## Contents

### Preface

- Audience ....................................................................................................................................................... xiii
- Downloading Oracle Communications Documentation ............................................................................. xiii
- Documentation Accessibility ......................................................................................................................... xiii
- Document Revision History ......................................................................................................................... xiii

### 1 About Payments

- About Payments ........................................................................................................................................ 1-1
  - About BRM-Initiated Payment Processing ............................................................................................... 1-1
    - About Collecting BRM-Initiated Payments ......................................................................................... 1-1
    - Supported BRM-Initiated Payment Methods ...................................................................................... 1-2
  - About Externally Initiated Payment Processing .................................................................................. 1-2
    - About Collecting Externally Initiated Payments .............................................................................. 1-2
    - Supported Externally Initiated Payment Methods ........................................................................... 1-2
- About Payment Methods .......................................................................................................................... 1-3
  - Cash, Check, and Postal Order Payment Methods .............................................................................. 1-3
  - Credit Card Payment Method ................................................................................................................ 1-3
  - Direct Debit Payment Method ................................................................................................................. 1-4
  - Invoice Payment Method ........................................................................................................................ 1-4
  - Prepaid Payment Method ........................................................................................................................ 1-4
  - Nonpaying (Subordinate) Payment Method ......................................................................................... 1-5
  - Undefined Payment Method .................................................................................................................. 1-5
    - About Credit Limits for Undefined Payment Methods ................................................................. 1-5
  - Voucher Payment Method ....................................................................................................................... 1-5
  - Wire Transfer Payment Method ............................................................................................................ 1-5
  - Finding Payment Info ............................................................................................................................... 1-5
- About Payment Attributes .......................................................................................................................... 1-6
  - Account and Bill Number ......................................................................................................................... 1-6
  - Payment Method .................................................................................................................................... 1-6
  - Payment Channel .................................................................................................................................... 1-6
  - Payment Status ....................................................................................................................................... 1-6
  - Batch ID .................................................................................................................................................. 1-6
  - Transaction ID ...................................................................................................................................... 1-6
  - Subtransaction ID ................................................................................................................................. 1-6
- About Validating Payments .......................................................................................................................... 1-7
2 About BRM-Initiated Payment Processing

About BRM-Initiated Payments .................................................................................... 2-1
  About Transactional and Non-transactional Payment Processing .................................. 2-2
About Account Verification for Online Processing ......................................................... 2-2
  Prerequisites ..................................................................................................................... 2-2
  About Action and Response Reason Codes ...................................................................... 2-2
  Supported Transaction Types ............................................................................................ 2-3
About Credit Card Transactions ...................................................................................... 2-4
About Merchant Numbers and Account Identifiers ......................................................... 2-4
  Paymentech Merchant Information .................................................................................. 2-4
  Using More than One Merchant ....................................................................................... 2-4
About Credit Card Validation and Authorization ............................................................. 2-5
  About Credit Card Validation .......................................................................................... 2-5
  About Credit Card Authorization .................................................................................... 2-5
  The Credit Card Validation and Authorization Process .................................................... 2-5
Paymentech and International Transactions ..................................................................... 2-6
About the Paymentech HeartBeat Application .................................................................. 2-6
About Applying Charges Directly to Credit Card Accounts ............................................ 2-7
  General Ledger Impact of Charges .................................................................................. 2-7
3 Configuring BRM-Initiated Payment Processing

Overview of Setting Up BRM-Initiated Payment Processing .................................................. 3-1
Information You Need from Paymentech .................................................................................. 3-2
Information Paymentech Needs from You .................................................................................. 3-2
How Paymentech Manager Handles Electronic Check Processing ........................................ 3-4
About Electronic Check Processing (ECP) Methods ................................................................ 3-4
Payment Formats and Batch Processing ................................................................................... 3-5
Points to Consider ...................................................................................................................... 3-5
Setting Up Merchants and Payment Processors ....................................................................... 3-5
Using More Than One Payment Processor ............................................................................... 3-7
Connecting Your Payment Processor Data Managers to the BRM Database ........................... 3-8
Configuring the Connection Manager for Paymentech ............................................................. 3-8
Enabling Direct Debit Processing ............................................................................................. 3-8
Requiring Additional Protection Against Credit Card Fraud ..................................................... 3-8
    Specifying the Maximum Number of Digits Allowed for CVV2 Verification ......................... 3-9
Disabling Paymentech Real-Time Credit Card Validations ....................................................... 3-10
Configuring the Paymentech Data Manager ............................................................................ 3-10
Specifying Merchant IDs and Merchant Numbers ................................................................... 3-10
Adding Soft Descriptor Information ........................................................................................ 3-11
Handling Concurrent Online Paymentech Requests ............................................................... 3-11
Increasing Registration Speed When Paymentech is Offline .................................................. 3-12
Setting the Connection Timeout Length and Retries ................................................................. 3-13
Specifying the Batch Mode Encryption Key ............................................................................. 3-13
Using the Paymentech HeartBeat Application ......................................................................... 3-14

About Collecting BRM-Initiated Payments .............................................................................. 2-7
When to Run the pin_collect Utility .......................................................................................... 2-8
Increasing Performance of the pin_collect Utility .................................................................... 2-8
Setting the Minimum Amount to Collect .................................................................................. 2-8
About Depositing BRM-Initiated Payments ............................................................................. 2-8
When to Run pin_deposit ........................................................................................................... 2-8
Increasing Performance of the pin_deposit Utility .................................................................... 2-9
About Resolving Failed BRM-Initiated Payment Transactions ................................................ 2-9
When to Run the pin_clean Utility ............................................................................................ 2-9
Example of Running pin_clean ............................................................................................... 2-9
About Recovering BRM-Initiated Payment Transactions .......................................................... 2-10
When to Run the pin_recover Utility ........................................................................................ 2-10
How BRM-Initiated Payment Transactions are Performed ...................................................... 2-10
How BRM Performs Credit Card Charges ............................................................................... 2-13
How BRM Performs Direct Debit Charges .............................................................................. 2-14
    About Paymentech Direct Debit Implementation ................................................................ 2-14
    Creating a Custom Direct Debit Implementation .................................................................. 2-15
How BRM Performs a Batch of Direct Debit Charges ............................................................. 2-15
How BRM Checks the Results of BRM-Initiated Batch Payment Operations ......................... 2-15
How BRM Validates Credit Card and Direct Debit Transactions ............................................. 2-16
How BRM Handles Credit Card Information During Account Creation .................................. 2-16
About Credit Card Fraud Prevention ....................................................................................... 2-17
Troubleshooting HeartBeat Errors ................................................................. 3-14
Changing How BRM Handles Paymentech Address Validation Return Codes ................................................................. 3-15
Handling AVS Validations for International Credit Cards ................................................................. 3-16
Customizing How the Results of Credit Card Transactions are Processed ................................................................. 3-16
Changing How BRM Handles Paymentech Authorization Return Codes ................................................................. 3-18
Testing Paymentech Credit Card Processing................................................................................................. 3-19
   Setting Up the Paymentech Simulator ................................................................................................. 3-20
   Defining the Credit Card Functionality to Test ......................................................................................... 3-20
   Setting Up the Paymentech DM Configuration File for Testing ................................................................. 3-20
   Specifying an IP Address for the Paymentech Simulator ........................................................................... 3-21
   Running the Paymentech Simulators ................................................................................................. 3-21
   Simulating Failed Credit Card Transactions ......................................................................................... 3-21
   Resolving Failed Credit Card Transactions ......................................................................................... 3-22
   About Paymentech Fraud Prevention Using CID and CVV2 ........................................................................... 3-22
About Paymentech Soft Descriptor Credit Card and Checking Statement Information ......................... 3-22
Implementing a Direct Debit Payment Method ................................................................................................. 3-23
   Direct Debit Options ................................................................................................................................. 3-23
   Direct Debit Installation ................................................................................................................................. 3-23
   Direct Debit Components ................................................................................................................................. 3-23
   Implementing a Custom Direct Debit Payment Method ........................................................................... 3-24
      Overview of Adding a Custom Direct Debit Implementation ........................................................................... 3-24
      Creating /payinfo Storable Classes ........................................................................................................ 3-24
      Modifying Customer Center ......................................................................................................................... 3-24
      Creating New Opcodes ................................................................................................................................. 3-25
      Creating New Event Storable Classes ........................................................................................................ 3-25
      Creating a New Data Manager ......................................................................................................................... 3-25
      Updating the /config/payment Storable Object ......................................................................................... 3-25

4 Configuring Payment Channels
   About Payment Channel Information ................................................................................................. 4-1
   Setting Up Payment Channel Information ................................................................................................. 4-1
      Defining Payment Channel Information in BRM ......................................................................................... 4-2
      Mapping Payment Channel IDs for BRM-Initiated Payments ........................................................................... 4-2
      Configuring Payment Channel IDs for Externally Initiated Payments ................................................................. 4-3
   Assigning Payment Channel IDs to Externally Initiated Payments ........................................................................... 4-3

5 Configuring Payment Collection Dates for Automatic Payments
   About Configuring Payment Collection Dates for Automatic Payments ................................................................. 5-1
      About Configurable Payment Collection Dates and On-Demand Billing ................................................................. 5-2
      About Configurable Payment Collection Dates and Delayed Billing ........................................................................... 5-3

6 Configuring Payment Fees
   About Failed Payments ................................................................................................................................. 6-1
   About Payment Fees ................................................................................................................................. 6-2
   Configuring BRM for Payment Fees ........................................................................................................... 6-2
      Defining Payment Attributes for Payment Fees ......................................................................................... 6-3
Defining Reason Codes for Failed Payments ................................................................. 6-3

Creating Payment Fees ............................................................................................... 6-4
  Defining a Payment Fee ............................................................................................ 6-4
  Defining Thresholds for Payment Fees ................................................................. 6-6
  Defining Exemptions from Payment Fees ............................................................ 6-7
  Removing a Payment Fee From an Account Balance .............................................. 6-8

Customizing Payment Fees ........................................................................................ 6-8
  How Payment Fees Are Applied ........................................................................... 6-8
  Customizing Payment Fees .................................................................................... 6-9
  Storing Additional Information with Payment Fees .............................................. 6-10

7 Configuring Payment Incentives

About Payment Incentives ......................................................................................... 7-1
  About Setting Up Payment Incentives .................................................................... 7-2
  About Payment Incentive Processing ................................................................... 7-2
  How Payment Reversals Affect Payment Incentives ............................................ 7-4

Enabling BRM for Payment Incentives ................................................................. 7-4

Creating Payment Incentive Products .................................................................... 7-5
  Defining a Payment Incentive ................................................................................ 7-5

Customizing Payment Incentives ............................................................................. 7-7
  How Payment Incentives Work ............................................................................. 7-7
  How Payment Incentives are Triggered ............................................................... 7-8
  Customizing How to Trigger Payment Incentives ................................................ 7-8
  How Payment Incentives Are Granted .................................................................. 7-9
  Customizing How to Grant Payment Incentives ................................................... 7-10
  How Payment Incentives Are Reversed ............................................................... 7-11
  Manually Reversing a Payment Incentive ............................................................ 7-12

8 Configuring Payment Suspense Manager

About Payment Suspense Manager .......................................................................... 8-1
  Suspended Payment Processing Overview .......................................................... 8-2
  About Setting Up Payment Suspense Manager .................................................... 8-3
  About the Payment Suspension Process ................................................................ 8-4
  About Payment Validation ..................................................................................... 8-6
  About Processing Suspended Payments in a Payment Batch ................................ 8-7
  About Processing Suspended Payments in the BRM Database ............................ 8-7
  About Payment Correction ................................................................................... 8-8
  About Distributing One Payment to Multiple Accounts ....................................... 8-8
  About Allocating an Account-Level Payment to Multiple Bill Units .................. 8-10
  Understanding Payment Recycling ........................................................................ 8-10
  About Original Payments ..................................................................................... 8-11
  About Payment Transfer Direction and Verification ........................................... 8-11
  About Recycling Payments from Suspense ....................................................... 8-12
  About Recycling Payments to Suspense ............................................................. 8-12
  How Payment Reversals Work with Suspense and Recycling ............................. 8-12
  How BRM Tracks Suspended Payments ............................................................... 8-13
How Direct Reversals and Refunds Relate to Suspense ................................................................. 8-16
About Directly Reversing Payments from BRM ............................................................................ 8-16
About Refunding Payments ........................................................................................................... 8-17
About Removing Unallocatable Payments from Suspense ......................................................... 8-17
About Payment Suspense Manager and Client Applications ..................................................... 8-18
Summary of Payment Suspension Guidelines and Restrictions .................................................... 8-20
General Guidelines ...................................................................................................................... 8-20
Suspended Payment Guidelines .................................................................................................. 8-21
Distributed Payment Guidelines .................................................................................................. 8-21
Configuring BRM for Payment Suspense Manager ................................................................. 8-22
Enabling Payment Suspense in BRM ............................................................................................ 8-22
Creating the Payment Suspense Account ....................................................................................... 8-23
Working with Suspense Reason Codes and Action Owner Codes .............................................. 8-23
About the Reasons.locale File ...................................................................................................... 8-24
Loading Reason Codes into the BRM Database ............................................................................. 8-26
Setting Up Permissions for Payment Center ................................................................................. 8-27
About Customizing Payment Suspense Manager ........................................................................ 8-27
About How Payments Are Suspended During Payment Processing ......................................... 8-28
How Payments Are Recycled to and from Suspense .................................................................. 8-28
How Recycled Payments Are Retrieved ...................................................................................... 8-30
How Payments Are Reversed ....................................................................................................... 8-31
How Payments Are Reversed During Recycling ......................................................................... 8-31
How Payments Are Removed As Unallocatable ......................................................................... 8-32
Customizing Payment Suspense Validation ................................................................................. 8-32
Customization Example: Suspending Large Payments ............................................................... 8-33
Customization Example: Threshold for Suspending Payments .................................................... 8-34
Customization Example: Finding Unconfirmed Payments ............................................................. 8-34
Customization Example: Error Handling...................................................................................... 8-34
Default Payment Validation Process .............................................................................................. 8-34
Payment Validation Flags ............................................................................................................... 8-37
Customizing Payment Guidance to Suspense ............................................................................. 8-37
Customizing Payment Failure Reason Codes ................................................................................. 8-37
Customizing Payment Tool ........................................................................................................... 8-38
Adding a Cash Reversal Batch ...................................................................................................... 8-38
Customizing Suspense Criteria for Payment Tool ......................................................................... 8-39
Handling Custom Payment Methods ............................................................................................. 8-39
Adding Multidatabase and Multischema Support in Payment Processing ................................. 8-40
Setting Up Database Links ........................................................................................................... 8-40
Setting Up Schema Qualifications ............................................................................................... 8-41

9 Configuring Top-ups

About Topping Up Accounts ......................................................................................................... 9-1
About Standard Top-ups ............................................................................................................... 9-1
Standard Top-up Payment Methods ............................................................................................ 9-2
About Sponsored Top-ups ............................................................................................................. 9-3
About Sponsored Top-up Groups ................................................................................................ 9-3
About Sponsored Top-up Credit Limits ....................................................................................... 9-4
10 Handling Atypical Payments

Handling Overpayments and Underpayments ................................................................. 10-1
Handling Late or Missed Payments ................................................................................. 10-2
Handling Multiple Payments to the Same Account ......................................................... 10-2
Applying Multiple Payments to an Account Via Payment Gateways .................................... 10-2
Handling Failed Unconfirmed Payments ........................................................................ 10-3
Submitting Failed Unconfirmed Payments with Payment Tool ........................................ 10-4
Requirements for Posting Unconfirmed Payments .......................................................... 10-4
Customizing Unconfirmed Payment Processing ............................................................... 10-5

11 Managing Externally Initiated Payments

About Externally Initiated Payments .............................................................................. 11-1
Supported Externally Initiated Payment Methods .............................................................. 11-1
Processing a Batch of Payments by Using Payment Tool ................................................. 11-2
Who uses Payment Tool? ............................................................................................... 11-3
About Allocating Payments ........................................................................................... 11-4
About Required and Suggested Allocations ................................................................. 11-4
About Allocating Multiple Payments for the Same Bill ................................................. 11-4
Allocating Payments to Bills and Items ........................................................................... 11-4
Allocating an Account-Level Payment to Multiple Bill Units ....................................... 11-5
Allocating Payments Later ............................................................................................ 11-6
Allocating Payments in More Than One Currency ......................................................... 11-6
Allocating Externally Initiated Payments by Due Amount ............................................... 11-6
Finding Bills by Due Amount ......................................................................................... 11-7
About Reversing Payments ........................................................................................... 11-8
Supported Payment Reversal Types ............................................................................... 11-9
Processing a Batch of Payment Reversals by Using Payment Tool ................................... 11-9
About Externally Initiated Refunds .............................................................................. 11-9
Supported Batch Refund Types .................................................................................... 11-10
Processing a Batch of Refunds by Using Payment Tool ................................................ 11-10
Managing Refunds with Your Custom Application ......................................................... 11-10
Managing Non-Validated Batch Entries ....................................................................... 11-11
Processing Lockbox Batches .......................................................................................... 11-11
About the Columns in Batch Windows ............................................................................ 11-12
Importing Batch Data into Payment Tool ...................................................................... 11-14
Handling Overpayments and Underpayments by Using Payment Tool ............................. 11-15
Working with Multiple Currency Types in the Payment Tool ........................................... 11-15
Applying Multiple Payments to the Same Account ......................................................... 11-15
Manually Allocating Account-Level Payments to Accounts with Multiple Bill Units ........... 11-15
Configuring Payment Tool to Lock at the Account Level During Batch Processing ............. 11-16
12 Managing Suspended Payments

About Payment Center ................................................................................................. 12-1
  How BRM Processes Suspended Payments ................................................................. 12-1
About Searching for Payments ..................................................................................... 12-2
  About Payment Center Validation ............................................................................. 12-2
About Allocating Suspended Payments .................................................................... 12-3
  About Deferred Payment Allocation ........................................................................ 12-4
  About Overallocations and Underallocations ............................................................. 12-4
  About Allocating Suspended Payments to Multiple Bill Units .................................. 12-5
  Working with Overpayments and Underpayments .................................................... 12-5
About Allocating Suspended Payments to Multiple Bill Units .................................. 12-4
Configuring Payment Center for Custom Payment Methods ..................................... 12-6
Customizing the Date Format for Payment Center .................................................... 12-7
Improved Performance of Searches for Payment Events in Payment Center ............. 12-8

13 Resolving Failed BRM-Initiated Payment Transactions

About Failed BRM-Initiated Payment Transactions .................................................... 13-1
  How BRM Records Transactions .............................................................................. 13-2
Checking for Transaction Errors ................................................................................ 13-3
Deleting Failed Verifications ...................................................................................... 13-4
Resolving Authorizations ............................................................................................ 13-4
Resolving Refunds ....................................................................................................... 13-5
Resolving Transactions by Using a Request for Response (RFR) File ......................... 13-6
Resubmitting Transactions .......................................................................................... 13-7
Checking for Transactions in Paymentech Send Files .............................................. 13-8
Resolving Payments .................................................................................................... 13-8
Deleting Transactions .................................................................................................. 13-9
Troubleshooting Unresolvable Credit Card Transactions .......................................... 13-10

14 Reserving Resources for Concurrent Network Sessions

About Resource Reservation Manager ....................................................................... 14-1
About Creating Reservations ...................................................................................... 14-2
  About Storing Reservations in IMDB Cache .............................................................. 14-2
  About Storing Reservations in the BRM Database .................................................... 14-2
Setting the Type of Resource Reserved ...................................................................... 14-3
Setting an Expiration Time for the Reservation Request ........................................... 14-3
  Setting the Expiration Time for Prepaid Services ...................................................... 14-3
About Extending a Resource Reservation Amount ..................................................... 14-4
About Extending a Resource Reservation Expiration Time .............................................................. 14-4
About Releasing a Partially Used Reservation ........................................................................... 14-5
About Releasing an Unused Reservation ...................................................................................... 14-5
About Reserving and Releasing Disputed Amounts ................................................................. 14-5
Sending Reservation Requests to the Resource Reservation Manager Opcodes ..................... 14-6
  Creating Reservations ................................................................................................................ 14-6
  Associating a Session with a Reservation .............................................................................. 14-7
  Extending the Reservation Amount ......................................................................................... 14-8
  Finding a Reservation ................................................................................................................. 14-9
  Releasing Reservations ........................................................................................................... 14-10
  Extending the Expiration Time for a Reservation ................................................................. 14-10
Customizing Resource Reservation .......................................................................................... 14-11
  Customizing Resource Reservation Rules ............................................................................... 14-11
  Customizing the Rules for Extending a Reservation ............................................................ 14-11
  Customizing the Rules for Releasing a Reservation .............................................................. 14-12
Installing Resource Reservation Manager .................................................................................. 14-12
  System Requirements ............................................................................................................... 14-12
  Software Requirements .......................................................................................................... 14-12
  Installing Resource Reservation Manager ............................................................................ 14-12
Uninstalling Resource Reservation Manager ............................................................................. 14-14

15 Payment Utilities

  load_pin_ach ............................................................................................................................ 15-2
  pin_balance_transfer .............................................................................................................. 15-4
  pin_clean ................................................................................................................................. 15-6
  pin_collect ............................................................................................................................. 15-8
  pin_deposit ............................................................................................................................. 15-11
  pin_recover ............................................................................................................................ 15-13
This book describes how to collect and manage payments data in Oracle Communications Billing and Revenue Management (BRM).

**Audience**

This document is intended for developers and system administrators.

**Downloading Oracle Communications Documentation**

Product documentation is located on Oracle Technology Network:

http://docs.oracle.com

Additional Oracle Communications documentation is available from Oracle E-Delivery:

https://edelivery.oracle.com

**Documentation Accessibility**

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at


**Access to Oracle Support**

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

**Document Revision History**

The following table lists the revision history for this book.

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E25095-01</td>
<td>March 2013</td>
<td>Initial release.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Made minor formatting and text changes.</td>
</tr>
<tr>
<td>E25095-03</td>
<td>February 2014</td>
<td>Documentation updates for BRM 7.4 Patch Set 20.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Made minor formatting and text changes.</td>
</tr>
</tbody>
</table>
About Payments

This chapter presents an overview of Oracle Communications Billing and Revenue Management (BRM) payments and payment processing.

Before reading this chapter, you should have a basic understanding of BRM concepts. See “Introducing BRM” in BRM Reports.

About Payments

A payment consists of the amount and method by which customers pay their bills. There are several different payment methods available in BRM, depending on the type of payment processing your company performs: BRM-initiated, or externally initiated.

When payments are received in BRM, a payment event is recorded, and the BRM system creates a payment accounts receivable (A/R) item. Payments can be submitted to BRM automatically, by a payment processor, or manually, by using Payment Tool.

You use Customer Center to search for and review customer payments. You can view the payment amount, payment date, payment method, and other payment attributes.

For information on accounts receivable, see “About Accounts Receivable” in BRM Managing Accounts Receivable.

About BRM-Initiated Payment Processing

There are two types of payment processing in BRM: BRM-initiated and externally initiated. All payments received in BRM fall into one of these categories.

BRM-initiated payment processing is triggered by BRM and requires no action from customers. Payments processed in this way are those for which a customer is automatically charged, such as credit card and direct debit payments.

To begin processing such payments, BRM sends a customer’s payment information to your online payment processor. The payment processing service charges the customer’s credit card or checking account, and then BRM automatically allocates the payment and updates the customer’s account balance. Any outstanding payment items are closed, and no A/R management is required by a customer service representative (CSR).

For more information, see “About BRM-Initiated Payment Processing”.

About Collecting BRM-Initiated Payments

You use the pin_collect utility to collect BRM-initiated payments. This utility is typically run automatically by one of the billing scripts. The customer’s credit card or
checking account is charged by your payment processor, and then the payments are sent to the BRM Payment Data Manager (DM).

