA QUERY BALANCE", opCode = 4104 } 
{operation = "PCM_OP_TCF AAA STOP ACCOUNTING", opCode = 4007 } 
{operation = "PCM_OP _TCF AAA STOP ACCOUNTING PREP INPUT", opCode = 4031 } 
{operation = "PCM_OP_TCF AAA UPDATE AND REAUTHORIZE", opCode = 4026 } 
{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 ACTIVITY", opCode = 151 } 
{operation = "PCM_OP_CUST POL GET DEALS", opCode = 278 } 
{operation = "PCM_OP BILL DEBIT", opCode = 105 } 
]

# Configuration items required to open a context on a BRM connection manager.
# If multiple entries are supplied in cmPointers, the BRM connection manager will
# handle selection of the actual machine / port to use
# cmPtrs have protocol "ip", ip address in decimal and the port number
# poolSize is how many connections to open to each of the BRM servers
# The actual login and password for the BRM machine are to be stored in the database
# and can be set by configuring the BCD domain in the New and Edit Domain dialogs

ConnectionManager = {
```
database = 1
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 }
]

# Default operation timeout to use if a specific opcode does
# not have an entry in the
# OperationTimeouts array.
# Default 250
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_TCF_AAA_AUTHORIZE" ,
        timeoutMilliseconds = 100 }
    {operation = "PCM_OP_TCF_AAA_QUERY_BALANCE" ,
        timeoutMilliseconds = 20 }
    {operation = "PCM_OP_TCF_AAA_STOP_ACCOUNTING" ,
        timeoutMilliseconds = 300 }
    {operation = "PCM_OP_TCF_AAA_UPDATE_AND_REAUTHORIZE" ,
        timeoutMilliseconds = 250 }
    {operation = "PCM_OP_TCF_AAA_STOP_ACCOUNTING_PREP_INPUT" ,
        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

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>Authentication, Authorization, and Accounting. Specified in Diameter RFC 3588.</td>
</tr>
<tr>
<td>ACS</td>
<td>Advanced Control Services configuration platform.</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BE</td>
<td>Billing Engine</td>
</tr>
<tr>
<td>BFT</td>
<td>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.</td>
</tr>
<tr>
<td>CAMEL</td>
<td>Customized Applications for Mobile network Enhanced Logic</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>CC</td>
<td>Country Code. Prefix identifying the country for a numeric international address.</td>
</tr>
<tr>
<td>CCA</td>
<td>Credit-Control-Answer, used in Diameter by the credit-control server to acknowledge a Credit-Control-Request (CCR) from the credit-control client.</td>
</tr>
<tr>
<td>CCR</td>
<td>Credit-Control-Request, used in Diameter by the credit-control client to request credit authorization from the credit-control server.</td>
</tr>
<tr>
<td>CCS</td>
<td>1) Charging Control Services (or Prepaid Charging) component.  2) Common Channel Signalling. A signalling system used in telephone networks that separates signalling information from user data.</td>
</tr>
<tr>
<td>Connection</td>
<td>Transport level link between two peers, providing for multiple sessions.</td>
</tr>
<tr>
<td>CORBA</td>
<td>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.</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>Diameter</td>
<td>A feature rich AAA protocol. Utilises SCTP and TCP transports.</td>
</tr>
<tr>
<td>DP</td>
<td>Detection Point</td>
</tr>
<tr>
<td>DTMF</td>
<td>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.</td>
</tr>
<tr>
<td><strong>EDR</strong></td>
<td>Event Detail Record</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------------</td>
</tr>
<tr>
<td><strong>Note:</strong> Previously CDR. The industry standard for CDR is EDR (Event Detail Record). Over time EDR will replace CDR in the Oracle documentation.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>ENUM</strong></th>
<th>E.164 Number Mapping.</th>
</tr>
</thead>
</table>

| **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. |

<table>
<thead>
<tr>
<th><strong>HPLMN</strong></th>