You can use multiple payment processors and configure multiple Data Managers to handle payment processing.

For more information on collecting BRM-initiated payments, see "About BRM-Initiated Payment Processing".

**Supported BRM-Initiated Payment Methods**

The following BRM-initiated payment methods are supported:

- Credit card
- Debit card
- Direct debit

---

**Note:** Only debit cards that do not require a personal identification number (PIN) are supported in BRM.

---

For information on creating new payment methods, see “Customizing Payment Methods” in *BRM Managing Customers*.

---

**About Externally Initiated Payment Processing**

Externally initiated payments, such as cash or check payments, are triggered by an action from a financial institution. They are usually in response to an invoice that was sent to the customer.

Typically, after such payments are received from a customer, they are sent to a bank. The bank then initiates the payment processing by sending you a list of payments that have been received and deposited. If the bank sends the payment information through a directly integrated third-party service (payment gateway), BRM automatically allocates the payments and updates the customer’s account balance. If the payment information is not sent through a payment gateway, you use Payment Tool to allocate the payment and update the account. See "Managing Externally Initiated Payments" for more information.

**About Collecting Externally Initiated Payments**

You use Payment Tool or a third-party payment application to collect externally initiated payments and post them in BRM.

For information on processing externally initiated payments, see "Managing Externally Initiated Payments".

**Supported Externally Initiated Payment Methods**

The following externally initiated payment methods are supported in BRM:

- Cash
- Check
- Inter-bank transfer
- Postal order
About Payment Methods

A payment method is the mode by which customers pay their bills. The payment method is selected for an account when the account is created, but it can be changed at any time.

**Important:** You can set up multiple payment methods for an account, and assign a different one to each bill unit in an account, but you can use only one payment method per bill unit.

By default, BRM supports the following payment methods:

- Cash, Check, and Postal Order Payment Methods
- Credit Card Payment Method
- Direct Debit Payment Method
- Invoice Payment Method
- Prepaid Payment Method
- Nonpaying (Subordinate) Payment Method
- Undefined Payment Method
- Voucher Payment Method
- Wire Transfer Payment Method

When you add a new payment method, you must update the `config/payment` storable object.

For a complete list of payment methods defined in your BRM system, see the `pin_pymt.h` file.

For information on creating custom payment methods, see “Customizing Payment Methods” in *BRM Managing Customers*.

Cash, Check, and Postal Order Payment Methods

Customers who pay their bills with cash, checks, or postal orders usually have the Invoice payment method defined in their accounts. You handle cash, check, and postal order payments by using Payment Tool.

For more information, see the Payment Tool Help.

Credit Card Payment Method

Credit card payments are BRM-initiated; therefore they are submitted directly to BRM by the Payment DM. Because some credit card payments are made automatically, accounts that pay bills by these methods should always use the balance forward billing type. See “About Accounting Types” in *BRM Configuring and Running Billing*.
When a customer registers for a credit card payment method, BRM attempts to validate the card by default. See “Validating Credit Cards at Registration” in BRM Managing Customers.

When a credit card payment is made, BRM returns a confirmation number that the customer can use to identify the payment. See "About Credit Card Payment Confirmation Numbers".

---

**Important:** Debit cards that are controlled by credit card companies, such as Visa and MasterCard, are supported by BRM; debit cards that require a personal identification number (PIN) to make purchases are not.

---

**Direct Debit Payment Method**

If a customer uses the direct debit payment, the customer’s bank account is debited automatically each billing cycle. Direct debit charges are verified by the bank routing number and the checking account number.

Because some direct debit payments are made automatically, accounts that pay bills by this method should use the balance forward billing type. See “About Accounting Types” in BRM Configuring and Running Billing.

Direct debit payments are BRM-initiated and are therefore submitted directly to BRM by a payment processor.

---

**Note:** BRM supports direct debit only in the United States and Canada.

---

For information on BRM-initiated payment processing, see "About BRM-Initiated Payment Processing".

---

**Invoice Payment Method**

Accounts that use the Invoice payment method pay by check, cash, or other externally initiated payment methods. By default, accounts that use an Invoice payment method receive invoices on a monthly basis.

You use Payment Tool to process invoice-generated payments. For more information, see the Payment Tool Help.

---

**Prepaid Payment Method**

Customers who use the Prepaid payment method pay for service usage in advance. They send check or cash payments and can also pay by using a prepaid voucher.

With prepaid balances, the customer is always expected to have a credit (negative) balance. For example, when an IP telephony customer pays $10 for 100 minutes of usage, the account currency balance is -10 US Dollars. As the customer makes calls, the balance increases until the credit limit (0) is reached.

Accounts that have prepaid balances should use balance forward accounting because payments are made before there is a due amount. (With open item accounting, you are billed only for open items that are due.)

When you run billing, no collection process is performed on prepaid balances because they are paid in advance of billing.
With prepaid balances, there is no credit risk, because customers cannot use services until they have paid for them.

**Nonpaying (Subordinate) Payment Method**
Accounts that use the Paid by parent account payment method are child accounts. These accounts have a nonpaying (subordinate) bill unit. Their bill is paid by their parent accounts. If a child account has two bill units (and thus two bills), one paying and one nonpaying, the child account can pay one bill and the parent account pays the other. See “About Account Groups” in *BRM Concepts*.

**Undefined Payment Method**
Accounts with the Undefined payment method never receive a payment request. When accounting and billing cycles occur, billing utilities that request payments ignore undefined accounts.

Undefined accounts require a login name and password so customers can be authenticated and authorized. You can only assign an undefined payment method to an account during account creation.

Revenue generated from undefined accounts is recorded as general ledger (G/L) data. You typically use undefined accounts for free trial offers. Creating an undefined account allows a customer to register without having to submit a credit card number.

You can also use undefined accounts for testing BRM and for creating CSR accounts.

**About Credit Limits for Undefined Payment Methods**
Even though accounts that use the Undefined payment method are never billed, credit limits are still enforced. This is because credit limits are enforced by rating, not billing. Because undefined accounts are not billed, the balance is never reduced. When the credit limit is reached, the account is inactivated, or the customer cannot log in due to authorization failure.

To avoid this, create an unlimited credit limit for undefined accounts. You can do this in plans designed specifically for those types of accounts or by adjusting account-specific credit limits in Customer Center.

**Voucher Payment Method**
To provide voucher payments for your customers, you must have Voucher Manager and Voucher Administration Center installed.

By default, vouchers are not a standard payment method that can be assigned to an account; they can only be applied manually. When a customer buys a voucher, either a CSR or the customer enters the voucher ID & PIN into a CRM tool such as Customer Center, and BRM validates the voucher and transfers its prepaid resources to the specified account balance.

Voucher payments cannot be handled by the BRM-initiated payment process or by Payment Tool.

For more information, see “About Managing Voucher Inventory” in *BRM Telco Integration*. 
About Payment Attributes

Wire Transfer Payment Method

Wire transfers include any transfer of money from a customer’s bank account to your company, or your company’s payment processor, via automated teller machine (ATM), computer, telephone, or the like.

Customers who pay their bills with wire transfer payments usually have the Invoice payment method defined in their accounts.

You handle wire transfer payments by using Payment Tool. For more information, see Payment Tool Help.

Finding Payment Info

For information about how BRM uses payment methods, see "About Payment Methods".

To find /painful objects that belong to an account, use the PCM_OP_CUST_FIND_PAYINFO opcode.

This opcode is given the account POID and returns the information from the storable /painful object.

About Payment Attributes

This section describes a subset of the payment attributes that help to characterize each payment:

- Account and Bill Number
- Payment Method
- Payment Channel
- Payment Status
- Batch ID
- Transaction ID
- Subtransaction ID

Account and Bill Number

The account number identifies the account in BRM to which the payment is applied. If the account number is missing or incorrect, BRM uses the bill number to guide the payment to the correct account. The bill number therefore identifies a specific bill and determines to which bill the payment applies and is typically allocated.

Payment Method

The payment method identifies how customers pay their bill; for example, by credit card or check. See “About Payment Methods”.

You use Customer Center to manage the payment method for an account. For more information, see the Customer Center Help.

Payment Channel

The payment channel identifies how the payment was sent to your third-party payment processor; for example, via Internet or Voice over IP (VOIP). You can create custom
payment rules based on the payment channel ID; for example, you can grant a payment incentive if customers pay their bills over the Internet.
For more information, see "Configuring Payment Channels".

Payment Status

The payment status identifies the success or failure of a payment when it is received. BRM uses the status code to determine how to process each payment. The status description is displayed in Payment Tool to assist payment clerks in handling externally initiated payment batches.
For more information, see "About Payment Status".

Batch ID

All payments in a payment batch contain the same batch ID. This helps you to identify payments that do not contain a payment ID.

Transaction ID

BRM uses the payment transaction ID to internally manage A/R and to identify payment transactions that occurred between BRM and the payment processor. All BRM-initiated payments, such as credit card and direct debit payments, contain a transaction ID.

BRM identifies failed transactions by keeping a record of each transaction in the BRM database. For information on resolving failed BRM-initiated payment transactions, see "Resolving Failed BRM-Initiated Payment Transactions".

If unconfirmed payment processing is enabled in your BRM system, when a failed unconfirmed payment is received, BRM identifies the original successful payment by using its transaction ID. For more information on unconfirmed payment processing, see "About Unconfirmed Payment Processing".

---

Important: The maximum length of a payment transaction ID is 16 characters for BRM-initiated payments and 30 characters for externally initiated payments. If your company generates transaction IDs by appending characters to the payment batch IDs, make certain that the batch IDs are short enough to still be within the limit after the additional characters are appended. If the payment transaction ID does not comply with the length restrictions, BRM will not generate an ID for the payment. (BRM-initiated payment transaction IDs are restricted by Paymentech processing requirements.)

Subtransaction ID

The subtransaction ID identifies a payment that was reversed due to suspended payment recycling. BRM uses subtransaction IDs internally to track a recycled payment back to its original payment. The subtransaction ID of a recycled payment is the same as the transaction ID of the payment from which it originated.

Payments that have never been recycled have a SUB_TRANS_ID value of NULL.
For information on Payment Suspense Manager and tracking suspended payments, see "How BRM Tracks Suspended Payments".
About Validating Payments

Before payments can be processed and posted by BRM, they must pass validation. Payment validation is initiated automatically, through the payment gateway, or manually, by a payment clerk. When a payment arrives in BRM, the PCM_OP_PYMT_COLLECT opcode calls the PCM_OP_PYMT_VALIDATE_PAYMENT opcode to perform the validation. BRM validates the payment information in the input flist and, in the output flist, indicates whether the opcode successfully performed the validation.

The PIN_FLD_STATUS value returned by the opcodes indicates whether the payment arrived in BRM as successful or failed for financial reasons. If you have the Payment Suspense Manager functionality installed, payments can also arrive as suspended. See "About Payment Suspense Manager" for more information on payment suspense.

- If the payment is successful and passes the validation process, the PIN_FLD_STATUS value is PIN_PYMT_SUCCESS, and BRM posts the payment to the account.
- If the payment meets the validation criteria, but fails for financial reasons, the PIN_FLD_STATUS value is PIN_PYMT_FAILED, and BRM posts the payment to the account as a failed payment.
- If the payment status is marked for suspense, the PIN_FLD_STATUS value is PIN_PYMT_SUSPENSE, and PCM_OP_PYMT_COLLECT calls the PCM_OP_PYMT_POL_SUSPEND_PAYMENT policy opcode to process the payment. For more information about suspended payment validation, see "About Payment Validation".

If an account-level payment is made to an account with multiple bill units, the PCM_OP_PYMT_POL_VALIDATE_PAYMENT opcode validates that the payment is an account-level payment and the account has multiple bill units. If the payment is successful and passes the validation process, it adds the reason ID, PIN_REASON_ID_MBI_DISTRIBUTION_REQD to the PIN_FLD_PAYMENT_REASONS array in the output flist. This reason ID helps in later payment processing by distinguishing a normal payment from an account-level payment made to an account with multiple bill units.

If the payment fails the validation, BRM informs you that the payment cannot be posted. For example, if a payment specifies no account number and no bill number, BRM cannot post the payment. You need to create an exception batch to handle payments that fall into this category.

For more information on payment status, see "About Payment Status".

For information specific to BRM-initiated payment validation, see "Changing How BRM Handles Paymetech Authorization Return Codes".

About Payment Status

BRM uses the PIN_FLD_STATUS field of a payment to validate payments before they are posted in BRM. By default, payments are received with a status of successful, failed, or invalid.

Successful payments are automatically posted to the account to which they belong. The payment amount is removed from the current balance on the account, and any remaining amount is allocated according to your business policies. Failed payments are payments that are declined for financial reasons, such as an overdrawn account or an expired credit card. Invalid payments are payments that cannot be posted correctly for the following reasons:
- The account that the payment applies to is closed.
- Both the account number and the bill number are incorrect and cannot be found in BRM.
- The POID for the account number does not exist in BRM.

If the Payment Suspense Manager feature is enabled, payments can have a status of `suspended`.

Value ranges for the PIN_FLD_STATUS field:

- **Successful payments** have a value \( \geq \text{PIN PYMT SUCCESS} \) and \( < \text{PIN PYMT SUSPENSE} \). The numeric range for successful payments is 0-14.
- **Suspended payments** have a value \( \geq \text{PIN PYMT SUSPENSE} \) and \( < \text{PIN PYMT FAILED} \). This range includes payments that arrive in BRM as `failed` for financial reasons, but that also meet the criteria for suspending a payment. The numeric range for suspended payments is 15-29.
- **Failed payments** have a value \( \geq \text{PIN PYMT FAILED} \) and \( < \text{PIN PYMT STATUS MAX} \). The numeric range for financially failed payments is 30-44.
- Payments with a status \( \geq \text{PIN PYMT STATUS MAX} \) are not supported by BRM.

If a payment is invalid, you must manually fix it before it can be posted in BRM, unless the Payment Suspense Manager feature is enabled. In such cases, invalid payments might be received as suspended.

**Default BRM Status Codes and Descriptions**

The following values described in Table 1–1 are assigned by default:

<table>
<thead>
<tr>
<th>Value</th>
<th>Default code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN_PYMT_SUCCESS</td>
<td>0</td>
<td>The payment was automatically posted to the account to which it belongs. The payment amount is removed from the current balance on the account, and any remaining amount is allocated according to your business policies. Status codes 1 through 14 are configurable. If you have the Payment Suspense Manager feature enabled, they can also be used for payments successfully recycled from suspense to a customer account.</td>
</tr>
<tr>
<td>PIN_PYMT_WRITEOFF_SUCCESS</td>
<td>10</td>
<td>The payment was successfully applied to a written-off account. The write-off reversal feature must be enabled. See “Configuring Write-off and Write-off Reversals” in <em>BRM Managing Accounts Receivable</em>.</td>
</tr>
</tbody>
</table>
About Allocating Payments

Payment allocation is the process of applying a payment toward an account’s open items, balancing all credits and debits, and then closing all balanced items.

Table 1–1 (Cont.) Default BRM Status Codes and Descriptions

<table>
<thead>
<tr>
<th>Value</th>
<th>Default code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN_PYMT_SUSPENSE</td>
<td>15</td>
<td>The payment: • Arrived in BRM as invalid but meets the criteria for suspending a payment. The payment is saved to the payment suspense account.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Arrived in BRM as valid but was manually suspended before or after it posted to the customer account.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This status code is not used for recycled payments. The Payment Suspense Manager feature must be enabled. See “Configuring Payment Suspense Manager”.</td>
</tr>
<tr>
<td>PIN_PYMT_FAILED_SUSPENSE</td>
<td>16</td>
<td>The payment arrived in BRM as failed for financial reasons but meets the criteria for suspending a payment. Failed suspended payments are saved to the payment suspense account.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This status code is used only for payments that originally post to the suspense account. It is not used for recycled payments.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Payment Suspense Manager feature must be enabled. See “Configuring Payment Suspense Manager”.</td>
</tr>
<tr>
<td>PIN_PYMT_RECYCLED_SUSPENSE</td>
<td>17</td>
<td>The payment was generated for an amount that remains in the payment suspense account after an original payment has been partially distributed to customer accounts. You can continue to generate distributed payments until this remaining suspended payment is used up.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>For example, an original payment fails validation and enters BRM as a suspended payment with a payment status of PIN_PYMT_SUSPENSE. The original payment is then partially distributed and a new suspended payment is generated for the remainder. This new suspended payment is assigned a status of PIN_PYMT_RECYCLED_SUSPENSE.</td>
</tr>
<tr>
<td>PIN_PYMT_RETURNED_SUSPENSE</td>
<td>19</td>
<td>The payment was distributed to a customer account from the payment suspense account but was then resuspended.</td>
</tr>
<tr>
<td>PIN_PYMT_FAILED</td>
<td>30</td>
<td>The payment does not comply with the financial practices of your company because, upon collection, it has been dishonored or rejected by the bank. For example, payments can fail due to expired credit cards, incorrect account details, or insufficient funds.</td>
</tr>
<tr>
<td>PIN_PYMT_STATUS_MAX</td>
<td>45</td>
<td>Payments with a value equal to or greater than 45 are not supported by BRM.</td>
</tr>
</tbody>
</table>
Payments are allocated automatically by BRM or manually by a payment clerk. BRM-initiated payments for credit card or direct debit accounts are automatically allocated during the collection process; externally initiated payments for invoice accounts are allocated by a clerk or CSR through Payment Tool or Customer Center.

The allocation level determines how the funds are applied:

- **Account**
  
  When a payment is made at account-level (without specifying bill or item), the payment can be allocated to accounts with single bill unit or multiple bill units.
  
  - **Payment allocated to accounts with single bill unit**: Payment is allocated to the items of the bill unit that contains the default balance group for the account and update the account balance accordingly. When a payment is applied to an account as unallocated, the account balance is updated but the open bills and bill items are not closed. Unallocated payments can be allocated to specific bills and items at any time by using Customer Center or your CRM application.
  
  - **Payment allocated to accounts with multiple bill units**: Payment is distributed to different bill units of the account based on distribution logic implemented in the PCM_OP_PYMT_POL_MBI_DISTRIBUTE policy opcode.

- **Bills**
  
  Payments allocated to one or more bills close the bills and the account balance is updated accordingly.

  By default, bill allocation is determined during payment validation. BRM uses the bill number to find the correct bill. If the bill number is missing or cannot be found, BRM uses the bill amount to find the correct bill. If neither the bill number nor the bill amount can be determined, BRM allocates the payment to the oldest bills first, because they are collected first.

  **Note:** You cannot allocate a payment to a nonpaying (subordinate) bill unit of a child account.

- **Items**
  
  Payments allocated to one or more items close each item and update the balance accordingly. If all items in a bill are closed, the bill is also closed. Item-level allocation also updates the account balance.

  Underpayments and overpayments may also require allocation. See "Handling Overpayments and Underpayments".

  An account-level payment applied to an account with multiple bill units may also require allocation. See "Allocating Account-Level Payments to Multiple Bill Units".

  For information on how to allocate payments in Payment Tool, see "About Allocating Payments".

**Allocating Account-Level Payments to Multiple Bill Units**

If an account-level payment is made to an account having multiple bill units, you can allocate the payment to multiple bill units of the account.
To allocate an account-level payment to multiple bill units of an account, BRM calls the `PCM_OP_PYMT_COLLECT` opcode. `PCM_OP_PYMT_COLLECT` performs the following operations:

1. Opens a transaction.

2. Calls the `PCM_OP_PYMT_VALIDATE_PAYMENT` opcode to validate the payment.

   `PCM_OP_PYMT_VALIDATE_PAYMENT` invokes the `PCM_OP_PYMT_POL_VALIDATE_PAYMENT` policy opcode to validate the payment. This policy opcode:
   a. Checks the appropriate `/config/business_params` objects to find out if Payment Suspense Manager is enabled.
   b. Checks that the payment is an account-level payment and that the account has multiple bill units.
   c. If the payment is successful and passes the validation process, adds the reason ID, `PIN_REASON_ID_MBI_DISTRIBUTION_REQD` to the `PIN_FLD_PAYMENT_REASONS` array in the output flist.

3. Calls the `PCM_OP_PYMT_MBI_DISTRIBUTE` opcode to distribute the payment to multiple bill units. The `PCM_OP_PYMT_MBI_DISTRIBUTE` opcode invokes the `PCM_OP_PYMT_POL_MBI_DISTRIBUTE` policy opcode if the following conditions are true:
   - The `PCM_OP_PYMT_POL_VALIDATE_PAYMENT` policy opcode returns the `PIN_REASON_ID_MBI_DISTRIBUTION_REQD` reason ID.
   - The `PIN_FLD_SELECT_STATUS` field in the input flist of `PCM_OP_PYMT_COLLECT` is *not* `PIN_SELECT_STATUS_MBI_DISTRIBUTED`.

   `PCM_OP_PYMT_POL_MBI_DISTRIBUTE` distributes the payment as per default distribution logic. You can customize how payments are distributed by using the `PCM_OP_PYMT_POL_MBI_DISTRIBUTE` policy opcode. This policy opcode searches for all the open `/bill` objects of all the `/billinfo` objects for the given `/account` object, sorted by the bill due date.

   `PCM_OP_PYMT_POL_MBI_DISTRIBUTE` returns the `PIN_FLD_BILLINFO` array that contains an array of `PIN_FLD_BILLS` having the distributed payment amount for each bill. The `PIN_FLD_BILLINFO` array is added to the output flist of `PCM_OP_PYMT_MBI_DISTRIBUTE`, which is passed to the `PCM_OP_PYMT_SELECT_ITEMS` opcode to get item-level distribution.

4. Calls the `PCM_OP_PYMT_SELECT_ITEMS` opcode for item-level payment distribution for each bill unit.
5. If the reason ID in the output flist of PCM_OP_PYMT_SELECT_ITEMS is PIN_REASON_ID_MBI_DISTRIBUTION_REQD, PCM_OP_PYMT_COLLECT detaches the distribution part from the output flist to use later while recycling payment.

6. If the reason ID in the output flist of PCM_OP_PYMT_SELECT_ITEMS is PIN_REASON_ID_MBI_DISTRIBUTION_REQD and the PIN_FLD_STATUS value is successful (value in the range of 0 to 14), PCM_OP_PYMT_COLLECT changes the status to PIN_PYMT_SUSPENSE to suspend the payment before calling the PCM_OP_PYMT_POL_PRE_COLLECT policy opcode.

7. Calls the PCM_OP_PYMT_POL_PRE_COLLECT policy opcode to perform policy checks before the payment occurs.

8. Calls the PCM_OP_PYMT_POL_SUSPEND_PAYMENT policy opcode to get the suspense account.

9. Calls the PCM_OP_BILL_RCV_PAYMENT opcode to record the payment and create the payment item.

10. Prepares the input flist of the PCM_OP_PYMT_RECYCLE_PAYMENT opcode by using the distribution detached from PCM_OP_PYMT_SELECT_ITEMS.

11. Calls PCM_OP_PYMT_RECYCLE_PAYMENT to distribute the suspended payment to multiple bill units.

12. Prepares the output flist for PCM_OP_PYMT_COLLECT such that the output flist contains all the payment events created as a result of the recycle payment.

About Reversing Payments

Payment reversals are necessary when a payment is recorded in the BRM database, but the payment is not deposited. Reversing the payment allows BRM to treat the payment as if it never happened.

Payments are reversed in BRM for a variety of reasons, the most common of which is that the funds for a payment are never actually deposited: for example, when a check does not clear.

When a payment is reversed, any bills or items previously closed by the payment are reopened, and the payment is deactivated in the BRM system.

There are three types of reversals in BRM:

- **Reversals that you perform directly by using Payment Tool or a third-party application**
- **Reversals that occur indirectly during payment suspense recycling**
- **Reversals that occur when you remove a suspended payment as unallocatable**

The last two reversal types are related to payment suspense. For more information on reversals related to payment suspense, see “How Direct Reversals and Refunds Relate to Suspense”.

**Reversals that you perform directly by using Payment Tool or a third-party application**

*Direct reversals* occur when you use Payment Tool or a third-party application to manually reverse a payment that was recorded in the BRM database but never actually deposited. They remove an active payment completely from the system and reopen any bills and items so the payment can be made again.
Direct reversals are the most frequent type of reversal in BRM, and they occur outside of the suspense and recycling processes. Unlike reversals that occur during recycling, direct reversals are not initiated by the creation of a new recycled payment.

BRM uses the PCM_OP_BILL_REVERSE opcode to process direct reversals. For more information, see "How BRM Reverses Payments".

**Reversals that occur indirectly during payment suspense recycling**

*Indirect reversals* occur when you transfer a suspended payment to a customer account or from a customer account to suspense. With an indirect reversal, the payment is removed from the source account and moved to the target account, resulting in a complete reversal of the payment in the source account so that the payment information can be transferred to the destination account as a recycled payment. If the payment being recycled had been posted in a customer account, any bills and items that were closed due to the payment are reopened.

---

**Note:** To reverse a batch of payments from the BRM system, use Payment Tool. For more information, see "How BRM Reverses Payments".

---

Indirect reversals are assigned a G/L ID of 113, placing the payment amount in a special G/L bucket so that you can keep a separate record of these reversals. For more information on G/L IDs and reversals, see "Working with Suspense Reason Codes and Action Owner Codes".

For more information about reversals due to recycling, see "Understanding Payment Recycling" and "How Payments Are Reversed".

**Reversals that occur when you remove a suspended payment as unallocatable**

If a suspended payment cannot be fixed but you want to track revenue for these payments and generate reports on how much unallocatable revenue you have, you can remove them from suspense as unallocatable. When you do this, BRM reverses the payment in the payment suspense account and assigns it to a special G/L ID bucket. These reversals are rare in BRM. Even though they are part of suspense, they occur outside of the recycling process.

For more information about removing payments as unallocatable, see "About Removing Unallocatable Payments from Suspense".

---

**About Payment Fees**

When payments are received in BRM, they are posted as *successful* or *failed* by default. *Failed payments* are those that have been dishonored or rejected by the bank for financial reasons. For example, payments can fail due to expired credit cards, incorrect account details, and insufficient funds. You can charge customers a fee for sending payments that fail.

For more information, see "Configuring Payment Fees".

---

**About Payment Incentives**

You can provide payment incentives for customers who pay their bills early and in full. Incentives can include currency resources such as monetary credit (for example, a 5% reduction in the monthly bill amount) or non-currency resources (for example, 20 free minutes).
For more information, see "Configuring Payment Incentives".

**About Credit Card Payment Confirmation Numbers**

When a credit card payment completes successfully, BRM returns a confirmation number that the customer can use later to identify the payment. BRM uses the payment item number as the confirmation number.

The PCM_OP_PYMT_COLLECT opcode returns the confirmation number in the PIN_FLD_ITEM_NO output flist field of the PIN_FLD_RESULTS array.

**About Account Top-ups**

The BRM top-up features enable your customers to *top up*: add currency or non-currency resources to: balances in their own accounts or in other customer accounts. For example, a customer can top up her account balance with a $50 payment from her credit card, or she can top up her teenage son’s account with a $50 payment from her account.

BRM supports two types of top-ups:

- Standard top-ups
- Sponsored top-ups

For information, see "About Topping Up Accounts".

**About Payment Suspense Manager**

Payment Suspense Manager is an optional payment feature that lets you more effectively handle the following payment scenarios:

- Payments that fail the BRM validation process.
  
  A payment may fail validation because it contains a missing or incorrect account number and bill number or because the payment attributes do not comply with your custom BRM validation rules. If Payment Suspense Manager is enabled in your BRM system, such payments are saved to a payment suspense account so that your payment processing can continue without having to fix the payments, allocate them manually, or save them to an exception batch.

- Payments that were posted incorrectly to customer accounts.
  
  If a payment was posted incorrectly, you can suspend it and then repost it to the correct account: you do not need to use Payment Tool or a custom CRM application to reverse the payment from the BRM system and then resubmit it in a new payment batch.

- Payments that pay the bills for multiple accounts.
  
  You can subdivide a suspended payment into a list of distributed payments and apply each payment to an individual customer account.

- Account-level payments allocated to accounts with multiple bill units.
  
  You can allocate an account-level payment to multiple bill units of an account. See "Allocating Account-Level Payments to Multiple Bill Units".

For more information about Payment Suspense Manager, see "Configuring Payment Suspense Manager".
About Unconfirmed Payment Processing

By default, BRM requires acknowledgment from a bank or payment processor before BRM-initiated payments are posted. In some cases, such as nontransactional payments, the response from the bank or payment processor does not occur immediately with the request for funds.

To avoid the possible delay in posting payments, you can configure a new payment Data Manager (DM) to post payments immediately: before the funds are confirmed by the bank or payment processor. The DM requires an input flist of payments from BRM and must return the results to BRM in the output flist.

If the payment processor later sends a failure notification (for example, due to insufficient funds or an expired credit card), BRM reverses the initially successful payments and posts the failed payments.

**Important:** Only credit card and debit card payment methods can be posted before they are confirmed.

For information on reversing failed unconfirmed payments, see "Handling Failed Unconfirmed Payments".

About Reversing Account Write-offs During Payment Collection

A write-off is an accounts receivable transaction that removes an uncollectable balance from a customer’s account so it is not considered as an asset for accounting purposes. If a CSR writes off a balance in an account, and later a payment for that account is received, you can reverse the write-off so that the payment can be allocated. You can perform the write-off reversal manually by calling the reversal opcodes, or you can configure BRM to automatically reverse the write-off during the payment collection process.

For more information about write-offs and write-off reversals, see “Configuring Write-offs and Write-off Reversals” in *BRM Managing Accounts Receivable*.

About Payment Processors

You use payment processors to collect payments. The BRM system supports the following payment processors and middleware.

About Automated Clearing Houses

An automated clearing house (ACH) is a secure system that connects banks with financial institutions and enables the electronic transfer of funds to and from checking and savings accounts. For example, you use an ACH to handle check payments and direct debit payments.

When using an ACH, it may take a couple of days before the money is transferred because the ACH waits until it has received confirmation that the transaction was valid.

ACHs do not handle credit card or debit card transactions.

About Credit Card, Debit Card, and Direct Debit Processors

BRM supports the Paymentech credit card and debit card processor.
For information on using the Paymentech processing service with BRM, see "Configuring BRM-Initiated Payment Processing".

**About Payment Gateways**

A *payment gateway* is any third-party payment transaction application that is integrated with BRM. Payment gateways are designed by software development companies for enterprise merchants who already have a BRM billing system in place. A payment gateway provides a way for merchants to receive and process external payment data seamlessly with BRM. A payment gateway receives payment files from multiple sources (such as banks, credit card processors, and automated clearing houses), preprocesses them, formats the files for BRM, and calls BRM opcodes to record the payments in the database.

Before you can load payments into BRM, you must configure your payment gateway with the same payment information you defined in the BRM database.

See also "Applying Multiple Payments to an Account Via Payment Gateways".

**How BRM Collects Payments**

The main payment collection opcode is PCM_OP_PYMT_COLLECT. This opcode communicates with Payment Tool or a payment processor to perform payment collections, refunds, and reversals. It processes payments according to the payment processing type: BRM-initiated, or externally initiated.

- **BRM-Initiated Payment Processing**
- **Externally Initiated Payment Processing**

By default, BRM allocates payments automatically. You can prevent allocation by using the PIN_BILLFLG_DEFER_ALLOCATION flag (0x1000000) in the PCM_OP_PYMT_COLLECT input flist PIN_FLD_CHARGES array.

**BRM-Initiated Payment Processing**

For BRM-initiated payment processing, the normal flow of PCM_OP_PYMT_COLLECT is as follows:

1. Creates a batch checkpoint.
2. Calls the PCM_OP_PYMT_POL_PRE_COLLECT policy opcode, to perform policy checks before the charge, payment, or refund occurs after allocating the PIN_FLD_CHARGE array elements to open items. See “Setting the Minimum Amount to Charge” in *BRM Configuring and Running Billing*.
3. Calls the PCM_OP_PYMT_SELECT_ITEMS opcode, to identify the items to which the payments are applied. See "Selecting the Items to Which Payments Are Applied".
4. Calls the PCM_OP_PYMT_CHARGE opcode, to perform a BRM-initiated payment transaction. This opcode checks the /config/payment object to determine which opcode to call in order to retrieve the payment information from the Payment DM and to create the charge: PCM_OP_PYMT_CHARGE_CC or PCM_OP_PYMT_CHARGE_DD.
5. Calls the PCM_OP_BLL_RCV_PAYMENT opcode. This opcode:
   a. Calls the PCM_OP_ACT_USAGE opcode to update the /event/billing/payment/payment type object.
b. Calls the PCM_OP_BILL_ITEM_TRANSFER opcode to allocate each payment to open items for each bill unit (/billinfo object) specified for the account.

6. Calls the PCM_OP_PYMT_APPLY_FEE opcode, to apply payment fees. If a payment fee is applied, the POID of the payment fee event is added to the output flist for PCM_OP_PYMT_APPLY_FEE.

7. Calls the PCM_OP_PYMT_POL_COLLECT policy opcode, to process the result of a credit card transaction for a specified account bill unit. This opcode sends the charge result to the policy opcode which specifies both the result to be returned to the caller and the actions to be performed on the account’s bill unit.

For BRM-initiated payments, the PCM_OP_PYMT_POL_COLLECT policy opcode calls the PCM_OP_CUST_SET_STATUS opcode if the operation requires a status change.

8. By using checkpoints, finalizes the batch.

### Externally Initiated Payment Processing

For externally initiated payment processing, the normal flow of PCM_OP_PYMT_COLLECT is as follows:

1. Calls the PCM_OP_PYMT_VALIDATE_PAYMENT opcode, to determine the status of the payment records. See "About Validating Payments".

2. Calls the PCM_OP_PYMT_POL_PRE_COLLECT policy opcode, to perform policy checks before the charge, payment, or refund occurs after allocating the PIN_FLD_CHARGE array elements to open items. See “Setting the Minimum Amount to Charge” in BRM Configuring and Running Billing.

3. Calls the PCM_OP_PYMT_SELECT_ITEMS opcode to identify the items to which the payments in the batch are applied. See "Selecting the Items to Which Payments Are Applied".

If necessary, this opcode then calls the PCM_OP_PYMT_POL_OVER_PAYMENT policy opcode and the PCM_OP_PYMT_POL_UNDER_PAYMENT policy opcode, to allocate overpayment and underpayment of funds, respectively.

4. Calls the following standard opcodes:

   - Calls PCM_OP_BLL_RCV_PAYMENT to create the /event/billing/payment/pay_type object.
     
     If the payment status is marked as a failed unconfirmed payment, the PCM_OP_BILL_REVERSE_PAYMENT opcode reverses the unconfirmed successful payments that have a value of PIN_PYMT_FAILED in the PIN_FLD_STATUS field. The PIN_FLD_AMOUNT value in the input flist CHARGES array is set to the value of PIN_FLD_AMOUNT_ORIGINAL_PAYMENT, and the value of PIN_FLD_AMOUNT is set to 0.

   - Calls PCM_OP_PYMT_ITEM_TRANSFER to allocate the payment to open items.

   - Calls PCM_OP_PYMT_APPLY_FEE to record failed payments and apply payment fees. If a payment fee is applied, the POID of the payment fee event is added to the output flist for the PCM_OP_PYMT_APPLY_FEE opcode.

   - If the cease_billing action was received, calls PCM_OP_CUST_SET_BILLINFO to change the status of the bill unit.
5. If mandated by the policy FM, calls the PCM_OP_PYMT_POL_COLLECT policy opcode to perform the following actions listed in Table 1–2, which are based on the payment result:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clear_pending</td>
<td>Apply the payment to reduce the pending receivable of the bill unit by the amount specified.</td>
</tr>
<tr>
<td>set_status</td>
<td>Change the bill unit’s status to that given, using the reasons indicated by the flags.</td>
</tr>
<tr>
<td>issue_refund</td>
<td>Apply a refund by crediting the bill unit with the refund amount specified.</td>
</tr>
<tr>
<td>cease_billing</td>
<td>Discontinue billing an account after a determined period of inactivity by marking the given bill unit as no longer billed.</td>
</tr>
</tbody>
</table>

6. Creates the final batch.

**Selecting the Items to Which Payments Are Applied**

The PCM_OP_PYMT_COLLECT opcode calls the PCM_OP_PYMT_SELECT_ITEMS opcode to identify the items to which the payment is applied. PCM_OP_PYMT_SELECT_ITEMS is also called by Payment Tool.

PCM_OP_PYMT_SELECT_ITEMS does the following:

1. Selects open items based on the input fields and the accounting type of the account. For more information, see "How Items Are Selected for Payments".

2. If the amount passed in on the input flist is not in the primary account currency, PCM_OP_PYMT_SELECT_ITEMS attempts to convert it to the primary account currency.

3. One of the following actions is taken:
   - If called by PCM_OP_PYMT_COLLECT, PCM_OP_PYMT_SELECT_ITEMS returns the PIN_FLD_AMOUNT and the list of selected items.
   - If called by Payment Tool, PCM_OP_PYMT_SELECT_ITEMS compares the sum of the amount due from all the selected items to the PIN_FLD_AMOUNT value to see if the payment is an exact payment, an overpayment, or an underpayment. If it is an overpayment, it calls the PCM_OP_PYMT_POL_OVER_PAYMENT policy opcode. If it is an underpayment, it calls the PCM_OP_PYMT_POL_UNDER_PAYMENT policy opcode.

4. Returns the list of selected items.

**How Items Are Selected for Payments**

Items are selected by PCM_OP_PYMT_SELECT_ITEMS based on the fields in the input flist.

- The contents of the PIN_FLD_BILLS array is examined. For BRM-initiated payments, the contents of the PIN_FLD_BILLINFO array is also examined.
  - The PIN_FLD_BILLS array indicates which /bill storable object PCM_OP_PYMT_SELECT_ITEMS is collecting or paying off. If one or more bills are
How BRM Calculates Payment Collection Dates

By default, BRM-initiated payments, such as payments made by credit card or direct debit, are collected on the date that bills are finalized. Alternatively, you can configure BRM to collect a BRM-initiated payment on the date a bill is due or on a specified number of days before the bill is due. See "Configuring Payment Collection Dates for Automatic Payments".
To support configurable payment collection dates, the PCM_OP_BILL_POL_CALC_PYMT_DUE_T policy opcode calculates a bill’s payment collection date after calculating its due date.

**Note:** Although configurable payment collection dates are used only for BRM-initiated payments, they are calculated and stored for bills associated with *all* payment methods.

To calculate payment collection dates, the PCM_OP_BILL_POL_CALC_PYMT_DUE_T policy opcode performs these tasks:

1. Finds the `/payinfo` object that is linked to the `/billinfo` object with which the bill is associated.
2. Reads the value of the PIN_FLD_PAYMENT_OFFSET field in the `/payinfo` object.
3. Does one of the following:
   - If the value is -1, sets the payment collection date to the date the bill is finalized.
   - If the value is 0, sets the payment collection date to the date the bill is due.
   - If the value is any positive integer (x), sets the payment collection date to x days before the bill is due.

**Note:** If x makes the payment collection date earlier than the date the bill is finalized, the PCM_OP_BILL_POL_CALC_PYMT_DUE_T policy opcode uses the finalization date instead.

4. Stores the payment collection date in the PIN_FLD_COLLECTION_DATE field of the `/billinfo` object.

**Note:** By default, the `pin_collect` utility collects BRM-initiated payments for all bills associated with `/billinfo` objects whose PIN_FLD_COLLECTION_DATE is the day the utility is run or the day before the utility is run.

---

**How BRM Receives Payments**

When payments are received in BRM, they are processed by the PCM_OP_PYMT_COLLECT opcode. To record the payments, PCM_OP_PYMT_COLLECT calls the PCM_OP_BILL_RCV_PAYMENT opcode. This opcode records the payment items and events and updates account balances by calling the PCM_OP_BILL_ITEM_TRANSFER opcode. If there is excess money after paying the item list off, PCM_OP_BILL_RCV_PAYMENT adds that amount to the sub-balance as a credit. If there is no sub-balance, one is created.

- When posting a suspended payment, PCM_OP_BILL_RCV_PAYMENT stores the reason code, action owner code, and original reason code for the payment in the payment event’s PIN_FLD_EVENT_MISC_DETAILS array. The reason codes are passed in from the PCM_OP_PYMT_COLLECT input flist’s PYMT_REASONS array.

The PIN_FLD_EVENT_MISC_DETAILS array uses specific element IDs as follows:
Array Element 0: Stores the reason code used to determine the G/L ID of the payment being moved to the payment suspense account.

Array Element 1: Stores the action owner code for suspended or failed payments.

Array Element 2: Stores the original reason code for failed payments that BRM suspended. This ensures that the reason initially associated with the financial failure is not lost if BRM places the payment in suspense. Element 2 is used only for Payment Suspense Manager; if payment suspense is not enabled, only element 0 will be present.

When the failed payment leaves suspense and PCM_OP_BILL_RCV_PAYMENT creates new payment objects to record the corrected payment, Elements 0 and 1 are no longer needed. The object does not contain the elements 0 and 1 recorded for suspense, and Element 2 becomes the new Element 0.

When processing multiple resource voucher top-ups that include non-currency resources, PCM_OP_BILL_RCV_PAYMENT retrieves the non-currency balance impacts from the PIN_FLD_TOPUPRESOURCE_INFO substruct in the PCM_OP_PYMT_COLLECT input flist and passes them to the PCM_OP_ACT_USAGE opcode so that they are recorded in the corresponding /event/billing/payment/voucher object. See "How BRM Performs Top-ups".

PCM_OP_BILL_RCV_PAYMENT uses the PIN_FLD_SESSION_OBJ field in the input flist to reference the type of session in which the event occurred: either /event/billing/batch/refund or /event/billing/batch/payment, depending on the batch.

How BRM Reverses Payments

Payments are directly reversed from BRM by using Payment Tool. To process a payment reversal batch, the PCM_OP_BILL_REVERSE opcode performs the following operations:

1. Opens a transaction and checks the appropriate /config/business_params objects to find out if Payment Suspense Manager is enabled.

2. Validates that the PIN_FLD_FLAGS field is not present in the input flist or, if the flag is present, that it is not set to PIN_REVERSE_FLAG_REVERSE_AS_UNALLOCATED (1). If the flag is set, the operation fails because payments that were removed already from BRM as unallocatable cannot also be reversed.

3. Checks the PIN_FLD_STATUS field of each reversal associated with the payment being reversed.

4. Calls the PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH opcode, which performs the following operations:
   - Validates that the payment does not have a SUB_TRANS_ID value, and is therefore the original payment. If the payment has a SUB_TRANS_ID value, the operation will fail.
- Finds all distributed payment events that have the same SUB_TRANS_ID value as this payment’s TRANS_ID value, and have not been reversed already due to the recycling process.
- Assigns a reversal TRANS_ID value to each payment returned by the search and populates the reversal flist with each TRANS_ID value.

5. Checks the PIN_FLDOBJECT field of each /event/billing/reversal object associated with the payment being reversed to ensure that no part of the payment has been removed from suspense as unallocatable.

6. Calls PCM_OP_BILL_REVERSE_PAYMENT to reverse the list of payments.
   - If the payment was originally made to a customer account, the list includes any recycled payment generated if the payment was moved into the suspense account after it posted to the customer account.
   - If the payment was originally made to the suspense account, the list contains all payments generated from the original payment, including distributed payments and any payment remaining in the suspense account.

In both cases, the sum of all the payments reversed in this operation should equal the amount of the original payment.

PCM_OP_BILL_REVERSE_PAYMENT verifies that the reversal operation was successful for all of the payments. It uses the PIN_FLDOBJECT_SESSION_OBJ field in the input flist to reference the reversal batch event.

It also checks the /config/business_params object to determine whether payment incentives are enabled. If so, PCM_OP_BILL_REVERSE_PAYMENT calls the PCM_OP_PYMT.Reverse_Incentive_0pcode, which removes the payment incentive trigger in the /billinfo object, thus eliminating the payment incentive.

7. Populates the payment batch with the sum of the reversal flist.

8. Returns the reversal information for all the payments in the PIN_FLDOBJECT_MULTI_ RESULTS array.

The PIN_FLDOBJECT_RESULTS field of the output flist indicates whether the reversal was successful. Direct reversal is not allowed if either of the following conditions is true:
   - The payment has a SUB_TRANS_ID value and, therefore, is not an original payment.
   - PCM_OP_BILL_REVERSE is called from the PCM_OP_PYMT.Recycle_0pcode. In this case, a reversal takes place, but it is not a direct reversal. See "How Payments Are Reversed".

To customize how written off payments are reversed, use the PCM_OP_BILL_POL.Reverse_0pcode. See “Customizing Reversal of Payments Allocated to Written-Off Accounts” in BRM Managing Accounts Receivable.

If you have Payment Suspense Manager enabled, you can reverse only original payments. For information on payment reversals that occur during payment suspense processing, see "How Payments Are Reversed".

For information on processing a payment reversal batch by using Payment Tool, see "About Reversing Payments".

How BRM Refunds Payments

Refunds are accounts receivable actions. For more information, see "Managing Refunds with Your Custom Application".
The PCM_OP_BILL_ITEM_REFUND opcode is used to create a refund item for a /bill or /billinfo object. After the refund items are created, BRM uses the PCM_OP_PYMT_COLLECT opcode to refund the payment amount to the account. For BRM-initiated payments, use the pin_collect utility to refund payments based on the payment method. For externally initiated payments, use Customer Center to perform the refund operation.

How BRM Writes Off Payments

The following opcodes are used for writing off payments:

- PCM_OP_AR_ACCOUNT_WRITEOFF
- PCM_OP_AR_BILL_WRITEOFF
- PCM_OP_AR_ITEM_WRITEOFF

Write-offs and write-off reversals are accounts receivable actions. For more information, see “Writing off Debts and Reversing Write-Offs with Your Custom Application” in BRM Managing Accounts Receivable.

Related Information

For more information about payments and accounts receivable, see:

- "About Accounts Receivable" in BRM Managing Accounts Receivable
- About BRM-Initiated Payment Processing
- Managing Externally Initiated Payments
About BRM-Initiated Payment Processing

This chapter describes BRM-initiated payment processing, and provides information on how to set it up in your Oracle Communications Billing and Revenue Management (BRM) system.

For information on using Paymentech to process credit card or direct debit payments, see “Configuring BRM-Initiated Payment Processing”.

For information about handling externally initiated payments, such as checks or cash, see “Managing Externally Initiated Payments”.

About BRM-Initiated Payments

A BRM-initiated payment is a payment that requires no action from the customer. The customer’s credit card or checking account is charged by your online payment processor or automated clearing house (ACH). The customer’s account balances are updated automatically and any outstanding payments are closed.

BRM supports direct debit of funds by using Paymentech and all of the credit cards supported by Paymentech. It also supports debit cards that do not require a personal identification number (PIN) to perform transactions. See your Paymentech documentation for the latest information.

When you select direct debit payment support during installation, the direct debit payment method is defined in the BRM system, and direct debit is available as a payment method when creating or modifying an account. For more information about direct debit processing, see "Implementing a Direct Debit Payment Method".

Figure 2–1 shows how BRM handles BRM-initiated payments:

Figure 2–1  BRM-Initiated Payments

![Diagram showing BRM-Initiated Payments process]

1. Billing application makes payment request
2. Online payment processor deposits payment
3. Payment is recorded in BRM database
About Transactional and Non-transactional Payment Processing

BRM-initiated payment processing is either transactional or nontransactional:

- **Transactional**: Payments that occur in their entirety in one communication session between BRM and a payment processor. The communication session is bidirectional and occurs entirely online. By default, all BRM-initiated payments are processed this way.