<th>Home PLMN</th>
</tr>
</thead>
</table>

| **HTML** | HyperText Markup Language, a small application of SGML used on the World Wide Web. It defines a very simple class of report-style documents, with section headings, paragraphs, lists, tables, and illustrations, with a few informational and presentational items, and some hypertext and multimedia. |

| **IN** | Intelligent Network |

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

| **IP** | 1) Internet Protocol 2) Intelligent Peripheral - a box that is able to play announcements |

| **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 Network Charging and Control product. Component acronym is MM (formally MMX). |
| **MIN** | Mobile Identification Number, also known as an MSID. |
| **MM** | Messaging Manager. Formally MMX, see also XMS (on page 122) and Messaging Manager (on page 121). |
| **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 |
| **Oracle** | Oracle Corporation |
| **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 |
| **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). |
| **SGML** | Standard Generalized Markup Language. The international standard for defining descriptions of the structure of different types of electronic document. |
| **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 subscribed operator’s network. |
| **SIP** | Session Initiation Protocol - a signalling 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, either: Short Message Service, or Service Management Server. |
| **SN** | Service Number |
| **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. |
| **USSD** | Unstructured Supplementary Service Data - a feature in the GSM MAP protocol that can be used to provide subscriber functions such as Balance Query and Friends and Family Access. |
| **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 Network Charging and Control Messaging Manager (on page 121) service and the Short Message Service. The published code is MM (on page 121) (formally MMX). |
Index

A

A voice call charged against BRM
A Voice Call When Funds Expire • 79
Caller hangs up flow • 73
Caller hangs up scenario • 74
Funds expire flow • 78
Messages
caller hangs up • 76
funds expire • 80
Preconditions for voice call scenarios • 72
Subscriber not found on BRM flow • 81
Subscriber not found on BRM scenario • 81
A Voice Call When Funds Expire
A voice call charged against BRM • 79
About adding custom fields • 26
Adding custom fields • 50
About BRM
Overview of the BRM Charging Driver • 2
About Creating Balance Type Mappings • 18, 26
Creating balance type mappings • 45
About creating products and deals for direct named events
Creating a product and deal for named events • 62
About editing eserv.config parameters
Editing eserv.config parameters for the BRM Charging Driver • 21
About installing the BRM Charging Driver
Installing the bcdScp package • 16
Installing the bcdSms package • 15
About integrating NCC and BRM
Overview of the BRM Charging Driver • 2
About statistics and reports
Generating reports • 69
Generating statistics • 68
About this Document
Audience • v
Prerequisites • v
Related documents • v
Scope • v
ACS • 2
Adding custom fields
About adding custom fields • 50
Creating additional custom fields • 51
Adding storable classes
Creating custom classes • 53
An SMS call charged against BRM
Preconditions for SMS call scenarios • 83
SMS call delivered successfully flow • 83
SMS call delivered successfully scenario • 84
SMS call that fails permanently flow • 85
SMS call that fails permanently scenario • 85
API • 2
Appendix A • 21, 28, 35
Example eserv.config file • 112
Audience
About this Document • v
B
cbdActionHandler content
Configuring cbdActionHandler • 22
cbdBillingClient content
Configuring cbdBillingClient • 33
BE • 6
BearerID • 23
BFT • 81
BRM and NCC components
Overview of the BRM Charging Driver • 2
BRM Charging Driver features