- **Nontransactional**: Payments that occur in two or more communication sessions between BRM and a payment processor. The initial session occurs online when BRM sends a payment request to the payment processor. Subsequent sessions can occur online or offline: for example, processing service personnel may send payment information to you over a network or on CDs. BRM-initiated payments are processed this way whenever the initial communication is interrupted: for example, the processing system may have a system failure, or missing data may need to be supplied manually.

About Account Verification for Online Processing

BRM supports Paymentech’s Account Verification function in the BRM-initiated payment processing for Paymentech. This Account Verification functionality complies with Paymentech Online version 7.4 Revision 3. For information, see the OnLine Processing Format Specification version 7.4 Revision 3 technical specification on the Chase Paymentech library Web site.

Paymentech recommends the use of Account Verification to differentiate credit card/account validation requests from authorization requests. This is because Visa imposes a penalty for all authorization requests that are neither deposited nor reversed.

The Account Verification function supports the following Paymentech methods of payment for credit cards:

- **VI**, the entry used to indicate Visa as the method of payment.
- **MC**, the entry used to indicate MasterCard as the method of payment.

The account verification function supports **EC** as the method of payment entry for direct debit cards in the United States and Canada.

Prerequisites

In order to be able to use Paymentech’s 120-byte batch request/response format, you must complete the required certification with Paymentech for online payment processing. This step should be completed before you allow customers to log in to a production system. You can obtain more information on obtaining the required certification from the ChasePaymentech Web site at:

http://www.chasepaymentech.com/

If you plan to enhance your existing payment processing with Paymentech’s Account Verification function, then before you do so, ensure that all pre-authorized payments are deposited or reversed.

About Action and Response Reason Codes

BRM Paymentech Manager sends the following Action Codes to indicate the type of service Paymentech must perform on the transaction:
When a transaction needs validation only, Paymentech Manager sends the action code LO and a transaction amount is $0.00.

Paymentech in turn, validates this direct debit transaction against an Automatic Clearing House (ACH) eligibility file, Notification of Change (NOC) file, and Electronic Check Processing (ECP) internal negative file for Canadian ECP.

If the account verification is successful for a transaction with an LO action code and the amount set to $0.00, Paymentech responds with a Response Reason Code 101 (Validation passed Paymentech negative file and data edit check).

When the Paymentech Manager needs to verify the direct debit transaction against a third-party negative file for United States ECP, it sends the action code VO and a transaction amount $0.00.

If the account verification of a transaction with an VO action code for an amount set to $0.00 passed the third-party negative file for United States ECP, Paymentech responds with a Response Reason Code 102 (Account verification Passed external negative file).

When the Paymentech Manager needs to verify the account for VISA or MasterCard, it sends the action code VF and a transaction amount $0.00.

If the account verification of a transaction with an VF action code for an amount set to $0.00 is successfully approved, Paymentech responds with a Response Reason Code 104 (No Reason to Decline).

For more information, see the 120-Byte Batch Processing Versions 2.0.0-3.0.0 Rev 2 Addendum in Support of Account Verification technical specification on the Chase Paymentech library Web site.

Supported Transaction Types

BRM supports the following transaction types to describe the circumstances under which a transaction takes place.

- A transaction type 1 indicates a single mail/telephone order transaction where the cardholder is not present at a merchant location and completes the sale through the phone or mail. The transaction is not for recurring services and does not include sales that are processed through an installment plan.

- A transaction type 2 indicates a recurring transaction that represents an arrangement between a cardholder and a merchant where transactions are going to be on a periodic basis.

- A transaction type 7 indicates a channel encrypted transaction between a cardholder and a merchant. The transaction was completed through the internet, using a form of Internet security such as Secure Sockets Layer (SSL) but authentication was not performed.

The BRM Paymentech Manager Configuration file stores "" (blank) as the default value for the transaction type field. Configure the BRM_Home/sys/dm_fusa/pin.conf configuration file to provide the required transaction type.

For information on transaction types in the online processing detail record, see the OnLine Processing Format Specification version 7.4 Revision 3 technical specification on the Chase Paymentech library Web site.
About Credit Card Transactions

To process credit card payments and authorizations, BRM uses two types of credit card transactions: online and batch:

- **Online transaction** - handles a single transaction, for example, a credit card validation or authorization.
  
  To process online transactions for Paymentech, BRM creates a permanent socket connection to the credit card processor.

- **Batch transaction** - handles more than one transaction at a time. You use batch processing when you run the `pin_collect` and `pin_deposit` billing utilities. See "pin_collect" and "pin_deposit".
  
  To process batch transactions for Paymentech, BRM creates a new socket connection for each batch. When the batch has been sent, the connection is closed.

About Merchant Numbers and Account Identifiers

To determine where to deposit your BRM-initiated payments, the payment processors use a merchant ID and a merchant number. Your payment processor assigns your merchant numbers.

If you collect payments in multiple currencies, you need a merchant ID and merchant number pair for each currency.

---

**Note:** The Paymentech DM determines the type of credit card from the credit card number.

---

A *merchant ID* consists of the following parts:

- The *merchant* identifies a type of account. In most cases, all of your accounts use the same merchant. The default merchant is the first merchant listed for the payment processor, which is defined in the `/config/ach` storable object.

- The ISO currency code, such as 840 for US Dollars.

Paymentech Merchant Information

The following example shows merchant IDs and numbers in the Paymentech Data Manager (DM) configuration file. In this example, `mid_ispname_840` is the first merchant ID, and `050505` is the first merchant number.

---

**Note:** There is a merchant ID for each currency, and each merchant ID is mapped to a different merchant number.

---

- `dm_fusa mid_ispname_840 050505`
- `dm_fusa mid_ispname_250 050506`
- `dm_fusa mid_ispname_276 050507`

Using More than One Merchant

You might use more than one merchant if you separate deposits based on payment method; for example, if you deposit payments to a third-party service provider.

If you use multiple merchants, each merchant must be entered in the following files:
About Credit Card Validation and Authorization

Credit card validation validates the customer’s address by checking the ZIP code and street address. Credit card authorization validates the customer’s credit card by checking the card number, expiration date, credit limit, and so forth.

By default, validation occurs during registration, and when a customer changes their credit card number. Authorization occurs at the following times:

- During billing, the pin_collect utility authorizes payments and deposits them. (A credit card deposit is also called a settlement.)
- If there are charges during registration, for example, cycle forward fees or purchase fees.
- When a CSR changes an account payment method by using Customer Center.

About Credit Card Validation

If you use the Address Verification System (AVS), PayMentech gives you a discount for each credit card transaction charge. By default, BRM sends the customer’s name, address, and ZIP code for validation. However, you can get the AVS discount even if you only supply the ZIP code.

**Important:** AVS supports addresses in the United States and Canada only. For information on changing the AVS validation results, see "Changing How BRM Handles PayMentech Address Validation Return Codes".

About Credit Card Authorization

Credit card authorizations made during registration or later by using Customer Center do not charge the customer’s account balance. The payment is recorded in the BRM database, and the balance in the account is adjusted, but the deposit is made later by the pin_deposit utility when you run billing.

The Credit Card Validation and Authorization Process

Credit card validation and authorization occurs in the same transaction, but BRM handles one at a time.

1. BRM sends a validation request along with an authorization to charge $1.00.
The credit card processor returns a validation code and an authorization code. BRM ignores the authorization code, and uses the validation code to determine if the address validation passed. For example, by default an address validation fails if the 5-digit ZIP code is wrong.

**Note:** Since BRM ignores the authorization, the customer is not charged $1.00.

If the address validation fails, the next step, authorization, does not take place.

**Note:** If the Paymentech DM detects non-ASCII data in the address fields during the validation step, the result of the validation request is ignored. This has the same effect as not performing the validation check. This can occur when characters from another language are sent.

2. BRM sends another validation request along with an authorization to charge for an actual amount, for example, a purchase fee.

The credit card processor returns a validation code and an authorization code. This time, BRM ignores the validation code and uses the authorization code to determine if the authorization passed.

You can change how BRM responds to validation and authorization return codes. See the following sections for Paymentech:

- Changing How BRM Handles Paymentech Address Validation Return Codes
- Changing How BRM Handles Paymentech Authorization Return Codes

### Paymentech and International Transactions

You can use Paymentech for credit card processing transfers outside the United States. Paymentech supports different currencies for different credit cards.

The Paymentech Address Verification System (AVS), which verifies customer addresses at time of purchase, is turned off if any non-ASCII encoding is entered in the address fields. You can customize the use of AVS further by changing some policy opcodes. For more information, see "Configuring BRM-Initiated Payment Processing".

Paymentech supports only US and Canadian direct debit accounts. The routing number must be 9 digits and the checking account number can be up to 17 digits.

### About the Paymentech HeartBeat Application

The Paymentech HeartBeat application is a background process that checks the application-to-application connectivity between BRM and Paymentech. When the Paymentech DM (dm_fusa) successfully connects to Paymentech to process BRM-initiated payments, Paymentech acknowledges the secure connection by sending a HeartBeat message. The Paymentech DM responds by sending back a HeartBeat
message to Paymentech to confirm that the connection is working properly and that the expected transactions and data types are being sent.

If Paymentech does not receive a response message from BRM within 120 seconds of sending a request message, or if the response message is incorrect, Paymentech resets the connection to a listen state. BRM handles this as a socket disconnect and recovers accordingly. Errors are written to the `BRM_Home/sys/dm_fusa/dm_fusa.pinlog` file.

**Important:** If BRM stops receiving HeartBeat messages and is in the middle of a transaction, the connection will not disconnect.

The request and response messages should continue for the duration of the connection and are comprised of a unique sequence number and a current timestamp.

For information on initializing the HeartBeat application and troubleshooting errors, see “Using the Paymentech HeartBeat Application”.

### About Applying Charges Directly to Credit Card Accounts

You use Customer Center to apply charges directly to a customer’s credit card.

When you charge a credit card, BRM performs only a credit card authorization. The payment is deposited by running the `pin_deposit` payment utility. By default, you run the `pin_deposit` utility every day by running the `pin_bill_day` script.

See "About Depositing BRM-Initiated Payments".

### General Ledger Impact of Charges

The general ledger impact of charges made directly to credit cards is recorded when the payment is made, not when the charge is initiated. By default, payments are associated with G/L ID 109.

### About Collecting BRM-Initiated Payments

The `pin_collect` utility collects the balance due for bills that are paid by credit card or direct debit. The balance due is calculated by the `pin_bill_accts` utility, and is derived from the total from all bill items, minus amounts that were adjusted, transferred, and so forth.

The `pin_collect` utility collects payments for accounts that had a bill created by the `pin_bill_accts` utility on that day. This is why you run the `pin_bill_accts` utility first.

If you miss a billing day, the `pin_collect` utility still collects on accounts whose billing day was missed. This is because the `pin_bill_accts` utility creates bills for the missed billing days, and the `pin_collect` utility collects payments for those bills.

You can use the `-rebill` option to collect on accounts whose billing day is before the day that you run the `pin_collect` utility.

The `pin_collect` utility performs the credit card authorization and deposits the payment at the same time (as opposed to `pin_deposit`, which only deposits payments that have been pre-authorized.)

The `pin_collect` utility searches for outstanding checkpoint events for the specified `/account` object. If any are found, it flags the result as “Checkpoint Outstanding”.

For information about the `pin_collect` utility syntax, see "pin_collect".
When to Run the pin_collect Utility

Run the pin_collect utility at the following times:

- In the pin_bill_day script for all accounts.
- In the pin_bill_week script with the rebill option on all active accounts.
- In the pin_bill_month script with the rebill option on all closed/inactive accounts.

---

**Important:** When you use multiple payment processors, you run this utility for each payment processor. See “Using More Than One Payment Processor”.

---

You can also run the pin_collect utility manually to rebill accounts from a specific date.

Increasing Performance of the pin_collect Utility

To increase billing performance, you run multiple threads of the pin_collect utility. See “Tuning Billing Performance” in BRM System Administrator’s Guide.

Setting the Minimum Amount to Collect

By default, pin_collect does not collect amounts less than two dollars. To change the minimum amount, see “Specifying the Minimum Payment to Collect” in BRM Configuring and Running Billing.

About Depositing BRM-Initiated Payments

The pin_deposit utility deposits pre-authorized credit card payments into your account.

Pre-authorization occurs in two cases:

- When a customer specifies a credit card payment method.
- When a CSR issues a charge in Customer Center.

The pin_deposit utility searches for all pre-authorized but unpaid credit card transactions made within the past 30 days (from yesterday), and sends the credit card authorization codes and the transaction dates to the credit card processor for depositing.

For information about the pin_deposit utility, see “pin_deposit”.

When to Run pin_deposit

Use the pin_bill_day script to run the pin_deposit utility daily.

You should run the pin_deposit utility daily because VISA authorizations expire in 7 days. (MasterCard authorizations expire in 30 days.) You can deposit pre-authorized payments after the authorization has expired, but the transactions cost more to process.
Important: When you use multiple payment processors, you run this utility for each payment processor. See "Using More Than One Payment Processor".

If performance warrants, you can modify the scope of the search by using the start and end options to change the starting and ending dates of the search.

Increasing Performance of the pin_deposit Utility
To increase billing performance, you run multiple threads of the pin_deposit utility. See "Tuning Billing Performance" in BRM System Administrator’s Guide.

About Resolving Failed BRM-Initiated Payment Transactions
Use the pin_clean utility to troubleshoot failed BRM-initiated payment transactions, such as credit card or direct debit transactions. For more information, see "Resolving Failed BRM-Initiated Payment Transactions".

For information about the pin_clean utility, see "pin_clean".

When to Run the pin_clean Utility
You should run the pin_clean utility every day to look for failed BRM-initiated payment transactions.

Tip: The pin_clean utility writes output to stdout, so you can write a script to run the pin_clean utility daily and write the output to a file.

Example of Running pin_clean
The following example shows a typical pin_clean session:

1. Start the pin_clean utility:

   % pin_clean

2. A summary of checkpoint records appears, followed by a list of options.

   CheckPoint Log Summary:
   PIN_CHARGE_CMD_VERIFY  3
   PIN_CHARGE_CMD_AUTH_ONLY  3
   PIN_CHARGE_CMD_CONDITION 0
   PIN_CHARGE_CMD_DEPOSIT  0
   PIN_CHARGE_CMD_REFUND  2

   Choose function by number:
   1) View PIN_CHARGE_CMD_VERIFY
   2) View PIN_CHARGE_CMD_AUTH_ONLY
   3) View PIN_CHARGE_CMD_CONDITION
   4) View PIN_CHARGE_CMD_DEPOSIT
   5) View PIN_CHARGE_CMD_REFUND
   6) Delete All
   7) Done

   In this example, there are three failed verification transactions, three failed authorization transactions, and two failed refund transactions.
About Recovering BRM-Initiated Payment Transactions

Use the **pin_recover** utility to resolve failed BRM-initiated payment transactions.

When to Run the **pin_recover** Utility

You should run this utility whenever the **pin_clean** utility finds failed transactions. See "Resolving Failed BRM-Initiated Payment Transactions".

For information about the **pin_clean** utility, see "**pin_recover**".

---

**Caution:** If you use a credit card payment processor other than Paymentech, make sure that it uses duplicate transaction detection. If not, using the **pin_recover** utility can cause customers to be billed twice.

---

How BRM-Initiated Payment Transactions are Performed

---

**Note:** The BRM-initiated payment collection utilities (**pin_collect** and **pin_deposit**) process payments for multiple bills associated with multiple **/billinfo** objects. To process externally initiated payments, such as checks and cash payments, for multiple **/billinfo** objects, write custom code that records the deposits you have received and calls **PCM_OP_PYMT_COLLECT**.

---

To perform a BRM-initiated payment transaction, **PCM_OP_PYMT_COLLECT** calls **PCM_OP_PYMT_CHARGE**. The input flist contains a **PIN_FLD_SESSION_OBJ** field, which defines the session in which the event occurred: either a payment batch event or a refund batch event.

The input flist also contains an array of specific operations to perform, so any number of operations can be batched together into a single call. The command is specified within each operation, so a single batch can contain a mixture of different commands.

The billing information for each operation, such as credit card number and expiration date, can be specified as options on the input flist. If it isn’t specified, **PCM_OP_PYMT_CHARGE** retrieves the necessary information from the account storable objects specified within the operations. Once it has all the data, the operations are forwarded to the opcode responsible for processing that payment method, for example, **PCM_OP_PYMT_CHARGE_CC** and **PCM_OP_PYMT_CHARGE_DD** for credit card charges and direct debit charges, respectively.
Each of the commands listed below in Table 2–1 can be executed as an operation. They are also called by Customer Center:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN_CHARGE_CMD_VERIFY</td>
<td>Verify the address information.</td>
</tr>
<tr>
<td>PIN_CHARGE_CMD_AUTH_ONLY</td>
<td>Authorize a charge. The credit limit is decreased on the credit card, but no charge is posted.</td>
</tr>
<tr>
<td>PIN_CHARGE_CMD_CONDITION</td>
<td>Authorize and deposit a charge.</td>
</tr>
<tr>
<td>PIN_CHARGE_CMD_DEPOSIT</td>
<td>Deposit a previously authorized charge.</td>
</tr>
<tr>
<td>PIN_CHARGE_CMD_RECOVER_PAYMENT</td>
<td>Recovers payments by using outstanding checkpoints.</td>
</tr>
<tr>
<td>PIN_CHARGE_CMD_REFUND</td>
<td>Refund a charge.</td>
</tr>
<tr>
<td>PIN_CHARGE_CMD_RESUBMIT</td>
<td>Resubmit the batch of charges.</td>
</tr>
<tr>
<td>PIN_CHARGE_CMD_RFR</td>
<td>For transactional payments, requests the RFR file to retrieve payments.</td>
</tr>
<tr>
<td></td>
<td>For nontransactional payments, reads the RFR file to retrieve payments.</td>
</tr>
</tbody>
</table>

For each operation specified, the result of the operation is stored in the corresponding `/event/billing/charge` storable object.

The result is stored both as a numeric value returned by the payment processor, and as an enumerated result. The enumerated result should generally be used by applications to determine what happened, since its values are independent of which settlement house was used.

Possible result values for an operation are shown in Table 2–2:

<table>
<thead>
<tr>
<th>Input PIN_FLD_RESULT</th>
<th>Output PIN_FLD_RESULT</th>
<th>Output PIN_FLD_DESCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN_CHARGE_RES_PASS</td>
<td>PIN_RESULT_PASS</td>
<td>Validation successful</td>
</tr>
</tbody>
</table>
How BRM-Initiated Payment Transactions are Performed

2-12  BRM Configuring and Collecting Payments

General soft declines are failures that can be retried later with possible success. This includes reasons like insufficient credit limit and other transitory causes. General hard declines are failures that are unlikely to succeed if retried. These include reasons like lost and stolen credit card and chronic payment failures.

Note: By default, account balances are not updated if there is a decline. To update account balances when a decline occurs, you must customize the PCM_OP_PYMT_POL_CHARGE policy opcode.

You can send multiple charges in one call by using the PIN_FLD_CHARGES array on the input flist. This array is designed for batch processing; you just make one call to PCM_OP_PYMT_CHARGE for a whole list of charges (or accounts to charge). These entries on the input flist of this opcode are of special interest:

- The PIN_FLD_POID entry at the top of the input flist is only used for routing; it only requires a correct (user) database number.
- The PIN_FLD_ACCOUNT_OBJ entry is the POID of the account actually being charged (verified) by this element of the PIN_FLD_CHARGES array. In a batch, this POID is presumably different for every element.

If the PCM_OPFLG_CALC_ONLY flag is set, the opcode does not change any fields in the database and does not create an /event/billing/charge object. However, it does provide a charge calculation to the caller by returning the fields that would have been used to create the event object and the charge item.

<table>
<thead>
<tr>
<th>Input PIN_FLD_RESULT</th>
<th>Output PIN_FLD_RESULT</th>
<th>Output PIN_FLD_DESCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN_CHARGE_RES_SRVC_UNAVAIL</td>
<td>PIN_RESULT_PASS</td>
<td>Validation successful</td>
</tr>
<tr>
<td>PIN_CHARGE_RES_FAIL_ADDR_AVS</td>
<td>PIN_RESULT_FAIL</td>
<td>Unable to verify address</td>
</tr>
<tr>
<td>PIN_CHARGE_RES_FAIL_ADDR_LOC</td>
<td>PIN_RESULT_PASS</td>
<td>Street address mismatch</td>
</tr>
<tr>
<td>PIN_CHARGE_RES_FAIL_ADDR_ZIP</td>
<td>PIN_RESULT_FAIL</td>
<td>ZIP code mismatch</td>
</tr>
<tr>
<td>PIN_CHARGE_RES_FAIL_CARD_BAD</td>
<td>PIN_RESULT_FAIL</td>
<td>Credit card number not valid</td>
</tr>
<tr>
<td>PIN_CHARGE_RES_FAILDECL_SOFT</td>
<td>PIN_RESULT_FAIL</td>
<td>General soft decline</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Card failed credit check</td>
</tr>
<tr>
<td>PIN_CHARGE_RES_FAILDECL_HARD</td>
<td>PIN_RESULT_FAIL</td>
<td>General hard decline</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Card failed credit check</td>
</tr>
<tr>
<td>PIN_CHARGE_RES_FAIL_NO_ANS</td>
<td>PIN_RESULT_FAIL</td>
<td>No answer from settlement house</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unable to validate at this time</td>
</tr>
<tr>
<td>PIN_CHARGE_RES_CHECKPOINT</td>
<td>PIN_RESULT_FAIL</td>
<td>Unable to validate at this time</td>
</tr>
<tr>
<td>PIN_CHARGE_RES_CVV_BAD</td>
<td>PIN_RESULT_FAIL</td>
<td>Security ID check failed</td>
</tr>
</tbody>
</table>

Table 2–2  (Cont.) Result Values for Operation

Note: By default, account balances are not updated if there is a decline. To update account balances when a decline occurs, you must customize the PCM_OP_PYMT_POL_CHARGE policy opcode.

You can send multiple charges in one call by using the PIN_FLD_CHARGES array on the input flist. This array is designed for batch processing; you just make one call to PCM_OP_PYMT_CHARGE for a whole list of charges (or accounts to charge). These entries on the input flist of this opcode are of special interest:

- The PIN_FLD_POID entry at the top of the input flist is only used for routing; it only requires a correct (user) database number.
- The PIN_FLD_ACCOUNT_OBJ entry is the POID of the account actually being charged (verified) by this element of the PIN_FLD_CHARGES array. In a batch, this POID is presumably different for every element.

If the PCM_OPFLG_CALC_ONLY flag is set, the opcode does not change any fields in the database and does not create an /event/billing/charge object. However, it does provide a charge calculation to the caller by returning the fields that would have been used to create the event object and the charge item.
About BRM-Initiated Payment Processing

---

**Caution:** Do not set the PCM_OPFLG_CALC_ONLY flag if you are connected to a payment processor, for example Paymentech. This may cause the charge to be sent to the credit card company, even though the charge is not created in BRM. This may result in a double charge on the account.

---

**PCM_OP_PYMT_CHARGE** works as follows:

1. Opens a transaction.
2. Using checkpoints, creates the /event/billing/charge event for each payment.
3. Calls PCM_OP_PYMT_CHARGE_CC or PCM_OP_PYMT_CHARGE_DD, depending on the payment method, to process the charges from the payment DM.
4. Updates the checkpoint in the charge event for each transaction received from the payment DM.
5. For each PIN_FLD_CHARGES in the input flist:
   a. Closes billing for the account.
   b. Creates a payment item and records the account number, bill number, and transaction ID from the input flist, and the fact that the money has been received.
   c. Adds the PIN_FLD_STATUS value in the output flist.
      For failed payments, sets the PIN_FLD_STATUS value to **PIN_PYMT_FAILED**. For successful payments, sets the PIN_FLD_STATUS value to **PIN_PYMT_SUCCESS**.
   d. Calls the PCM_OP_PYMT_POL_CHARGE policy opcode to update the reasons array.
   e. Sends the payment status and the reasons array (PIN_FLD_REASON_ID and PIN_FLD_REASON_DOMAIN_ID) to PCM_OP_PYMT_COLLECT.
   f. For payments with a successful status, applies the charge to the customer’s account. For payments with a failed status, sends the PIN_FLD_STATUS value to the PCM_OP_PYMT_APPLY_FEE to create the payment fees.
6. Closes the transaction.

**How BRM Performs Credit Card Charges**

To perform a credit card charge, PCM_OP_PYMT_CHARGE calls PCM_OP_PYMT_CHARGE_CC.

The PCM_OP_PYMT_CHARGE_CC input flist contains the PIN_FLD_SESSION_OBJ field, which references either the /event/billing/batch/payment or /event/billing/batch/refund object. This determines the batch type being submitted (payment or refund).

The PCM_OP_PYMT_CHARGE_CC input flist contains an array of specific operations to perform, so any number of operations can be batched together into a single call. The command is specified within each operation, so a single batch can contain a mixture of different commands. This opcode supports all commands handled by PCM_OP_PYMT_CHARGE.

The operations are forwarded to the credit card processing Data Manager (for example, the Paymentech DM) for processing.
How BRM-Initiated Payment Transactions are Performed

With most credit card payment services, performing an authorization is much faster than a conditional deposit (authorization plus deposit). Thus, for applications where latency is important, it may be desirable to perform just the authorization step in real-time. BRM daily billing performs the necessary deposits for all outstanding authorizations from the previous day. This removes a significant amount of latency from the real-time process, but still authorizes the charge so it is guaranteed to deposit successfully.

The set of Paymentech return codes handled by BRM is listed in the `BRM_Home/sys/dm_fusa/fusa_codes` file. As explained in "Changing How BRM Handles Paymentech Authorization Return Codes", these codes can be modified. See "How BRM-Initiated Payment Transactions are Performed" for a list of the PIN result codes from BRM-initiated payment transactions.

Unless the PCM_OPFLG_CALC_ONLY flag is specified, PCM_OP_PYMT_CHARGE_CC creates an `/event/billing/charge/cc` storable object for each operation. If an array of operations was specified, then more than one event storable object is created. The event storable objects are created even if the credit card operations cannot be performed.

```
Caution: Do not set the PCM_OPFLG_CALC_ONLY flag if you are connected to a payment processor, for example Paymentech. This may cause the charge to be sent to the credit card company, even though the charge is not created in BRM. This may result in a double charge on the account.
```

How BRM Performs Direct Debit Charges

BRM supports direct debit transactions from customer checking accounts only. To perform a batch of Paymentech direct debit transactions, PCM_OP_PYMT_CHARGE calls PCM_OP_PYMT_CHARGE_DD.

This opcode is used for the Paymentech direct debit implementation shipped with BRM, and is available for you to use in creating a custom direct debit implementation for the bank or payment clearing house you choose.

```
Important: Debit cards that require a personal identification number (PIN) are not supported.
```

About Paymentech Direct Debit Implementation

PCM_OP_PYMT_CHARGE_DDEBIT supports all commands handled by PCM_OP_PYMT_CHARGE.

Unless the PCM_OPFLG_CALC_ONLY flag is specified, this routine creates an `/event/billing/charge/ddebit` storable object for each operation. If an array of operations is specified, then more than one event object is created.

```
Caution: Do not set the PCM_OPFLG_CALC_ONLY flag if you are connected to a payment processor, for example Paymentech. This may cause the charge to be sent to the credit card company, even though the charge is not created in BRM. This may result in a double charge on the account.
```
Creating a Custom Direct Debit Implementation

By default, PCM_OP_PYMT_CHARGE_DD returns direct debit payment information, a charge status, and a payment status to the calling opcode for updating respective events. Effectively this opcode performs a loopback operation that you must change before you can implement direct debit charging. It does not output transaction data for a direct debit clearinghouse. BRM users must create an application to extract the information from the database for a specific direct debit clearinghouse.

For more information on adding a direct debit to your BRM system, see "Implementing a Direct Debit Payment Method".

How BRM Performs a Batch of Direct Debit Charges

To perform a batch of Paymentech direct debit transactions, PCM_OP_PYMT_CHARGE calls PCM_OP_PYMT_CHARGE_DD. The processing is performed on a per-batch basis; only one command and one payment method can exist in the same batch.

PCM_OP_PYMT_CHARGE_DD supports all commands handled by PCM_OP_PYMT_CHARGE, except that it doesn’t create a payment structure and handles transaction charges of $1 only. See "How BRM-Initiated Payment Transactions are Performed" for a list of the PIN result codes from BRM-initiated transactions.

The input flist contains a PIN_FLD_SESSION_OBJ field, which defines the session in which the event occurred: either a payment batch event or a refund batch event (/event/billing/batch/payment or /event/billing/batch/refund).

Unless the PCM_OPFLG_CALC_ONLY flag is specified, this routine creates an /event/billing/charge/dd storable object for each operation. If an array of operations was specified, then more than one event storable object is created. The event storable objects are created even if the direct debit operations cannot be performed.

Caution: Do not set the PCM_OPFLG_CALC_ONLY flag if you are connected to a payment processor, for example Paymentech. This may cause the charge to be sent to the credit card company, even though the charge is not created in BRM. This may result in a double charge on the account.

For more information on credit card handling, see “About the Billing Utilities” and the pin_collect, pin_recover, and pin_deposit sections in particular.

The set of Paymentech return codes handled by BRM is listed in the BRM_Home/sys/dm_fusa/fusa_codes file. As explained in "Changing How BRM Handles Paymentech Authorization Return Codes", these codes can be modified.

How BRM Checks the Results of BRM-Initiated Batch Payment Operations

To check the results of batch payment operations, set the PIN_FLD_COMMAND value in the PCM_OP_PYMT_COLLECT input flist to PIN_CHARGE_CMD_RECOVER PAYMENT. This causes PCM_OP_PYMT_CHARGE to call PCM_OP_PYMT_RECOVER.

PCM_OP_PYMT_RECOVER posts the results of charges for which no information was returned.

PCM_OP_PYMT_RECOVER calls the following opcodes:
To check the results of a batch of credit card charges, PCM_OP_PYMT_RECOVER_CC calls PCM_OP_PYMT_RECOVER_CC. This opcode posts results of credit card charges for which no information was returned. PCM_OP_PYMT_RECOVER_CC is specific to the Paymentech DM.

To check the results of a batch of direct debit charges, PCM_OP_PYMT_RECOVER_CC calls PCM_OP_PYMT_RECOVER_DD. This opcode posts results of direct debit charges for which no information was returned. The results are passed back and used for transaction reconciliation. PCM_OP_PYMT_RECOVER_DD is specific to the Paymentech DM.

How BRM Validates Credit Card and Direct Debit Transactions

To validate credit card and direct debit transactions, use PCM_OP_PYMT_VALIDATE. This opcode is called by PCM_OP_CUST_PREP_CUSTOMER and PCM_OP_CUST_POL_VALID_PAYINFO during registration.

PCM_OP_PYMT_VALIDATE calls the PCM_OP_PYMT_POL_VALIDATE policy opcode to determine the success or failure of a BRM-initiated transaction validation. See "Changing How BRM Handles Paymentech Address Validation Return Codes".

PCM_OP_PYMT_VALIDATE reads the /config/payment storable class to determine the transaction type and the opcode to call, and then calls the appropriate opcode to validate the transaction.

- Credit card transactions: PCM_OP_PYMT_VALIDATE_CC.
  - PCM_OP_PYMT_VALIDATE_CC performs a batch of online credit card validations and applies the validation policy to the results.

- Direct debit transactions: PCM_OP_PYMT_VALIDATE_DD.
  - PCM_OP_PYMT_VALIDATE_DD performs a batch of online direct debit validations and applies the validation policy to the results. PCM_OP_PYMT_VALIDATE_DD calls the appropriate DM to process validations, then returns the results to the Internet.

Note: BRM supports direct debit transactions from checking accounts only.

For both opcodes, the input flist contains an array of specific operations to perform, so any number of operations can be batched together into a single call. The command is specified within each operation, so a single batch can contain different commands. The PIN_FLD_SESSION_OBJ in the input flist is either /event/billing/batch/refund or /event/billing/batch/payment, depending on the batch type: payment or refund.

How BRM Handles Credit Card Information During Account Creation

During the account creation process, PCM_OP_CUST_COMMIT_CUSTOMER passes credit card information to PCM_OP_PYMT_VALIDATE and PCM_OP_PYMT_COLLECT, which collect any credit card payments charged at account creation and validate the credit card information returned by the payment processor.

- PCM_OP_PYMT_COLLECT calls the PCM_OP_PYMT_POL_SPEC_COLLECT policy opcode, which passes the bill unit associated with the payment and returns a list of open items to be paid in the PIN_FLD_ITEMS array.
About Credit Card Fraud Prevention

Paymentech offers an additional fraud prevention option for Visa and American Express transactions. Visa and American Express debit and credit cards have a non-embossed identifier.

- For Visa cards, this field holds the three-digit CVV2 (Card Verification Value 2) ID.
- For American Express, this fields holds the four-digit CID (Card identifier).

Many people who defraud with credit cards know the account numbers and expiration dates of the card, but don’t have the card in their possession and cannot provide the CVV2 or CID number. The Visa CVV2 number is on the back of the card in the signature panel. The American Express CID number is on the front of the card. CSRs can request this information when registering customers.

For security reasons, these numbers are not stored in BRM. PCM_OP_PYMT_VALIDATE omits the PIN_FLD_SECURITY_ID field from input flist of PCM_OP_ACT_USAGE so it isn’t written to the /event/billing/charge/cc object. The result is that the CVV2/CID information is stored in the database with a NULL value.

You can configure BRM to make the identification numbers required or optional when a CSR registers a customer, adds a credit card to an account, or changes a customer’s credit card information. In such cases, PCM_OP_PYMT_VALIDATE_CC sends the information in the PIN_FLD_SECURITY_ID field directly to the payment processor. The PIN_FLD_SECURITY_RESULT field is part of the PIN_FLD_RESULTS element. The value is validated when PCM_OP_VALIDATE_CC issues the PIN_CHARGE_CMD_VERIFY command to the Paymentech DM.

You can also configure Customer Center and BRM to validate the maximum number of digits entered for the CVV2 number. See "Specifying the Maximum Number of Digits Allowed for CVV2 Verification".

For more information, see "Requiring Additional Protection Against Credit Card Fraud".

Important: In the Payments tab, the received payment is displayed as allocated, regardless of whether a bill has been created yet.
Configuring BRM-Initiated Payment Processing

This chapter provides instructions for setting up Oracle Communications Billing and Revenue Management (BRM) credit card and direct debit processing by using Paymentech.

To use Paymentech with BRM, you must install the Paymentech Manager software. Paymentech Manager integrates the Paymentech software with BRM.

**Important:** Paymentech Manager is an optional component, not part of BRM.

Before reading this chapter, read “About Billing” in BRM Configuring and Running Billing for information about how BRM handles billing. See “About BRM-Initiated Payment Processing” for information about BRM-initiated payment processing.

**Note:** The initials FUSA are sometimes used to represent Paymentech in BRM file names. For example, the Paymentech Data Manager (DM) is named dm_fusa.

Unless otherwise noted, the procedures described in this chapter apply to both credit card and direct debit processing with Paymentech. BRM-initiated payments refers to both credit card and direct debit transactions.

For information about creating a custom DM to handle direct debit processing, see "Implementing a Direct Debit Payment Method".

**Overview of Setting Up BRM-Initiated Payment Processing**

To enable BRM-initiated payment processing for Paymentech:

1. Install BRM. For more information, see “Putting Together Your BRM System” in BRM Installation Guide.
2. Install the Paymentech Manager software. See “Installing Paymentech Manager” in BRM Installation Guide.
3. Establish a link with Paymentech.

**Note:** To carry out transactions with Paymentech, you need a leased line, such as a T-1 link or a frame-relay link. Connecting to Paymentech by using a standard modem is not supported.
4. To use the HeartBeat application to monitor connectivity, provide Paymentech with the IP address and port number of the machine running the Paymentech DM. See "Using the Paymentech HeartBeat Application".

5. Edit the payment processor configuration file (BRM_Home/sys/data/pricing/example/pin_ach) and load it into the BRM database. This file specifies merchant names for Paymentech and any other payment processors that you use. See "Setting Up Merchants and Payment Processors".

6. To use direct debit processing, Visa CVV2, or American Express CID for your credit card processing, edit the Connection Manager (CM) configuration file (BRM_Home/sys/cm/pin.conf). See "Configuring the Connection Manager for Paymentech".

7. Specify connection parameters by editing the Paymentech DM configuration file (BRM_Home/sys/dm_fusa/pin.conf). The Paymentech DM (dm_fusa) provides a link between BRM and the Paymentech credit card processor. See "Configuring the Paymentech Data Manager".

8. Start the Paymentech DM. See “Starting and Stopping the BRM System” in BRM System Administrator’s Guide.

9. Test the installation. See "Testing Paymentech Credit Card Processing".

Information You Need from Paymentech

Before setting up BRM-initiated payment processing, you need the following information from Paymentech:

- The IP address and port for the Paymentech online server (the server used for registering customers) and batch server (the server used for handling regular payments).
- The presenter ID and password and the submitter ID and password. See the Paymentech documentation for more information.
- Merchant numbers for each currency you support. See "About Merchant Numbers and Account Identifiers". The same sets of merchant IDs and merchant numbers can be used for both credit card and direct debit.

Even if you already use Paymentech for credit card processing, you need to plan for a setup and testing period for Paymentech direct debit.

Information Paymentech Needs from You

Paymentech needs the following information specific to your BRM software configuration. Table 3–1 lists the defaults suggested by BRM.

<table>
<thead>
<tr>
<th>Paymentech Information</th>
<th>BRM Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>The IP address and port number for the machine running the Paymentech Data Manager (dm_fusa).</td>
<td>None. This is required only if you want to use the Paymentech HeartBeat application, which is integrated with the Paymentech Data Manager. It is required to secure the port. For more information, see &quot;Using the Paymentech HeartBeat Application&quot;.</td>
</tr>
</tbody>
</table>

Is this for an existing Presenter ID (PID)? | No |
### Table 3–1 (Cont.) BRM Default Values for Paymentech

<table>
<thead>
<tr>
<th>Paymentech Information</th>
<th>BRM Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the application software that formats the file?</td>
<td>Written by in-house programmers</td>
</tr>
<tr>
<td>What is the communications software that sends the file?</td>
<td>Customized by the software vendor listed above</td>
</tr>
<tr>
<td>What is the online data communications protocol used to send the online authorization transaction?</td>
<td>TCP/IP Berkley Socket Interface</td>
</tr>
<tr>
<td>What is the batch data communications protocol used to send the batch file?</td>
<td>TCP/IP Berkley Socket Interface</td>
</tr>
<tr>
<td>What online format will you use to send online authorizations?</td>
<td>See “Other Compatible Software” in BRM Installation Guide for information on the format version currently supported by BRM.</td>
</tr>
<tr>
<td>Will you load balance online authorizations (requires leased line) between Paymentech’s data centers, or will you use one data center as primary and one as backup?</td>
<td>Primary and Backup</td>
</tr>
<tr>
<td>What batch format will you use to send batch files?</td>
<td>See “Other Compatible Software” in BRM Installation Guide for information on the format version currently supported by BRM.</td>
</tr>
<tr>
<td>Will you receive the batch reply file by sending an RFR (Request For Response) record or not?</td>
<td>1 Call (IA) - No RFR record sent to pick up reply file.</td>
</tr>
<tr>
<td>Will you send authorizations separately from deposits OR will you send conditional deposits that will result in a deposit upon authorization approval?</td>
<td>Separate authorizations and deposits and conditional deposits.</td>
</tr>
<tr>
<td>What will the average size of your files be once in production? (How many records/transactions?)</td>
<td>None. This number should be based on your company’s projected customer registration growth and billing rate.</td>
</tr>
<tr>
<td>What is the projected submission schedule?</td>
<td>Daily.</td>
</tr>
<tr>
<td>Number of times per day?</td>
<td>Once.</td>
</tr>
</tbody>
</table>
How Paymentech Manager Handles Electronic Check Processing

BRM Paymentech Manager processes all electronic check processing (ECP) transactions in accordance with National Automated Clearing House Association (NACHA) operating rules.

BRM Paymentech Manager provides Account Verification functionality for transactions in batch mode from any custom client to Paymentech. For more on Account Verification functionality and the support for online transactions, see "About Account Verification for Online Processing".

About Electronic Check Processing (ECP) Methods

Valid entries for ECP Authorization Method are:

- **A.** Accounts Receivable. When ECP Authorization Method is set to A, values for Check Serial Number, and Image Reference Number are mandatory.

- **I.** Internet.

- **P.** Point of Purchase. When ECP Authorization Method is set to P, values for Check Serial Number, Terminal City, Terminal State, and Image Reference Number are mandatory.

- **T.** Telephone.

---

**Table 3–1 (Cont.) BRM Default Values for Paymentech**

<table>
<thead>
<tr>
<th>Paymentech information</th>
<th>BRM Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>What Paymentech functionality do you intend to test?</td>
<td>This list reflects a typical pre-paid services company. The first five features require a dedicated connection (leased line)</td>
</tr>
<tr>
<td></td>
<td>• Online Credit Card Authorization</td>
</tr>
<tr>
<td></td>
<td>• Online Electronic Check Processing (ECP) Verification</td>
</tr>
<tr>
<td></td>
<td>• Batch Electronic Check Processing (ECP) Validate &amp; Deposit</td>
</tr>
<tr>
<td></td>
<td>• Batch Deposits</td>
</tr>
<tr>
<td></td>
<td>• Batch Conditional Deposits (for authorization &amp; settlement)</td>
</tr>
<tr>
<td></td>
<td>• Batch Refunds</td>
</tr>
<tr>
<td></td>
<td>• Full AVS (Address Verification Service)</td>
</tr>
<tr>
<td></td>
<td>• Zip only AVS</td>
</tr>
<tr>
<td></td>
<td>• No AVS</td>
</tr>
<tr>
<td></td>
<td>• Visa CVV2</td>
</tr>
<tr>
<td></td>
<td>• Amex CID</td>
</tr>
<tr>
<td></td>
<td>• MasterCard CVC2</td>
</tr>
<tr>
<td></td>
<td>• Discover CID</td>
</tr>
<tr>
<td></td>
<td>• ECI Indicator (also called Transaction Type)</td>
</tr>
<tr>
<td></td>
<td>• International Currencies (please specify)</td>
</tr>
<tr>
<td></td>
<td>• Merchant Descriptor (requires Risk approval)</td>
</tr>
<tr>
<td></td>
<td>• Switch/Solo Cards</td>
</tr>
</tbody>
</table>
■ W. Written.

BRM Paymentech Manager supports these new authorization method values and the corresponding information as required by Paymentech.

If you customize electronic check processing with Paymentech, note that, when ECP Authorization Method is set to A or P:

■ Connection Manager ignores any input you provide in the fields that Paymentech mandates for Check Serial Number, Terminal City, Terminal State, and Image Reference Number.

■ The Check Serial Number, Terminal City, Terminal State, and Image Reference Number mandatory fields are blank in the input BRM Paymentech Data Manager receives from Connection Manager.

In BRM, when you customize electronic check processing for end-to-end payment operations with Paymentech, avoid setting ECP Authorization Method to A or P.

Payment Formats and Batch Processing

Paymentech batch requests/responses support the following.

■ A refund file can be in 120-byte format, even if the corresponding authorization/deposit was completed in 96-byte format.

■ The Request for Response (RFR) header record must be in the same byte format as the response file. That is, to pick up a 96-byte response file, Paymentech expects a 96-byte RFR header record; to pick up a 120-byte response file, Paymentech expects a 120-byte RFR header record.

Points to Consider

Note the following with respect to batch processing functionality complying with Paymentech Batch Version 3.0.0 Revision 4.2:

■ If you use the 120-byte message format, you must complete the certification for batch processing for Paymentech before you allow customers to log in to the production system.

■ For the UK Domestic Maestro (Switch/Solo) card (MOP = SW) with batch processing functionality complying with Paymentech Batch Version 3.0.0 Revision 4.2, Paymentech expects the card issue date and the issue number (if present) in the UK Domestic Maestro extension record.

■ BRM does not support registration of new subscribers with UK Domestic Maestro (Switch/Solo) card type. For already registered subscribers, transactions other than the refund (Action Code = RF) are not supported.

For more information about Paymentech’s 120-byte batch format, view the 120-Byte Batch Processing Format Specification version 3.0.0 - Revision 4.2 document at the Chase Paymentech Web site.

Setting Up Merchants and Payment Processors

**Important:**  This is a mandatory configuration task if you use Paymentech or another payment processor.
You specify merchants and the payment processor vendors that process your BRM-initiated payment transactions for the entire system. You can specify any number of payment processor vendor and merchant pairs.

To specify merchants and vendors, you edit the pin_ach file, then run the load_pin_ach utility to load the contents of the file into the /config/ach object in the BRM database.

---

**Important:** The utility needs a configuration (pin.conf) file in the directory from which you run the utility. See “Creating Configuration Files for BRM Utilities” in BRM System Administrator’s Guide.

---

1. Edit the pin_ach file in BRM_Home/sys/data/pricing/example. The pin_ach file includes examples and instructions.

   **Note:** The default merchant for each payment processor is the first merchant listed for the vendor.

   The file includes this entry for Paymentech:

   ```
   fusa 0.0.0.1 /payment -1 test 0
   ```

   Where:
   - **fusa** is the name of the payment processor.
   - **0.0.0.1 /payment -1** is a routing POID used to identify the database where the payment processor Data Manager (DM) runs. The object type and ID (/payment -1) are not significant.
   - **test** is the merchant name.
     Edit this field to specify your merchant name. This name must match the merchant name entry in the payment processing data manager (DM) configuration file. For example, if the merchant name in the dm_fusa pin.conf file is mid_ispDealer, the merchant name in pin_ach must be ispDealer.
   - **0** is the payment channel ID.
     Edit this field to specify the payment channel ID for each vendor. The channel_id value must match a payment channel ID configured in the /strings object. If a payment does not contain a payment channel ID, a value of 0 is saved with the payment by default, which configures it as Unspecified Payment Channel. For more information, see "Configuring Payment Channels".

   **Caution:** The load_pin_ach utility overwrites existing payment processor and merchant information. If you are updating this information, you cannot load new data only. You must load complete sets of payment processor and merchant entries each time you run the load_pin_ach utility.

2. Save the pin_ach file.

3. Use the following command to run the load_pin_ach utility:

   ```
   load_pin_ach pin_ach
   ```
Setting Up Merchants and Payment Processors

If you are not in the same directory as the `pin_ach` file, include the complete path to the file, for example:

```
load_pin_ach BRM_Home/sys/data/pricing/example/pin_ach
```

For more information, see "load_pin_ach".

To verify that the payment processor and merchant information was loaded, you can display the `/config/ach` object by using the Object Browser, or use the `robj` command with the `testnap` utility. See “Reading an Object and Writing its Contents to a File” in \textit{BRM Developer’s Guide}.

Using More Than One Payment Processor

You can use more than one payment processing Data Manager (DM) at one time to collect and validate payments. To use multiple payment processors, you need to run the following utilities for each payment processor vendor you use:

- `pin_collect`
- `pin_deposit`
- `pin_refund`

These utilities are typically run by the following billing scripts:

- `pin_bill_day`
  By default, this script is scheduled to run `pin_collect`, `pin_deposit`, and `pin_refund`.

- `pin_bill_week`
  By default, this script runs `pin_collect`.