Overview of the BRM Charging Driver • 6
BRM Charging Driver reports
Overview of the BRM Charging Driver • 5
BRM integration summary
Integrating NCC and BRM • 11
BrmBadPinEdrActive • 28
BrmBadPinResourceId • 28
BrmEdrObjectType • 29
BRMField • 24
BrmObjectType • 25
BRMReqMode • 24
BrmServicePoid • 25
cacheTimeout • 29
Caller hangs up flow
A voice call charged against BRM • 73
Caller hangs up scenario • 79
A voice call charged against BRM • 74
CAMEL • 74
CC • 27
CCA • 102
CCR • 102
CSS • 2
CDR • 120
C. (continued)

Charging for a Data Session
Data charging • 101
clientIDString • 29
Configuring additional bcdActionHandler Parameters
Configuring bcdActionHandler • 28
Configuring additional bcdBillingClient parameters
Configuring bcdBillingClient • 35
Configuring bcdActionHandler
bcdActionHandler content • 22
Configuring additional bcdActionHandler Parameters • 28
Mapping NCC currency codes to BRM values • 26
Mapping NCC information to BRM fields • 26
Mapping NCC sessions to BRM services • 22
Mapping the location number • 27
Configuring bcdBillingClient
bcdBillingClient content • 33
Configuring additional bcdBillingClient parameters • 35
Configuring BRM connections • 34
Mapping opcodes • 33
Specifying operation timeouts • 33
Configuring BRM connections
Configuring bcdBillingClient • 34
Configuring BRM for named event reservations and reservable direct named events • 63
Configuring BRM for the BRM Charging Driver • 11
Configuring BRM for vouchers
Creating BRM vouchers • 65
Configuring calls, events, and vouchers
Configuring NCC for BRM vouchers • 41
Configuring NCC for data calls • 40
Configuring NCC for named events • 41
Configuring NCC for voice calls • 40
Tracking BRM voucher redemptions with bad PINs • 41
Configuring named event refunds • 41, 64
Configuring NCC for BRM vouchers
Configuring calls, events, and vouchers • 41
Configuring NCC for data calls
Configuring calls, events, and vouchers • 40
Configuring NCC for named event reservations and reservable direct named events • 41

C. (continued)

Configuring NCC for named events
Configuring calls, events, and vouchers • 41
Configuring NCC for the BRM Charging Driver • 11
Configuring NCC for voice calls
Configuring calls, events, and vouchers • 40
Configuring non-reservable direct named events • 64
Configuring replication
Replication for the BRM Charging Driver tables • 20
Configuring the SLEE to run the scripts • 32
Modifying the BCD Client startup script • 43
Connection • 4
cxopenTimeout • 35
CORBA • 120
CPU • 43
Creating a customer
Creating a customer • 66
Creating a product and deal for data calls • 101
Creating the BRM product and deal for data calls • 61
Creating a product and deal for named events • 83
About creating products and deals for direct named events • 62
Creating a product and deal for voice calls • 72
Creating the BRM product and deal for voice calls • 60
Creating additional custom fields
Adding custom fields • 51
Creating balance type mappings
About Creating Balance Type Mappings • 45
Creating BRM vouchers • 42, 87
Configuring BRM for vouchers • 65
Creating custom classes
Adding storable classes • 53
Creating header and library files for the custom classes
Creating the custom header and library files • 54
Creating the BRM Domain
Procedure to create a BRM domain • 19
Creating the BRM product and deal for data calls
Creating a product and deal for data calls • 61
C, (continued)

Creating the BRM product and deal for voice calls
Creating a product and deal for voice calls • 60
Creating the custom header and library files
Creating header and library files for the custom classes • 54

D

Data charging
  Charging for a Data Session • 101
  Data session charging flow • 101
 Messages
  data session • 104
 Preconditions for data session charging on BRM • 101
Data session charging flow
  Data charging • 101
DB • 80
defaultOperationTimeout • 36
DefaultUnitType • 26
Diameter • 6
Document Conventions
  Icons • vi
  Typographical conventions • vi
DTMF • 120

E

Editing eserv.config parameters for the BRM Charging Driver
  About editing eserv.config parameters • 21
EDR • 9
ENUM • 27
Example eserv.config file
  Appendix A • 112

F

FOX • 6
Funds expire flow
  A voice call charged against BRM • 78

G

Generating reports
  About statistics and reports • 69
Generating statistics
  About statistics and reports • 68
Generating statistics and reports • 5, 11
Generating the custom JAR file
  Steps to generate JAR file • 55
GPRS • 120
GSM • 24
GUI • 121

H

HLR • 40
HPLMN • 120
HTML • vi

I

Icons
  Document Conventions • vi
IN • vi
INAP • 6
Installing the bcdScp package
  About installing the BRM Charging Driver • 16
Installing the bcdSms package
  About installing the BRM Charging Driver • 15
Installing the BRM Charging Driver • 11
Integrating NCC and BRM
  BRM integration summary • 11
IP • 9
IP address • 9
ISDN • 120
ITU • 23
IVR • 7
IVR BRM voucher recharge flow
  Recharge using BRM vouchers • 87
IVR BRM voucher recharge scenario
  Recharge using BRM vouchers • 87
IVR VWS voucher recharge flow
  Recharge using VWS vouchers • 96
IVR VWS voucher recharge scenario
  Recharge using VWS vouchers • 97

L

latencyStatisticsInterval • 36
loggedNotificationPeriod • 30
lowCreditBufferTime • 30

M

MAP • 6
Mapping NCC currency codes to BRM values
  Configuring bcdActionHandler • 26
Mapping NCC information to BRM fields • 50
  Configuring bcdActionHandler • 26
Mapping NCC sessions to BRM services • 40, 41, 57
  Configuring bcdActionHandler • 22
Mapping opcodes
  Configuring bcdBillingClient • 33
Mapping the location number • 51
  Configuring bcdActionHandler • 27
maxContextIdleTimeSeconds • 36
maxOutstandingRequests • 37
maxPollMilliseconds • 37
maxSelectMicroseconds • 37
M, (continued)

maxTries • 38

Messages

caller hangs up
  A voice call charged against BRM • 76
data session
  Data charging • 104
direct named event for non-reservable event • 62
  Other messages • 108
direct named event reservation • 62
  Other messages • 107
funds expire
  A voice call charged against BRM • 80
  redemption of BRM voucher
Recharge using BRM vouchers • 89
  redemption of VWS voucher against BRM account
Recharge using VWS vouchers • 98
  redemption of VWS voucher using bad PIN
Recharge using VWS vouchers • 99
top up with invalid voucher number or PIN
  Other messages • 107
Messaging Manager • 83, 121, 122
MIN • 57
MM • 121, 122
Modifying the BCD Client startup script
  Configuring the SLEE to run the scripts • 43
  Procedure to modify the script • 43
  Starting the BCD Client processes • 44
Modifying the BRM configuration files
  Steps to modify BRM configuration files • 56
MSC • 74
MSID • 76
MSISDN • 72
MT • 120

N

NccInfoFieldDummyEntry • 30
NccInfoFieldNumber • 31

O

Oracle • ii
ORB • 119
OSA • 120
Other messages
  Messages
    direct named event for non-reservable event • 108
    direct named event reservation • 107
top up with invalid voucher number or PIN • 107
Other NCC components
  Overview of the BRM Charging Driver • 6

P

Parameters
  BearerID • 23
  BrmBadPinEdrActive • 28
  BrmBadPinResourceId • 28
  BrmEdrObjectType • 29
  BRMField • 24
  BrmObjectType • 25
  BRMReqMode • 29
  BrmServicePoid • 25
  cacheTimeout • 29
  clientIdString • 29
  contextOpenTimeout • 35
  defaultOperationTimeout • 36
  DefaultUnitType • 26
  latencyStatisticsInterval • 36
  loggedNotificationPeriod • 30
  lowCreditBufferTime • 30
  maxContextIdleTimeSeconds • 36
  maxOutstandingRequests • 37
  maxPollMilliseconds • 37
  maxSelectMicroseconds • 37
  maxTries • 38
  NccInfoFieldDummyEntry • 30
  NccInfoFieldNumber • 31
  PIN_FLD_LOCATION • 51
  PIN_FLD_NCC_FIELD • 51
  PIN_FLD_NCC_INFO • 50
  PIN_FLD_NCC_NUMBER_OF_EVENTS • 52
  PIN_FLD_NCC_TARIFF_PLAN_ID • 52
  poidPrefix • 31
  recordCMIPAddressInStats • 38
  recordOpcodeInStats • 38
  recordPortInStats • 38
  recoverCmPtrSeconds • 39
  roundingScheme • 31