To modify the `pin_bill*` scripts to run the collect, deposit, and refund scripts for every payment processor:

1. Go to the `BRM_Home/bin` directory and open the `pin_bill*` utility in a text editor.
2. Find the entries for the billing utility and add new entries that specify the additional payment processor vendors.

For example, if you use `dm_fusa` and another vendor, find these existing entries:

```
pin_refund -active -pay_type 10003 -vendor fusa IF %ERR% EQU 0 IF %ERRORLEVEL% NEQ 0 SET ERR=%ERRORLEVEL%
```

And add entries to run the utility for each payment processor vendor:

```
pin_refund -active -pay_type 10003 -vendor fusa IF %ERR% EQU 0 IF %ERRORLEVEL% NEQ 0 SET ERR=%ERRORLEVEL%
pin_refund -active -pay_type 10003 -vendor new_vendor
```

Configuring BRM-Initiated Payment Processing   3-7
Connecting Your Payment Processor Data Managers to the BRM Database

Your payment processor data managers (DMs) require a connection to the BRM database.

1. Open the Connection Manager (CM) configuration file (`BRM_Home/sys/cm/pin.conf`).
2. Add a `dm_pointer` entry for each payment processor vendor DM. Use the following format:
   `- cm dm_pointer database_number ip host_name port_number`
3. Stop and restart the CM. See “Starting and Stopping the BRM System” in `BRM System Administrator’s Guide`.

Configuring the Connection Manager for Paymentech

The following procedures involve configuring your Connection Manager (CM).

Enabling Direct Debit Processing

Depending on the choices made during installation, the settings for direct debit might not be turned on. (Turned off is the default.)

1. Open the Connection Manager (CM) configuration file (`BRM_Home/sys/cm/pin.conf`).
2. Change the value of the direct debit entries according to the instructions in the file.
   
   For example:
   `- fm_pymt_pol dd_validate 1`
   `- fm_pymt_pol dd_revalidation_interval 3600`
   `- fm_pymt_pol dd_collect 1`
3. Save the file.

You do not need to restart the CM to enable this entry.

Requiring Additional Protection Against Credit Card Fraud

Paymentech offers additional fraud prevention using Visa CVV2 numbers and American Express CID numbers.
CSRs can request this information when they use Customer Center to register customers or update credit card information in customer accounts. By default, the CVV2 and CID numbers are considered to be optional when CSRs add or change a customer’s credit card information. To require the CVV2 or CID number as part of customer registration, change the following fields in the Connection Manager (CM) configuration file (BRM_Home/sys/cm/pin.conf).

Important: For security reasons, the CVV2 and CID numbers are stored in BRM with a NULL value. If you have the cvv2_required entry enabled, the information is sent directly to Paymentech for validation without being stored in the database. (Even if your CM doesn’t require this additional fraud prevention, Paymentech still accepts the information if it is sent.)

To require Visa CVV2:

1. Open the Connection Manager (CM) configuration file (BRM_Home/sys/cm/pin.conf).
2. Change the value in the following entry from the default, 0, to 1:
   - fm_pymt_pol cvv2_required 1
3. Save the file.

You do not need to restart the CM to enable this entry.

To require American Express CID:

1. Open the Connection Manager (CM) configuration file (BRM_Home/sys/cm/pin.conf).
2. Change the value in the following entry from the default, 0, to 1:
   - fm_pymt_pol cid_required 1
3. Save the file.

You do not need to restart the CM to enable this entry.

If these entries are missing from the CM configuration file, CVV2 and CID are not required for customer registration. For more information on how BRM handles these numbers, see “CVV2/CID Fraud Prevention Functionality” in BRM Managing Customers.

Specifying the Maximum Number of Digits Allowed for CVV2 Verification

By default, Customer Center and BRM accept a maximum of three CVV2 digits when validating a customer’s credit card.

To change the maximum number of CVV2 digits that can be entered, perform the following:

- For Customer Center: Use the Configurator application provided with Customer Center SDK to modify the maximum number of CCV2 digits allowed by Customer Center. You enter the information in the CVV2 Number - maximum digits allowed field of the Payment Configurator.

- For BRM: Customize the PCM_OP_CUST_POL_VALID_PAYINFO policy opcode to validate the number of digits passed in the PIN_FLD_SECURITY_ID input flist field of the PIN_FLD_CC_INFO array.
**Disabling Paymentech Real-Time Credit Card Validations**

During account creation, credit card information is validated in two phases:

1. BRM calls the PCM_OP_PYMT_VALIDATE opcode to do basic validations, including number sequence validations against the numbers defined in the `pin_cc_pattern.h` file.
2. BRM sends the credit card information to Paymentech, which does a real-time validation of the credit card number, expiration date, and so on.

For example, say you enter credit card information for an account that has a Visa number pattern and a future expiration date; however, the card is actually expired. The BRM validation will pass successfully because the credit card number sequence is valid but the Paymentech validation will fail.

To disable the real-time Paymentech credit card validation:

1. Open the Connection Manager (CM) configuration file (`BRM_Home/sys/cm/pin.conf`).
2. Change the value in the following entry to 0:
   ```
   - fm_pymt_pol cc_validate 0
   ```
3. Save the file.
4. Stop and restart the CM. See “Starting and Stopping the BRM System” in *BRM System Administrator’s Guide*.

**Configuring the Paymentech Data Manager**

Follow these procedures to configure the Paymentech Data Manager (DM):

- Specifying Merchant IDs and Merchant Numbers
- Adding Soft Descriptor Information
- Handling Concurrent Online Paymentech Requests
- Increasing Registration Speed When Paymentech is Offline
- Setting the Connection Timeout Length and Retries
- Specifying the Batch Mode Encryption Key
- Using the Paymentech HeartBeat Application

**Specifying Merchant IDs and Merchant Numbers**

For information about merchant numbers, see "About Merchant Numbers and Account Identifiers".

1. Open the Paymentech DM configuration file (`BRM_Home/sys/dm_fusa/pin.conf`).
2. Change the merchant entry. (merchant must match the entry described in “Setting Up Merchants and Payment Processors”.) Use this syntax:
   ```
   - dm_fusa mid_merchant_ISOcurrency# Merchant_number
   ```
   For example:
   ```
   - dm_fusa mid_ispname_840 050505
   ```
3. Save and close the file.
4. Stop and restart the Paymentech DM. See “Starting and Stopping the BRM System” in BRM System Administrator’s Guide.

Adding Soft Descriptor Information

You can help your customers understand their credit card and checking account statements by showing your DBA, a recognizable product name, and a customer service number for questions. To add this information:

1. Open the Paymentech DM configuration file (BRM_Home/sys/dm_fusa/pin.conf).
2. Turn on soft descriptors by changing the descriptor flag value to 1:
   
   - dm_fusa sd_descriptor_flag 1

3. Change the other related entries according to the instructions in the file.
4. Save and close the file.
5. Stop and restart the Paymentech DM. See “Starting and Stopping the BRM System” in BRM System Administrator’s Guide.

For an overview of soft descriptors, see “About Paymentech Soft Descriptor Credit Card and Checking Statement Information” and the Paymentech soft descriptor specifications.

To create multiple DBAs, product names, and phone number entries, you need to customize the PCM_OP_PYMT_POL_PRE_COLLECT policy opcode.

Handling Concurrent Online Paymentech Requests

You can increase billing performance by using the fusamux program. Since Paymentech allows only a single connection per customer, the fusamux program takes multiple DM backends and bundles them into a single connection. This enables BRM to process multiple transactions and send them to Paymentech in a single connection as shown in Figure 3–1.

Without fusamux, the Paymentech DM connects directly to Paymentech. When you use fusamux, the Paymentech DM connects to the fusamux application, which in turn connects to Paymentech. When you use fusamux, you need to change entries in the Paymentech DM to point to fusamux instead of pointing to Paymentech.

Figure 3–1  Paymentech Requests with or without Fusamux

Without fusamux

Paymentech DM → Paymentech

With fusamux

Paymentech DM → Fusamux → Fusamux → Paymentech

To configure the fusamux daemon:

1. Open the Paymentech DM configuration file (BRM_Home/sys/dm_fusa/pin.conf).
2. Edit the fusamux entries:
Configuring the Paymentech Data Manager

- Set the `fusamux online_port` and `fusamux online_srvr` entries to point to the Paymentech online server IP address and port number.
- Set the `fusamux_port` entry to the port on which the `fusamux` daemon listens.
- Set the `dm_fusa online_port` entry to the port on which `fusamux` listens.
- Set the `dm_fusa online_srvr` entry to point to the `fusamux` IP address.
- Set the `dm_fusa qm_n_be` entry to a number between 4 and 8.

3. Save the file.

4. Stop and restart the Paymentech DM. See “Starting and Stopping the BRM System” in BRM System Administrator’s Guide.

Increasing Registration Speed When Paymentech is Offline

If you know that your connection to Paymentech will be offline for a long time, you can speed up registration by not allowing timeouts. Instead, a connection to Paymentech results in a “no answer” error immediately. By default, registration can occur even though there is a “no answer” error. Also, a “no answer” error does not create checkpoint records, so you don’t have to resolve the transaction.

You can also create accounts that use the Undefined checkpoint records. For more information, see “Allowing Registration Without Credit Card Validation” in BRM Managing Customers.

---

**Note:** If you use this option, you can’t process credit card transactions by using the `pin_collect` or `pin_deposit` utilities. You need to wait and run billing when the Paymentech connection is restored.

---

**Tip:** If the credit card payment service is not available, and you still want to register customers, you need to isolate those accounts for later credit card authorization. Modify the PCM_OP_PYMT_POL_VALIDATE policy source file either to save a list of permissive registrations or to send email to the system administrator. Alternatively, you can write a simple application to periodically check accounts and flag the ones that have been registered without verification.

1. Open the Paymentech DM configuration file (`BRM_Home/sys/dm_fusa/pin.conf`).
2. Edit the `online_proto` entry:
   - Enter `linkdown` to disable timeouts and report “no answer” for all connections.
   - Enter `socket` to allow the connection to function normally.
3. Edit the `batch_proto` entry:
   - Enter `linkdown` to disable timeouts and report “no answer” for all connections.
   - Enter `socket` to allow the connection to function normally.
4. Save the file.
5. Stop and restart the Paymentech DM. See “Starting and Stopping the BRM System” in BRM System Administrator’s Guide.

**Setting the Connection Timeout Length and Retries**

If you have problems connecting to Paymentech, increase the connection timeout length and number of retries:

1. Open the Paymentech DM configuration file (BRM_Home/sys/dm_fusa./pin.conf).
2. Edit the connect_retrys entry. The default is 2. You can enter any number.
3. Change the timeout value for online attempts and for batch attempts separately:
   - To change the timeout value for online attempts, edit the fusa_timeout entry.
   - To change the timeout value for batch attempts, edit the fusa_batch_timeout entry.

   The default for both entries is 600 seconds.

4. Save the file.

5. Stop and restart the Paymentech DM. See “Starting and Stopping the BRM System” in BRM System Administrator’s Guide.

**Specifying the Batch Mode Encryption Key**

If you process multiple credit card transactions at a time, batch mode processing uses temporary send and receive files to capture records to and from Paymentech. To prevent any misuse of the temporary batch files, sensitive data like the credit card and security code is encrypted.

You specify the encryption method and key in the Paymentech configuration file. You can encrypt fields by using Advanced Encryption Standard (AES) encryption or Message-Digest algorithm 5 (MD5) encryption. For more information about these encryption methods, see "About AES Encryption" and “About MD5 Encryption” in BRM Developer’s Guide.

```
Tip: You should change the encryption key regularly. Before changing the encryption key, make sure all pin_recover operations using the -rfr and -resubmit parameters that depend on the current encryption key are completed.
```

To specify the encryption key:

1. Open the Paymentech DM configuration file (BRM_Home/sys/dm_fusa/pin.conf).
2. Search for the following line:
   ```
   - crypt md5
   ```
3. Change this entry to the following:
   ```
   - crypt md5| libpin_crypt4qm.so "Abracadabra dabracadabracada!!!!"
   ```
4. Add the following line for the encrypted AES key:
   ```
   - crypt aes| libpin_crypt_aes4qm.so
   "#aes|0D5E11BFDD97D2769D9B0DBFBD1BBF7E90612D45947BDF75C0FBF843E8AC844B6C3E95B08CADD505ACFCF44DA72C5FF6"
   ```
The file now contains the following entries:

- crypt md5| libpin_crypt4qm.so "Abracadabra dabracaabraasada!!!"
- crypt aes| libpin_crypt_aes4qm.so
 "&aes|0D5E11BFDD97D279D980DBFBD1BFF7E90612D45947BDF75CF0FB5A9E8AC84A6C3E95B08CADD505ACFCF44D72C5FF6"

5. Do one of the following:
   - If AES is the required encryption method, comment out the line containing the crypt entry for MD5 only. For example:
     ```
     #- crypt md5| libpin_crypt4qm.so "Abracadabra dabracaabraasada!!!"
     ```
   - If MD5 is the required encryption method, leave both lines uncommented. When both crypt entries are enabled in the pin.conf file, the Paymentech DM uses the MD5 encryption key.

6. Save and close the file.

7. Stop and restart the Paymentech DM. See “Starting and Stopping the BRM System” in BRM System Administrator’s Guide.

Using the Paymentech HeartBeat Application

The Paymentech Data Manager is integrated with the Paymentech HeartBeat application and runs the process by default after it is initialized by Paymentech.

To initialize the HeartBeat application, provide Paymentech with the IP address and port number of the machine running the Paymentech Data Manager (dm_fusa). The HeartBeat application will run automatically each time you process BRM-initiated payments.

For more information, see "About the Paymentech HeartBeat Application".

Troubleshooting HeartBeat Errors

The PIN_LOG_DIR/dm_fusa/dm_fusa.pinlog file records the Paymentech HeartBeat request and response message pairs throughout the connection with Paymentech. Each request message must have a response message within 120 seconds to avoid a socket disconnect.

The following entry is a typical HeartBeat request and response pair:

Received (20) chars: Heartbeat request [HO19999999813123258^M]
Sending Heartbeat response [HI199999998131233300^M]

If these entries are missing from the dm_fusa.pinlog file or are not continuous for the duration of the connection with Paymentech, the payment processing company should call Paymentech to troubleshoot why the connection was lost or the HeartBeat application was not enabled from their end.

---

**Note:** If a connection is made between the DM and Paymentech, and Paymentech does not initiate the HeartBeat messages, BRM assumes there is no HeartBeat application support and continues with payment processing as normal.

---

If an error occurs with the HeartBeat application during payment simulation, an error message similar to the following is written to the BRM_Home/apps/fusa_server/answer_s.pinlog file:
In order for this message to be logged, the payment processing simulator configuration file (BRM_Home/apps/fusa_server/pin.conf) must contain the following entries:

- answer_s loglevel 3
- answer_s logfile answer_s.pinlog

For more information, see "Defining the Credit Card Functionality to Test".

If a socket disconnect occurs with the payment processing simulator and no online transactions are occurring, errors similar to the following are written to the answer_s.pinlog file:

```
Socket read error in dm_fusa_respond_heartbeat() recv() returned (0)
Error(7) processing heartbeat monitor fd(5)
```

## Changing How BRM Handles Paymentech Address Validation Return Codes

Paymentech provides return codes when verifying customer addresses. To change how BRM responds to validation return codes, edit the PCM_OP_PYMT_POL_VALIDATE policy opcode source.

For example, by default an invalid address does not cause a validation failure. You can change the policy to fail validation if the customer’s street address is wrong. For example, change this code:

```c
case PIN_CHARGE_RES_FAIL_ADDR_LOC:
    /* street address failure is acceptable */
    result = PIN_RESULT_PASS;
    descr = "street address not correct";
    break;
```

To this:

```c
case PIN_CHARGE_RES_FAIL_ADDR_LOC:
    /* street address failure is acceptable */
    result = PIN_RESULT_FAIL;
    descr = "street address not correct";
    break;
```

For more information about credit card validation, see "About Credit Card Validation and Authorization".

PCM_OP_PYMT_POL_VALIDATE returns the result of validating a credit card transaction in the PIN_FLD_VENDOR_RESULTS field, including a description of that result. You can customize credit card validations based on the response from ACH by passing a PIN_FLD_VENDOR_RESULTS value in the input flist. For example, you can set the validation to pass or fail, or turn on logging based on the results from the ACH.

PCM_OP_PYMT_VALIDATE calls this policy opcode during customer registration to determine the success or failure of credit card validation.
You can change both the PIN_FLD_RESULT and PIN_FLD_DESCR values to modify BRM responses to validation results returned in PIN_FLD_VENDOR_RESULTS. For example, if your company doesn’t want to proceed with a transaction when the result is PIN_CHARGE_RES_SRV_UNAVAIL, change the PIN_FLD_RESULT_PASS value to PIN_FLD_RESULT_FAIL and change the PIN_FLD_DESCR value to “Service unavailable”.

If the PIN_FLD_RESULT value passed in on the input flist is NULL, PCM_OP_PYMT_POL_VALIDATE does nothing. Otherwise, the default validation result depends on the PIN_FLD_RESULT value. See Table 2–2, "Result Values for Operation".

---

**Important:** If you add custom result values to your system, do not assign them the following result codes, which are reserved by BRM: 0 - 17, 777, 888, 999, 1000 - 1017, 1777, and 1999.

---

**Handling AVS Validations for International Credit Cards**

By default, BRM sends a customer’s name, address, and ZIP code for validation when processing credit card charges. If you use the Address Verification System (AVS) to validate addresses, only credit cards with addresses in the United States and Canada pass validation.

To change how BRM handles credit card address verifications, do one of the following:

- Set the cc_validate flag in the CM’s pin.conf file to 0. This disables the address validation process by Paymentech for all credit cards, including United States and Canada credit cards.

- Set the cc_validate flag in the CM’s pin.conf file to 1 and modify PCM_OP_PYMT_POL_VALIDATE to ignore all AVS failure response codes for other countries. This changes how BRM responds to Paymentech’s validation return codes for countries other than the United States and Canada.

**Customizing How the Results of Credit Card Transactions are Processed**

To process the result of a credit card transaction for a specified account, use the PCM_OP_PYMT_POL_COLLECT policy opcode.

PCM_OP_PYMT_POL_COLLECT sets the PIN_FLD_RESULT and PIN_FLD_DESCR values returned in the output flist. It also specifies the actions to be performed on the account based on the results of the credit card transaction by returning a PIN_FLD_ACTIVITIES array.

PCM_OP_PYMT_COLLECT calls this opcode after the credit card has been charged.

The default behavior of PCM_OP_PYMT_POL_COLLECT is determined by the PIN_FLD_RESULT field passed in on the input flist.

- If the result is successful, PIN_CHARGE_RES_PASS is passed in. Depending on the PIN_FLD_COMMAND passed in on the flist, PCM_OP_PYMT_POL_COLLECT sets the PIN_FLD_DESCR value as shown in Table 3–2:
Customizing How the Results of Credit Card Transactions are Processed

Table 3–2 Input and Output Values

<table>
<thead>
<tr>
<th>Input PIN_FLD_RESULT</th>
<th>Input PIN_FLD_COMMAND</th>
<th>Output PIN_FLD_RESULT</th>
<th>Output PIN_FLD_DESCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN_CHARGE_RES_PASS</td>
<td>PIN_CHARGE_CMD_AUTH_ONLY</td>
<td>PIN_RESULT_PASS</td>
<td>Authorization successful</td>
</tr>
<tr>
<td>PIN_CHARGE_RES_PASS</td>
<td>PIN_CHARGE_CMD_CONDITION</td>
<td>PIN_RESULT_PASS</td>
<td>Authorization &amp; deposit successful</td>
</tr>
<tr>
<td>PIN_CHARGE_RES_PASS</td>
<td>PIN_CHARGE_CMD_DEPOSIT</td>
<td>PIN_RESULT_PASS</td>
<td>Deposit successful</td>
</tr>
<tr>
<td>PIN_CHARGE_RES_PASS</td>
<td>PIN_CHARGE_CMD_REFUND</td>
<td>PIN_RESULT_PASS</td>
<td>Refund successful</td>
</tr>
</tbody>
</table>

PCM_OP_PYMT_POL_COLLECT then specifies actions to be performed on the account based on the PIN_FLD_COMMAND, the payment amount, and the pending receivable amount as shown in Table 3–3:

Table 3–3 PCM_OP_PYMT_POL_COLLECT Actions

<table>
<thead>
<tr>
<th>Input PIN_FLD_COMMAND</th>
<th>Actions When Payment &gt;= Pending Receivable</th>
<th>Action When Payment &lt; Pending Receivable</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN_CHARGE_CMD_AUTH_ONLY</td>
<td>Clear the pending receivable amount.</td>
<td>Credit toward outstanding bill.</td>
</tr>
<tr>
<td>PIN_CHARGE_CMD_CONDITION</td>
<td>If the account status is currently set to inactive, change it to active.</td>
<td></td>
</tr>
<tr>
<td>PIN_CHARGE_CMD_REFUNDS</td>
<td>Set the status flag value to PIN_STATUS_FLAG_DEBT.</td>
<td></td>
</tr>
<tr>
<td>PIN_CHARGE_CMD_DEPOSIT</td>
<td>No action specified</td>
<td>No action specified</td>
</tr>
</tbody>
</table>

- If the result is unsuccessful, and PIN_CHARGE_RES_FAIL_CARD_BAD or PIN_CHARGE_RES_FAIL_DECL_HARD are passed in, PCM_OP_PYMT_POL_COLLECT sets the PIN_FLD_RESULT value and the PIN_FLD_DESCR description, and then specifies these actions as shown in Table 3–4:

Table 3–4 Input and Output Values

<table>
<thead>
<tr>
<th>Input PIN_FLD_RESULT</th>
<th>Output PIN_FLD_RESULT</th>
<th>Output PIN_FLD_DESCR</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN_CHARGE_RES_FAIL_CARD_BAD</td>
<td>PIN_RESULT_FAIL</td>
<td>Credit card operation declined</td>
<td></td>
</tr>
<tr>
<td>PIN_CHARGE_RES_FAIL_DECL_HARD</td>
<td>PIN_RESULT_FAIL</td>
<td>Credit card operation declined</td>
<td>Set account status to inactive.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Set status flag value to PIN_STATUS_FLAG_DEBT.</td>
</tr>
</tbody>
</table>

The PIN_FLD_PAYMENT_REASONS array in the input flist contains a PIN_FLD_PAYMENT_REASONS array, which contains information related to failed payments. These values are recorded in the FAILED_ACCOUNTS array of the /process_audit/billing object.
The remaining input PIN_FLD_RESULT values are implemented in the same way. PCM_OP_PYMT_POL_COLLECT sets the output PIN_FLD_RESULT value and the PIN_FLD_DESCR description, and then reads the item for the account to determine if there is an amount that is 30 days past due. If so, it specifies the following actions listed in Table 3-5:

Table 3-5  Input and Output Values

<table>
<thead>
<tr>
<th>Input PIN_FLD_RESULT</th>
<th>Output PIN_FLD_RESULT</th>
<th>Output PIN_FLD_DESCR</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN_CHARGE_RES_FAILDECL_SOFT</td>
<td>PIN_RESULT_FAIL</td>
<td>Credit card operation declined</td>
<td>■ Set account status to inactive.</td>
</tr>
<tr>
<td>PIN_CHARGE_RES_FAILADDR_AVS</td>
<td></td>
<td></td>
<td>■ Set status flag value to PIN_STATUS_FLAG_DEBT.</td>
</tr>
<tr>
<td>PIN_CHARGE_RES_FAILADDR_LOC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIN_CHARGE_RES_FAILADDR_ZIP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIN_CHARGE_RES_FAILNO_ANS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PIN_CHARGE_RES_SRVC_UNAVAIL</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

You can customize PCM_OP_PYMT_POL_COLLECT to specify any of these actions:

■ clear_pending
■ set_status
■ cease_billing

For example, to discontinue billing an account after a determined period of inactivity, specify the cease_billing action. The default implementation does not specify the cease_billing action for any input PIN_FLD_RESULT values.

Changing How BRM Handles Paymentech Authorization Return Codes

The Paymentech authorization codes used by BRM are listed in BRM_Home/sys/dm_fusa/fusa_codes. This file maps Paymentech authorization codes to BRM result codes.

The fusa_codes file is not a complete list, but it includes the most common codes returned by Paymentech. If a Paymentech code is not included in the list, it is mapped to a hard decline.

You can change the mappings or add new mappings by editing the fusa_codes file.

Note: You can map a Paymentech code to any BRM result code except CHECKPOINT.

1. Open BRM_Home/sys/dm_fusa/fusa_codes.
2. Use the instructions in the file to edit the file.
3. Save the file.
4. Stop and restart the Paymentech DM. See “Starting and Stopping the BRM System” in BRM System Administrator’s Guide.

For more information about credit card authorization, see "About Credit Card Validation and Authorization".

Testing Paymentech Credit Card Processing

Paymentech provides connections for testing credit card and direct debit processing, but you need to schedule testing times with Paymentech. It can take several weeks to establish a link with Paymentech. You can use the BRM Paymentech simulators to test credit card and direct debit processing without connecting to Paymentech.

Use the payment processing simulator to do the following:

- Test the connections in your payment processing configuration.
- Test how to handle no response or dropped-line situations.
- Test any part of your BRM system that includes BRM-initiated payment processing. For example, you can create credit card accounts in Customer Center and use the simulator to charge them.
- Test how the BRM system responds to credit card validation and authorization. You can also test BRM’s response to the Visa fraud prevention system (CVV2). For example, you can test how BRM responds when trying to create an account that uses an invalid credit card.

---

**Note:** The Paymentech simulator does not check for the expiration date of the credit card.

---

The payment processing simulator is located in BRM_Home/bin. It includes two utilities, answer_s and answer_b.

---

**Caution:** Use the answer_s and answer_b utilities only in the test environment.

In the production environment, *uninstall* these utilities to prevent sensitive data from being used for verification.

---

- The answer_s utility simulates online transactions.

---

**Note:** The answer_s utility automatically simulates the Paymentech HeartBeat application during BRM-initiated payment processing. It verifies that the HeartBeat responses from the Paymentech Data Manager (dm_fusa) are on time and in the correct format when sent to the payment processor (which is the answer_s utility in this case). If not, the utility resets itself to a listen state, which the simulator handles as a socket disconnect and writes the errors to the BRM_Home/apps/fusa_server/answer_s.pinlog file, if one is configured. See "Defining the Credit Card Functionality to Test". For information on how to handle the errors, see "Troubleshooting HeartBeat Errors".
The `answer_b` utility simulates batch transactions.

**Setting Up the Paymentech Simulator**

Setting up the Paymentech simulator involves the following tasks:

- **Defining the Credit Card Functionality to Test**
- **Setting Up the Paymentech DM Configuration File for Testing**
- **Specifying an IP Address for the Paymentech Simulator**

**Defining the Credit Card Functionality to Test**

You can define which area of functionality to test with `answer_s` and `answer_b` by editing the Paymentech simulator configuration file (`BRM_Home/apps/fusa_server/pin.conf`). This file includes configuration instructions.

**Note:** The entries can be changed interactively because the `answer_a` and `answer_s` servers read them from the configuration file at each connection.

For information about validation and authorization, see "About Credit Card Validation and Authorization".

1. Open the simulator configuration file (`BRM_Home/apps/fusa_server/pin.conf`).
2. Change the response and result codes as necessary, for example:
   - `answer_s v_code 100`
   - `answer_s avs I3`
   - `answer_s s_code M`
   - `answer_b v_code 100`
   - `answer_b avs I3`
3. To write processing information to a log file, add the following entries:
   - `answer_s loglevel 3`
   - `answer_s answer_s.pinlog`
   - `answer_b loglevel 3`
   - `answer_b answer_b.pinlog`
4. Save and close the file.

**Setting Up the Paymentech DM Configuration File for Testing**

1. Open the Paymentech DM configuration file (`BRM_Home/sys/dm_fusa/pin.conf`).
2. Specify at least two `dm_fusa` back ends.
3. Change the `online_srvr` and `online_port` entries to point to the `answer_s` utility port number and IP address. By default, the port number is 5678.
4. Change the `batch_srvr` and `batch_port` entries to point to the `answer_b` utility port number and IP address. By default, the port number is 5679.
5. Save and close the file.
Specifying an IP Address for the Paymentech Simulator

Systems configured with multiple network cards use multiple IP addresses for each network card. You can configure the Paymentech simulator to listen to all IP addresses to determine where to connect, or, if you know the IP address: for example, one provided by Paymentech: you can define it specifically in the answer pin.conf file.

1. Open the simulator configuration file (BRM_Home/apps/fusa_server/pin.conf).
2. Do one of the following:
   - To enable Paymentech to listen to any IP address located on the machine where the answer utility is running, add the following entry to the file:
     ```
     - answer answer_name
     ```
   - To assign a specific IP address for the answer utility, add the following entry to the file:
     ```
     - answer answer_name IP_address
     ```
     Where IP_address is the IP address of the system running the simulator.
     For example:
     ```
     - answer answer_name 102.13.112.122
     ```

Running the Paymentech Simulators

The Paymentech simulators are in BRM_Home/bin.

---

**Note:** Start the simulators before you start the Paymentech DM.

---

You can start and stop the simulators through the command line:

```bash
start_answer &
stop_answer
```

Simulating Failed Credit Card Transactions

General soft declines are failures that can be retried later with possible success. This includes reasons like insufficient credit limit and other transitory causes. General hard declines are failures that are unlikely to succeed if retried. These include reasons like lost and stolen credit card and chronic payment failures.

To create a hard or soft decline on a credit card that you can use to test resolving failures, do the following:

1. Create a credit card account.
2. Stop the answer_b utility and the Paymentech DM.
3. Open the answer_b configuration file (BRM_Home/apps/fusa_server/pin.conf) and change the vcode value to 502.
4. Restart the answer_b utility. See "Running the Paymentech Simulators".
5. Restart the Paymentech DM. See “Starting and Stopping the BRM System” in BRM System Administrator’s Guide.
6. Advance the time one month and run pin_bill_day.
7. Verify that the amount due is not collected.
8. Use Event Browser to verify the PIN_FLD_RESULTS value in the /event/billing/payment/cc object is a 7 (soft decline) or an 8 (hard decline).

Resolving Failed Credit Card Transactions

In addition to the regular responses, answer_b also handles request for response (RFR) file requests by returning the contents of the RFR file specified in the answer_b configuration file.

To test recovery of failed transactions:
1. Create a credit card account.
2. If you have not already created failed credit card transactions, do the following to force a transaction failure:
   a. Advance the time one month.
   b. Run pin_bill_day.
   c. Stop the answer_b utility while billing runs.
3. Use Event Browser to verify the PIN_FLD_RESULTS value in the /event/billing/payment/cc object is a 6 (service unavailable).
4. Run the pin_clean utility to find transaction IDs for failed transactions.
5. Edit a fusa send file (fusas*). Enter the transaction IDs for the transactions that have checkpoint records. Fusa send files are located in the TEMP directory.
6. Enter the file name of the RFR file in the Paymentech simulator configuration file.
7. Resolve the failed transactions. See "Resolving Failed BRM-Initiated Payment Transactions".

About Paymentech Fraud Prevention Using CID and CVV2

Paymentech offers an additional fraud prevention option for Visa and American Express transactions. Visa and American Express debit and credit cards have a non-embossed identifier. The Visa CVV2 (Card Verification Value 2) number is on the back of the card in the signature panel. The American Express CID (Card identifier) number is on the front of the card. CSRs can request this information when registering customers. For more information, see "About Credit Card Fraud Prevention" and "Requiring Additional Protection Against Credit Card Fraud".

About Paymentech Soft Descriptor Credit Card and Checking Statement Information

You can add soft descriptors to customers’ credit-card or checking-account statements to make charges easier to understand. Soft descriptors are available for Paymentech direct debit and credit card processing. You can add to these statements:

- Your DBA (doing business as)
- Your product name
- A customer service phone number (instead of your headquarters city)

Visa gives a discount, the Visa PS2000 Direct Marketing interchange rate, to companies that provide a customer service number in this manner.

Use this format for DBA name, product name, and phone entries:
Implementing a Direct Debit Payment Method

This section explains how to implement a direct debit payment method with BRM.

Direct Debit Options

BRM provides you with the following options for charging customers using a direct debit payment method:

- A direct debit method using the Paymentech payment processing service. See "Enabling Direct Debit Processing" for details.
- Tools for creating a direct-debit-processing implementation using the bank of your choice. See "Implementing a Custom Direct Debit Payment Method".

---

**Caution:** To avoid corrupting your database, perform the instructions in this section on a test system. You will reinitialize your database with schema changes as the last step of this process.

Direct Debit Installation

Before installation, change the default values for these entries in the BRM_Home/setup/pin_setup.values file before you run the pin_setup script at installation:

```sh
$INIT_DDEBIT = 'YES';
$SETUP_DROP_ALL_TABLES = 'YES';
```

If you did not set these values when you installed BRM, you must do so and reinstall.

Direct Debit Components

To process Paymentech and BRM payment methods, BRM includes:

- Objects
Implementing a Direct Debit Payment Method

You can add other billing and payment methods that you need for your business, and customize the payment interface for those bill types.

Implementing a Custom Direct Debit Payment Method

This section describes how to add a custom direct debit payment method to your BRM system. See "Direct Debit Installation" for important installation information.

Overview of Adding a Custom Direct Debit Implementation

To implement a payment interface for other bill types:

1. Creating /payinfo Storable Classes as needed.
2. Modifying Customer Center to support communications between Customer Center and the modular payment interface.
3. Creating New Opcodes as needed.
4. Creating New Event Storable Classes as needed.
5. Creating a New Data Manager.
6. Updating the /config/payment Storable Object to support new opcodes.
7. “Viewing Payment Information for a Custom Payment Method” in BRM Managing Customers to see the options.

Note: Whenever you add opcodes, opcode flags, or new event types to BRM, you must also then edit the init_objects_ddebit.source file and run the pin_init script.

Creating /payinfo Storable Classes

Your new class is a subclass of the /payinfo storable class with information specific to the new payment method. The information contained in the object instances of your new subclass comes from account creation. For each new payment method:

1. Determine what information you require in order to bill the customer correctly.
2. Create a new storable class to store that information.

For information on creating new subclasses, see “Creating Custom Fields and Storable Classes” in BRM Developer’s Guide.

Modifying Customer Center

To get payment information from and to support data entry for the new payment methods in Customer Center, you must modify Customer Center. See “Using Customer Center SDK” in BRM Developer’s Guide.
Creating New Opcodes
To manipulate objects derived from your new storable class, create new opcodes and Facilities Modules to implement them. For more information, see “Writing a Custom Facilities Module” in BRM Developer’s Guide.

Creating New Event Storable Classes
Your direct debit implementation will probably require you to add information to the existing BRM default /event/billing classes.
This list should cover what you will be adding. Add a subclass if needed:
- /event/billing/charge
- /event/billing/payment
- /event/billing/recover
- /event/billing/refund
- /event/billing/validate
For example, if you need to extend /event/billing/charge for a program called my_dd, the class name would be /event/billing/charge/my_dd.
These storable classes are good models for what you’ll need. They are used for BRM’s default payment methods:
- /event/billing/charge/cc
- /event/billing/charge/dd
For information on creating storable classes, see “Creating Custom Fields and Storable Classes” in BRM Developer’s Guide.

Creating a New Data Manager
You must create a new Data Manager (DM) if collection or validation of your new payment method requires interaction with an external system. For information on creating a DM, see “Writing a Custom Data Manager” in BRM Developer’s Guide.

Updating the /config/payment Storable Object
When you add a new payment method, you must update the /config/payment storable object. See “Customizing Payment Methods” in BRM Managing Customers.
For information on how to view custom payment information in Customer Center, see “Viewing Payment Information for a Custom Payment Method” in BRM Managing Customers.
This chapter provides an overview of Oracle Communications Billing and Revenue Management (BRM) payment channel information and describes how to define and load the payment channel IDs and descriptions into the BRM database.

Before reading this chapter, you should understand BRM payment processing. See the following documents:

- About Payments
- How BRM Collects Payments

### About Payment Channel Information

*Payment channel information* is a payment property that identifies the delivery method by which customer payments are sent to a financial institution. For example, payment channels include the Internet, Interactive Voice Response (IVR) phone service, Automated Clearing House (ACH), and lockbox.

---

**Note:** You can copy and paste the key or you can type it.

---

You can use the payment channel information to implement customizations in BRM, such as suspending payments, charging failed payment fees, and offering early-payment incentives. For more information, see "How BRM Collects Payments".

### Setting Up Payment Channel Information

To set up payment channel information for your system, you must first define and load the information into BRM, and then configure it for BRM-initiated payment processing.

For BRM-initiated payment, the payment gateway needs to include the payment channel information in each payment. When the payments are received by BRM, they will be processed automatically with the correct channel ID.

For externally initiated payments, the payment gateway needs to map the external payment channel information to BRM channel IDs in each payment file. Therefore, the payment channel information should already be included in the imported payment batch. If a payment does not contain a payment channel ID, the payment batch-level channel ID is used for that payment. If neither the payment nor the batch contains a payment channel ID, the information can be entered manually by using Payment Tool.
Defining Payment Channel Information in BRM

The payment channel information you load into the BRM database consists of payment channel IDs and the text strings that describe them. To define payment channel IDs, you edit the `payment_channel.en_US` sample file in the `BRM_Home/sysmsgs/paymentchannels` directory. You then use the `load_localized_strings` utility to load the contents of the file into the `strings` objects.

When you run the `load_localized_strings` utility, use this command:

```
load_localized_strings payment_channel.locale
```

**Note:** If you’re loading a localized version of this file, use the correct file extension for your locale. For a list of file extensions, see Locale names.

For information on loading the `payment_channel.locale` file, see “Loading Localized or Customized Strings” in BRM Developer’s Guide. For information on creating new strings for this file, see “Creating New Strings and Customizing Existing Strings” in BRM Developer’s Guide.

Mapping Payment Channel IDs for BRM-Initiated Payments

For BRM-initiated payments such as credit card and direct debit payments, the payment channel for a particular vendor is retrieved from the payment processor configuration object and automatically saved in BRM with each payment.

To map the payment channels, you run the `load_pin_ach` utility to load the contents of the `pin_ach` file into the `/config/ach` object in the BRM database.

**Note:** The `load_pin_ach` utility requires a `pin.conf` configuration file. For more information, see “Creating Configuration Files for BRM Utilities” in BRM System Administrator’s Guide.

**Caution:** The `load_pin_ach` utility overwrites existing payment processor and merchant information. If you are updating information, you cannot load new data only. You must load complete sets of payment processor and merchant entries each time you run the `load_pin_ach` utility.

1. Edit the `pin_ach` file in `BRM_Home/sys/data/pricing/example` by specifying each vendor and its payment channel ID. The `channel_id` value must match a payment channel ID configured in the `/strings` object.

   The file contains instructions and an example.

2. Save the `pin_ach` file.

3. Use the following command to run the `load_pin_ach` utility:

   ```
   load_pin_ach ach_file
   ```
If you are not in the same directory as the `load_pin_ach` file, include the complete path to the file, for example:

```
load_pin_ach BRM_Home/sys/data/pricing/example/ach_file
```

4. Stop and restart the Connection Manager (CM). See “Starting and Stopping the BRM System” in BRM System Administrator’s Guide. If necessary, restart Payment Tool.

5. Verify that the `pin_ach` file was loaded successfully by using the Object Browser to display the `/config/ach` object, or use the `testnap` utility with the `robj` command. See “Reading an Object and Writing its Contents to a File” in BRM Developer’s Guide.

If a payment does not contain a payment channel ID, a value of 0 will be saved with the payment by default, which configures it as **Unspecified Payment Channel**.

For more information on setting up merchants and automated clearing houses, see "About BRM-Initiated Payment Processing".

**Configuring Payment Channel IDs for Externally Initiated Payments**

For externally initiated payments, you must configure the payment gateway or custom CRM tool with the payment channel ID information. When the payment batch is received, you use the PCM_OP_PYMT_COLLECT opcode or Payment Tool to load the channel ID into BRM.

You can run the `testnap` utility to check that the payment channel IDs were loaded properly. For more information on testnap, see “Testing Your Applications and Custom Modules” in BRM Developer’s Guide.

**Assigning Payment Channel IDs to Externally Initiated Payments**

You assign a payment channel ID to a payment batch or an individual payment by using Payment Tool. Each batch accepts payments, refunds, or reversals in only one payment channel ID.

For information on assigning or changing the payment channel ID, see Payment Tool Help.
This chapter explains how to configure Oracle Communications Billing and Revenue Management (BRM) payment collection dates for automatic customer payments. Before reading this chapter, read the following topics:

- “About Billing” in *BRM Configuring and Running Billing.*
- About Payments
- About BRM-Initiated Payment Processing
- “About Accounts Receivable” in *BRM Managing Accounts Receivable*

**About Configuring Payment Collection Dates for Automatic Payments**

By default, BRM-initiated payments, such as payments made by credit card or direct debit, are collected on the date that bills are finalized. Alternatively, you can configure BRM to collect a BRM-initiated payment on the date a bill is due or on a specified number of days before the bill is due.

To support configurable payment collection dates, BRM-initiated payment processing involves these steps:

1. You configure the payment collection date.

   During account creation or modification, a customer service representative (CSR) uses third-party customer relationship management (CRM) software to set the collection date for BRM-initiated payments. This date is one of the following:
   - Date the bill is finalized (default)
   - Date the bill is due
   - A specified number of days before the bill due date

   For information about the opcode to call to set this date, see "How BRM Calculates Payment Collection Dates".

2. BRM calculates the payment collection date.

   At the end of each billing cycle, the PCM_OP_BILL_POL_CALC_PYMT_DUE_T policy opcode calculates a bill’s payment collection date after the opcode determines the bill’s due date.

3. BRM collects the payment.

   BRM-initiated payments are collected by the `pin_collect` utility. This utility collects payments for bills whose payment collection date is one of the following days:
About Configuring Payment Collection Dates for Automatic Payments

- The day the utility is run
- The day before the utility is run

**Note:**
- The payment collection date of a bill (/bill object) is stored in the /billinfo object with which the bill is associated.
- To collect BRM-initiated payments for bills whose payment collection date is on a day other than the days listed above, use the pin_collect utility’s start and end parameters. See “Specifying Start and End Times” in BRM Configuring and Running Billing.

For more information, see “About Collecting BRM-Initiated Payments” and “pin_collect”.

About Configurable Payment Collection Dates and On-Demand Billing

Usually, you bill a customer only at the end of the customer’s billing cycle. However, you can use the Bill Now feature in Customer Center or the BRM on-demand billing feature to bill the customer immediately. When you use these features, multiple bills associated with a single bill unit (/billinfo object) may be generated during the same billing cycle. When this occurs, all subsequent bills generated before BRM collects the first bill are collected on the first bill’s payment collection date.

For example, Account A has one bill unit. Its monthly bill, which is paid by direct debit, is due 31 days after it is finalized. Its payment is collected 5 days before the due date. On August 10 (the end of the July 10–August 10 billing cycle), regular billing is run:

- Bill finalized = “Bill 1” (see Figure 5–1)
- Due date = September 10 (August 10 + 31 days)
- Payment collection date = September 5 (September 10 - 5 days)

**Figure 5–1  Regular Billing Cycle Dates**

The Bill 1 payment collection date (September 5) is stored in the /billinfo object associated with Bill 1.

On August 18, the Bill Now feature is used to bill the account:

- Bill finalized = “Bill Now” (see Figure 5–2)
- Due date = September 18 (August 18 + 31 days)
Payment collection date = September 13 (September 18 - 5 days)

**Figure 5–2  Bill Now Billing Cycle Dates**

However, the Bill Now payment collection date (September 13) is not stored in the /billinfo object. Instead, the earlier payment collection date (September 5) is applied to both bills as shown in Figure 5–3:

**Figure 5–3  Bill Now Payment Collection Date**

---

Note: If the Bill Now payment collection date were stored in the /billinfo object on August 18, it would overwrite the Bill 1 payment collection date, changing the date from September 5 to September 18. This would postpone Bill 1’s payment collection for over a week.

For more information about Bill Now and on-demand billing, see “About Bill Now” and “About On-demand Billing” in BRM Configuring and Running Billing.

### About Configurable Payment Collection Dates and Delayed Billing

The BRM delayed billing feature enables billing for all the bill units (/billinfo objects) in your system to be run a specified number of days after the end of their billing cycle. If you use delayed billing, be careful to avoid configuring payment collection dates that occur before bills are finalized.

For example, your system has a 14-day billing delay. Account A’s bill is due 21 days after the end date of its monthly billing cycle. If you set a payment collection date that is more than 7 days before the bill due date, the payment collection date will occur...
before the bill is finalized. In such cases, BRM ignores the payment collection date and collects the payment on the date the bill is finalized.

For information about delayed billing, see “Setting up Delayed Billing” in *BRM Configuring and Running Billing*. 
6

Configuring Payment Fees

This chapter provides an overview of how payment fees are handled in your Oracle Communications Billing and Revenue Management (BRM) system. It includes:

- An overview of failed payments and payment fee processing.
- Information on how to configure BRM for payment fees, create payment fee products, and manually remove a payment fee from an account balance.

For information on customizing payment fees, see "Customizing Payment Fees". For background information on payments, see "About Payments".

About Failed Payments

If you use BRM-initiated payment processing, payments are loaded directly into BRM by the payment processor. If you use externally initiated payment processing, you use Payment Tool to load payments into BRM. Regardless of the way they are loaded, by default they are posted as successful or failed, and are identified by PIN_FLD_STATUS value.

Successful payments are automatically posted to the account to which they belong. The payment amount is removed from the current balance on the account, and any remaining amount is allocated according to your business policies.

Failed payments are those that don't comply with the financial practices of your company because, upon collection, they have been dishonored or rejected by the bank. For example, payments can fail due to expired credit and debit cards, incorrect account details, and insufficient funds. You can configure BRM to charge payment fees for payments that have a failed status.

Note: If you have the Payment Suspense Manager feature enabled, failed payments are submitted to the payment suspense account, but can still receive payment fees. See "Configuring Payment Suspense Manager" for more information on Payment Suspense Manager.

For information on payment status, see "About Payment Status".

For information on failed unconfirmed payments, see "Handling Failed Unconfirmed Payments".
About Payment Fees

Payment fees are one-time, non-recurring penalties that can be charged to an account for payments that fail due to financial errors. For example, if a customer check is denied due to insufficient funds, or a credit card is invalid because it has expired, you can charge the customer a payment fee.

By default, payment fees can be charged only for failures that occur due to financial reasons, and not for failures that occur due to communication errors between BRM and the payment transaction service. Communication errors are considered unresolved transactions. You run the `pin_clean` utility to find all unresolved credit card and direct debit payments recorded in the BRM database. For more information, see "Resolving Failed BRM-Initiated Payment Transactions".

When a failed payment is posted in BRM, it is recorded in BRM with a balance impact of 0, and is identified by a transaction ID, a failure status, and a reason ID for the failure. BRM uses the failure status and reason ID to determine whether to apply payment fees when the payment is posted. Payment fees can be applied only to payments that have a PIN_FLD_STATUS value of `PIN_PYMT_FAILED`.

If you defined payment fees, the failed payment triggers the creation of a payment fee event, and the rated amount is applied to the customer’s current account balance. Failed payments can be posted to an account only when the account number is available in BRM. If a failed payment isn’t posted, the associated fee cannot be applied.