  ScalingFactor • 24
  serviceDomainInterfaceName • 32
  ServiceKey • 23
  UsedUnitsCumulative • 25

Overview of the BRM Charging Driver

About BRM • 2
About integrating NCC and BRM • 2
BRM and NCC components • 2
BRM Charging Driver features • 6
BRM Charging Driver reports • 5
Other NCC components • 6
The BCD actions shared library • 5
The BCD Client • 4
The BRM Connection Manager • 4
The Portal Communications Module API • 4
Unavailable BRM Charging Driver features • 9
Preconditions for data session charging on BRM • 102
Data charging • 101
Preconditions for recharge using BRM vouchers • 88, 93
Recharge using BRM vouchers • 87
Preconditions for recharge using VWS vouchers • 97
Recharge using VWS vouchers • 95
Preconditions for SMS call scenarios • 84, 85
An SMS call charged against BRM • 83
Preconditions for voice call scenarios • 74, 79, 81
A voice call charged against BRM • 72
Prerequisites
About this Document • v
Procedure to create a BRM domain • 41, 65
Creating the BRM Domain • 19
Procedure to modify the script
Modifying the BCD Client startup script • 43
Recharge using BRM vouchers
IVR BRM voucher recharge flow • 87
IVR BRM voucher recharge scenario • 87
Messages
redemption of BRM voucher • 89
Preconditions for recharge using BRM vouchers • 87
SMS BRM voucher recharge - invalid PIN flow • 93
SMS BRM voucher recharge - invalid PIN scenario • 93
Recharge using VWS vouchers
IVR VWS voucher recharge flow • 96
IVR VWS voucher recharge scenario • 97
Messages
redemption of VWS voucher against BRM account • 98
redemption of VWS voucher using bad PIN • 99
Preconditions for recharge using VWS vouchers • 95
recordCMIPAddressInStats • 38
recordOpcodeInStats • 38
recordPortInStats • 38
recoverCmPtrSeconds • 39
Related documents
About this Document • v
Replication for the BRM Charging Driver tables • 68
Configuring replication • 20
roundingScheme • 31
ScalingFactor • 24
Scope
About this Document • v
SCTP • 119
serviceDomainInterfaceName • 32
ServiceKey • 23
Session • 6
SGML • 120
SGSN • 120
SIM • 66
SIP • 6
SLEE • 4
SMS • vi
SMS BRM voucher recharge - invalid PIN flow
Recharge using BRM vouchers • 93
SMS BRM voucher recharge - invalid PIN scenario
Recharge using BRM vouchers • 93
SMS call delivered successfully flow • 85
An SMS call charged against BRM • 83
SMS call delivered successfully scenario
An SMS call charged against BRM • 84
SMS call that fails permanently flow
An SMS call charged against BRM • 85
SMS call that fails permanently scenario
An SMS call charged against BRM • 85
SN • 121
Specifying operation timeouts • 36
Configuring bcdBillingClient • 33
Starting the BCD Client processes
Modifying the BCD Client startup script • 44
Steps to Configure NCC for the BRM Charging Driver
Summary of NCC Configuration Tasks • 18
Steps to generate JAR file
Generating the custom JAR file • 55
Steps to modify BRM configuration files
Modifying the BRM configuration files • 56
STR • 51
Subscriber not found on BRM flow
A voice call charged against BRM • 81
Subscriber not found on BRM scenario
A voice call charged against BRM • 81
(continued)

Summary of NCC Configuration Tasks
Steps to Configure NCC for the BRM Charging Driver • 18
System Administrator • v

T
TCP • 119
Telco • v
Telecommunications Provider • 122
The BCD actions shared library
   Overview of the BRM Charging Driver • 5
The BCD Client
   Overview of the BRM Charging Driver • 4
The BRM Connection Manager
   Overview of the BRM Charging Driver • 4
The Portal Communications Module API
   Overview of the BRM Charging Driver • 4
Tracking BRM voucher redemptions with bad PINs • 41, 65
   Configuring calls, events, and vouchers • 41
Typographical conventions
   Document Conventions • vi

U
Unavailable BRM Charging Driver features
   Overview of the BRM Charging Driver • 9
Usage scenarios • 50
UsedUnitsCumulative • 25
USSD • 8

V
VLR • 120
Voice Call • 7
voucherPinLength • 32
VWS • 2

X
XMS • 83, 121