You create fees for failed payments by setting up pricing plans. The product in the plan contains a rate that is mapped to the payment fee event. When you create a payment fee product, any account that own the product will receive a payment fee if a payment fails; you can’t exempt accounts from receiving payment fees. To cancel out unwanted fees from an account’s balance, you set up account-level discounts for the payment fee event. See "Defining Exemptions from Payment Fees".

BRM applies the balance impact of the payment fee event to the default balance group of the bill unit.

Configuring BRM for Payment Fees

Before you can define payment fees, you need to configure BRM with the payment information. After your system is configured, you create the payment fees by using pricing plans.

Setting up payment fees involves the following processes:

- Defining Payment Attributes for Payment Fees
- Defining Reason Codes for Failed Payments
Defining Payment Attributes for Payment Fees

BRM calculates payment fees by using real-time rating. Payment fees are based on attributes you define in the Pricing Center rate plan selector. By default, the rate plan selector and rating opcodes work with three attributes when determining the payment fee: customer segment, payment channel, and payment method. The failed payment storable class contains these default fields upon which you can base your payment fees.

To charge payment fees based on any of these payment attributes, you must first define the attributes in the BRM database. For information on defining payment attributes, see the following information:

- "Creating and Managing Customer Segments" in BRM Managing Customers.
- Configuring Payment Channels
- "Customizing Payment Methods" in BRM Managing Customers.

To expand the attributes available in the rate plan selector, you can customize the PCM_OP_PYMT_POL_APPLY_FEE policy opcode such that the flist includes extra field types or provides additional filtering logic. You also extend the /event/billing/payment/failed storable class to include the added fields. After you perform this customization, you can create payment fee rate plans that enable BRM to consider the additional attributes.

For more information, see "Customizing Payment Fees".

Defining Reason Codes for Failed Payments

Reason codes explain why payments fail, and they enable you to charge a payment fee based on the reason for the failure.

The payment gateway and your third-party payment application must be configured to identify failed payments and send reason codes with the payment information. When a failed payment is received, the reason code is mapped to the reason code ID defined in the BRM database.

You define reason codes in the reasons.locale file and load them into the BRM database as a /strings object. The file contains instructions on how to add the new domain Reason Codes - Payment Failure Reasons. For example:

```plaintext
DOMAIN = "Reason codes-Payment Failure Reasons";
STR
  ID = 1001 ;
  VERSION = 13 ;
  STRING = 'Invalid Credit Card';
END
```

**Note:** If you add your own reason codes to the reasons.locale file, you should use IDs above 100,000.

To define reason codes for failed payments, you edit the reasons.en_US sample file in the BRM_Home/sys/msgs/reasoncodes directory. You then use the load_localized_strings utility to load the contents of the file into the /strings objects.

When you run the load_localized_strings utility, use this command:

```
load_localized_strings reasons.locale
```
Creating Payment Fees

You create payment fees by defining rates and configuring real-time rating. The rates you define for payment fees are based on the fields defined in the /event/billing/fee/failed_payment event. These fields enable you to charge and suppress payment fees.

**Note:** You should be familiar with real-time rating before you begin. For information on creating rates and pricing plans, see Pricing Center Help and "About Real-time Rate Plans" in BRM Setting Up Pricing and Rating.

Defining a Payment Fee

This example defines payment fees based on the payment method, which is the means by which customers pay their bills. It charges a $5 fee for failed wire transfers, credit card, debit card, and direct debit payments; a $7 fee for failed check, postal order, and invoice payments; and no fee for failed cash payments. Failed cash payments are handled by BRM as non-payments or overdue payments.

See "About Payment Methods" for more information about payment methods.

1. Start Pricing Center and begin creating a **System** product.

   **Note:** This defines payment fees for all customer accounts. To define fees only for certain accounts, create a **Subscription** product and purchase the product for the account.

2. Apply the product at the account level and define the purchase and ownership information.

3. In the **General Product Info** tab, type 1 in **Priority**.

4. Under Event Map, click **Add**.

   a. In the **Event** column, select **Failed Payment Fee Event**.

   b. In the **Measured By** column, select **Occurrence**.

   c. In the **Rate Plan Structure** column, select **Rate Plan Selector**.

5. Set up the rate plan for the $5 fee.
a. Under Rate Plan Selector, type a name for the payment fee.
b. Click Edit Plans and click New.
c. Define the Plan Details and Rate Plan Structure.
d. In the Balance Impacts tab, select US Dollars [840] as the Resource ID and type 5.00 in Fixed Amount.
e. Click OK.

6. Set up the rate plan for the $7 fee. Repeat step 5, but in the Balance Impacts tab, type 7.00 in Fixed Amount.

7. Set up the rate plan selector.
   a. Click ...+ in the first column, select Event, choose PIN_FLD_FAILED_ PAYMENT_FEE.PIN_FLD_PAY_TYPE from the attributes list, and click OK.
   b. Click + in the row column to create a new row for each payment method, except Cash payments. Omitting the cash payment method from the rate plan selector excludes it from being rated and no fees will be applied.
   c. In the first column of each row, type the element ID for each payment method.
   d. For the credit card, debit card, direct debit, and wire transfer payment methods, select the $5 payment fee rate plan.
   e. For the invoice, check, and postal order payment methods, select the $7 payment fee rate plan.
   
   The rate plan selector for this example looks like Figure 6–1 in Pricing Center:

   ![Rate Plan Selector](image)

8. Click OK and Apply.
Defining Thresholds for Payment Fees

You can define a threshold amount at which payment fees are applied for failed payments. If the amount of the failed payment is less than the threshold, the payment fee isn’t applied. For example, you can set payment fees to be applied only for failed payments of $20 or more.

To create a payment fee threshold, define a rate plan selector in which you base the fee on the PIN_FLD_AMOUNT_ORIGINAL_PAYMENT attribute for the failed payment fee event. Then define balance impacts to define the threshold quantity.

This example charges a $5 fee if the failed payment is over $99.

1. Start Pricing Center.
   a. To set up thresholds for all customer accounts, create a **System** product.
   b. To set up thresholds only for certain accounts, create a **Subscription** product and purchase the product for those accounts.

2. Apply the product at the account level and define the purchase and ownership information.

3. In the **General Product Info** tab, type 1 in **Priority**.

4. Under Event Map, click **Add**.
   a. In **Event**, select **Failed Payment Fee Event**.
   b. In **Measured By**, select **Amount**.
   c. In **Rate Plan Structure**, select **Single Rate Plan**.

5. Set up the rate plan that suppresses the payment fee.
   a. Under **Rate Plan**, click **Open Rate Plan** and set up the **Plan Details**.
   b. Under **Rate Structure**, select **Tier 1** and set up the **Tier Details**.
   c. Under **Rate Structure**, select **Rate 1** and set up the **Rate Data**.
   d. Under **Quantity Discount Brackets**, in **Based on**, select **Rate Dependent**.
   e. Under **Rate Structure**, select **No Minimum - No Maximum** to open the **Balance Impacts** tab.
   f. Under **Valid Quantities**, clear **Maximum** and type 100.
g. In Resource ID, select US Dollars [840].

h. Under Rate Structure, select Rate 1 and click Add to add another balance impact.

i. Select No Minimum - No Maximum to open the Balance Impacts tab.

j. Under Valid Quantities, clear Minimum and type 100.

k. Under Balance Impacts, type 5.00 in Fixed.

l. Click OK.

**Defining Exemptions from Payment Fees**

To exempt accounts from receiving payment fees, you create a real-time pipeline discount and associated it with the failed payment fee event.

For more information on discounting, see Pricing Center Help and "About Discounts" in BRM Configuring Pipeline Rating and Discounting.

1. Start Pricing Center and create a subscription discount that applies to the account.

2. Create a Discount Model that defines a discount of 100%.
   a. Using the Pipeline Toolbox, choose Discount/ChargeShare Trigger from the Discount list, select the DM10%OFF discount trigger, and change it to DM100%OFF.
   b. Create a discount rule and set the Rule Type to Threshold and the Drum Type to Charge.
   c. Enter the threshold amount and charge information.
   d. Change the DM10%OFF discount master to DM100%OFF and the DM10%OFF discount model to DM100%OFF.

3. Set up the event map.
   a. Under Map an Event to a Discount Model, click Add.
   b. In the Event column, select Failed Payment Fee Event.
   c. In the list of discount models, choose DM100%OFF and click Select.

4. Create a deal and add the discount to the deal.

5. Purchase the deal for the accounts that are exempt from the failed payment fee.

When the fee is rated, the following operations occur in BRM.

1. The balance impact of the /event/billing/fee/failed_payment is determined by BRM, and the balance impact is saved as a charge packet.

2. The real-time discounting pipeline calculates the charge packet discount and translates it into a second balance impact for the fee. This balance impact amount is the negative equivalent of the original fee amount.

3. The fee is saved with a balance impact amount of $0.

For example, an account that is exempt from the payment fee receives a $5 balance impact for the fee when it is rated. When the 100% discount is applied, a second balance impact of -$5 is saved with the fee. The two amounts cancel each other, and the fee is $0.

For more information on thresholds, see Pricing Center Help.
Removing a Payment Fee From an Account Balance

To remove a payment fee from an account balance, you use Customer Center to credit the balance of the account for the amount of the fee.

1. Open the account in Customer Center.
2. Go to the Post Paid tab and click the Balance Adjustment link.
3. Select the balance to adjust.
4. If this is a dual currency account, select the currency to adjust.
5. Type the amount of the adjustment and select Credit.
6. Select a reason and type any notes about the adjustment.
7. Select a transaction date and an accounting date for the adjustment.
8. Click OK.
9. Click Yes in response to the confirmation message if the information is correct.

When the adjustment is recorded in the BRM database, the amount of the adjustment is added to the amount in the Adjustments/Payments not applied field. The adjusted amount reduces the amount in the Due now field.

10. If the account uses open item accounting, you need to allocate the adjusted amount to one or more bills.

Until you perform the allocation, the credit reduces the amount due on the account, but it does not reduce the balance of any bills. When allocated, the credit reduces the due amount of the bill or bills.

For more information, see Customer Center Help.

Customizing Payment Fees

You can configure BRM to handle failed payments and to charge payment fees according to custom business policies.

- To apply payment fees, use PCM_OP_PYMT_APPLY_FEE. See "How Payment Fees Are Applied".
- To customize payment fees, use the PCM_OP_PYMT_POL_APPLY_FEE policy opcode. See "Customizing Payment Fees".

This section describes the following customization tasks:

- Customizing Payment Fees
- Storing Additional Information with Payment Fees

For background information on payment fees, see the following documents:

- Configuring Payment Fees.
- Creating Payment Fees.
- Defining Reason Codes for Failed Payments.

How Payment Fees Are Applied

Payment fees are applied by PCM_OP_PYMT_APPLY_FEE. This opcode creates payment fees for payments that fail, for example, due to insufficient account funds or
an expired credit card. It calls PCM_OP_ACT_USAGE to create the payment fee event to be rated.

PCM_OP_PYMT_APPLY_FEE is called by PCM_OP_PYMT_COLLECT.

The behavior of PCM_OP_PYMT_APPLY_FEE is determined by the PIN_FLD_STATUS field passed in on the input flist. Payments are eligible to receive payment fees if they have a PIN_FLD_STATUS value >= PIN_PYMT_FAILED and < PIN_PYMT_STATUS_MAX. The numeric range for financially failed payments is 30-44. For more information, see "About Payment Status".

The normal flow of PCM_OP_PYMT_APPLY_FEE is as follows:

1. Checks the input flist for the status of the payment and the reason ID that describes why the payment failed.

2. Calls the PCM_OP_PYMT_POL_APPLY_FEE policy opcode to perform custom checks before the failed payment fee is applied. See "Customizing Payment Fees". When it returns the output flist, PCM_OP_PYMT_APPLY_FEE validates the information, and creates the failed payment fee events for all failed payments based on the information. The default value in the PIN_FLD_BOOLEAN field on the output flist is 0, which specifies that the fee will be created.

3. If a payment fails, records the /event/billing/payment/payment_type event.

   Note: If the payment is an unconfirmed payment, records the /event/billing/payment/failed event.

4. Creates the /event/billing/fee/failed_payment event.

PCM_OP_PYMT_APPLY_FEE provides feedback on its success or failure through the PIN_FLD_RESULTS array in the output flist. The value in the PIN_FLD_RESULTS field specifies whether the payment fee event was created. A value of 0 signifies that the payment fee event was created and the payment fee applied. A non-zero value signifies that the payment fee event was not created.

In the case of a write-off reversal, the output flist sends a results array of reversal events and tax events (if created) that were passed in by PCM_OP_AR_REVERSE_WRITEOFF.

   Note: Failed payments can only be applied to account numbers that already exist in the BRM database.

The PIN_FLD_EVENTS array of the output flist stores the POID of the failed payment fee event or write off event, if one is created. The PIN_FLD_EVENTS array is contained in the PIN_FLD_RESULTS array.

Flags are not used directly by PCM_OP_PYMT_APPLY_FEE. They are passed in from PCM_OP_PYMT_COLLECT for PCM_OP_PYMT_SELECT_ITEMS. For example, Payment Tool can set the PCM_BILLFLG_DEFER_ALLOCATION flag to indicate which payments should be left unallocated.

Customizing Payment Fees

You can define additional rules for payment fee processing by configuring the PCM_OP_PYMT_POL_APPLY_FEE policy opcode. This policy opcode is called by PCM_OP_PYMT_APPLY_FEE.
PCM_OP_PYMT_POL_APPLY_FEE enables you to customize payment fees by preprocessing, filtering, and extending the information available in failed payment fee events. For example:

- You can charge different fee amounts based on thresholds, or on the reason ID associated with a failed payment.
- You can use payment attributes such as the customer segment, payment method, payment channel, or a combination of these attributes to charge payment fees. You create the filters by passing the values in the PIN_FLD_EXTENDED_INFO substruct or the PIN_FLD_CHARGES array.

The default behavior of PCM_OP_PYMT_POL_APPLY_FEE is determined by the PIN_FLD_STATUS and PIN_FLD_REASON_ID fields passed in on the input flist. It enhances the input flist by adding fields to filter and extend the information available in the payment fee events.

**Customization Example: Charging a Fee Based on the Customer Segment**

The following opcode customization is used to control which failed payment fee is assigned to an account based on the account’s customer segment:

```
If (customer_segment = "early bill payer")
Then set PIN_FLD_BOOLEAN to False
END
```

The PIN_FLD_BOOLEAN value of False specifies that the fee event is not created. When payments are posted, BRM uses the customer segment ID to determine if a payment fee is charged.

You can also use the PIN_RESULT_PASS and PIN_RESULT_FAIL return values in PCM_OP_PYMT_POL_APPLY_FEE to configure whether payment fees are applied.

**Storing Additional Information with Payment Fees**

You can store additional information for payment fees by extending the `/event/billing/fee/failed_payment` storable class. Use the PIN_FLD_FAILED_PAYMENT_FEE field inside the PIN_FLD_EXTENDED_INFO substruct. This enables you to record any additional criteria you defined to create the payment fee.

You can then use the PIN_FLD_REASON_ID field in the input flist to configure fees based on this value of the PIN_FLD_FAILED_PAYMENT_FEE field. It contains the reason for failure that was sent by the payment processor for failed credit card and direct debit transactions.

You can then charge payment fees based on custom payment attributes such as currency type:

1. Extend the `/event/billing/fee/failed_payment` storable class with the new attributes by using Developer Center.
2. Customize the PCM_OP_PYMT_POL_APPLY_FEES policy opcode to pass the value in the PIN_FLD_CHARGES array or the PIN_FLD_FAILED_PAYMENT_FEE field in the PIN_FLD_EXTENDED_INFO substruct.
3. Use Pricing Center to create the rates for the new attribute values.

See "Configuring Payment Fees".
Configuring Payment Incentives

This chapter provides an overview of how payment incentives are handled in your Oracle Communications Billing and Revenue Management (BRM) system. It includes:

- A summary of the payment incentive functionality.
- An overview of how BRM processes payment incentives.
- Information on how to enable BRM for payment incentives, create payment incentive products, and manually reverse a payment incentive.

For information on customizing payment incentives, see "Customizing Payment Incentives".

For background information on payments, see "About Payments".

About Payment Incentives

A payment incentive is a special compensation for customers who pay their bills early and in full. Payment incentives can take the form of free gifts, certificates, free minutes, and so forth. In typical implementations, they reward customers by reducing the bill amount or adding resource bonuses to their accounts. Payment incentives can include currency resources such as monetary credit or non-currency resources such as free minutes. For example, you can award 20 free minutes or provide a 5% reduction in the monthly bill amount. In addition, you can define attributes that govern whether a payment incentive is granted. These attributes specify the types of customers that qualify, the payment methods that qualify, and so forth.

BRM determines an account’s basic eligibility for a payment incentive at payment time but applies the payment incentive during billing, as shown in Figure 7–1:

**Figure 7–1  Payment Incentive Time Line**

Here, the payment incentive is applied to the current bill, but you can apply it to the next bill instead.
About Setting Up Payment Incentives

You create payment incentives in Pricing Center by defining a product and rate plan. You can create payment incentives for the following product types:

- **Subscription**: Create a subscription product for payment incentives that you want to apply on a recurring basis: for example, a reduction in the monthly bill. Subscription products must be purchased by the customer.

- **System**: Create a system product for payment incentives that you want to apply to an entire class of accounts. For example, if you want to reward all of your customers who pay by credit card, you create the payment incentive as a system discount.

A single payment incentive can impact multiple resources: for example, both free minutes and the amount due for a cycle forward fee. Customers may be eligible for multiple payment incentives depending on which products they purchase and whether any payment incentive system products are valid for their account. For more information on creating products and rate plans, see Pricing Center Help.

BRM calculates payment incentives through real-time rating based on attributes you define in the Pricing Center rate plan selector. By default, BRM works with three attributes when determining the payment incentive: customer segment, payment channel, and payment method. To expand the attributes available through the rate plan selector, you customize the PCM_OP_PYMT_POL_GRANT_INCENTIVE policy opcode such that the flist includes extra field types or provides additional filtering logic. You also extend the /event/billing/incentive storable class to include the added fields. Once you perform this customization, you can create payment incentive rate plans that enable BRM to consider the additional attributes.

In addition to creating payment incentives, you need to enable payment incentives in BRM. To perform this task, you modify the /config/business_params storable object.

For more information on performing each of these tasks, see:

- Enabling BRM for Payment Incentives
- Creating Payment Incentive Products
- Customizing Payment Incentives

About Payment Incentive Processing

BRM determines whether to apply a payment incentive when it allocates the payment for the previous bill. BRM makes this decision by verifying:

- Account eligibility for payment incentives.
- Full payment before the bill due date.

If the account qualifies for a payment incentive by meeting both of these conditions, BRM adds a trigger to the /billinfo object. This is known as provisioning the payment incentive.

During the next billing run, BRM checks the /billinfo object for this trigger. If the trigger is present, indicating that the payment incentive is provisioned, BRM passes the payment incentive information to the rating opcodes to calculate the payment incentive. Depending on how the payment incentive is set up, BRM performs one of these actions:

- If the payment incentive is a fee reduction, BRM subtracts the incentive amount from the total currency amount.
If the payment incentive is a resource grant such as free minutes, BRM reduces or increases the total resource amount by the incentive amount, depending on the conventions you use for non-currency resources.

In either case, the payment incentive is applied to the default balance group of the bill unit associated with the bill.

During real-time rating, BRM bases its calculations on either the current bill total or the last bill total, depending on how your pricing expert defined the product.

In default implementations, payment incentives are granted after BRM processes all billing time events including the application of taxes. Therefore, payment incentives cannot be based on a pre-tax bill amount: only on the total after-tax amount. However, you can customize the PCM_OP_PYMT_POL_GRANT_INCENTIVE policy opcode to consider all the /bill items on a before tax basis.

Figure 7–2 shows how BRM processes payment incentives:

Payment incentives are granted only in the billing run for the account’s normal billing cycle. BRM does not apply payment incentives for:

- On-demand billing runs.
- Bill-now billing runs.

If these types of billing runs occur during a billing cycle, BRM ignores any payment incentives. Later, BRM applies the payment incentive during the next normal billing run, provided there was an early payment within the normal billing cycle and the account is eligible.
Enabling BRM for Payment Incentives

How Payment Reversals Affect Payment Incentives

The provisioning of payment incentives can be reversed under certain circumstances, particularly ones that involve unconfirmed payments: those where a payment was allocated before the credit card processor or automated clearing house (ACH) verified funding. For example, a customer pays a bill early by personal bank check, and BRM allocates an unconfirmed payment, consequently applying the incentive. Then, the ACH notifies BRM that the bank account had insufficient funds, and the check failed. In this case, BRM needs to reverse both the payment and the payment incentive provision.

The payment reversal itself triggers the reversal of the payment incentive provision. If a payment is reversed, BRM reverses only those payment incentives that meet these conditions:

- The payment incentive has been provisioned.
- The payment incentive has not yet been applied to the account during a billing run.

If the payment incentive was already applied, you need to perform the adjustment manually. For information on streamlining manual reversals, see "Manually Reversing a Payment Incentive".

Enabling BRM for Payment Incentives

By default, BRM does not process payment incentives. You can enable this feature by modifying a field in the ar instance /config/business_params object created during BRM installation. As part of payment allocation, payment reversal, and billing, BRM reads this object, checking the payment incentive array to determine whether the PIN_FLD_PARAM_VALUE field for this array is set to enable payment incentives.

You modify the /config/business_params object by using the pin_bus_params utility. For information on this utility, see pin_bus_params.

To enable payment incentives:

1. Use the following command to create an editable XML file for the BusParamsAR parameter class:

   `pin_bus_params -r BusParamsAR bus_params_AR.xml`

   This command creates the XML file named bus_params_AR.xml.out in your working directory. If you don’t want this file in your working directory, specify the full path as part of the file name.

2. Search the XML file for following line:

   `<PaymentIncentive>disabled</PaymentIncentive>`

3. Change disabled to enabled.
Creating Payment Incentive Products

4. Use the following command to load the change into the /config/business_params object:

   ```
pin_bus_params bus_params_AR.xml
   ```

   You should execute this command from the BRM_Home/sys/data/config directory, which includes support files used by the utility. To execute it from a different directory, see pin_bus_params.

5. Read the object with the testnap utility or the Object Browser to verify that all fields are correct.


6. Stop and restart the Connection Manager (CM). For more information, see "Starting and Stopping the BRM System" in BRM System Administrator’s Guide.

7. For multiple databases, run the pin_multidb script with the -R CONFIG parameter. For more information on this script, see pin_multidb.

---

Creating Payment Incentive Products

You create payment incentives by defining products and rate plans. The rate plans you set up for payment incentives are based on the fields defined in the /event/billing/incentive event. You use these fields to create attribute combinations in Pricing Center that BRM compares with the actual event to determine whether it should rate the payment incentive and, if so, which rate to use.

---

Note: You should be familiar with real-time rating before you begin. For detailed information on creating rates and pricing plans, see Pricing Center Help and Setting Up Pricing and Rating.

---

Defining a Payment Incentive

This example uses a rate plan selector to define payment incentives based on combinations of two attributes: the payment method and customer segment:

- The payment incentive awards a 15% reduction on the total bill amount for all customers in the “Platinum Subscriber” customer segment who pay by credit card.
- It also awards a $10 reduction on the total bill amount plus 30 free minutes for customers in the “Silver Subscriber” customer segment who pay by cash.

This example includes a restriction for customers in the “Silver Subscriber” customer segment. These customers do not qualify for a payment incentive unless their total bill is over $100.
1. Start Pricing Center and begin creating a **System** product.

   **Note:** A rate plan selector can only contain fields defined in the payment incentive event. For example, you can apply a payment incentive based on a customer segment, payment channel, or payment method, or a combination of these attributes.

2. Apply the product at the account level and define the purchase and ownership information.

3. In the **General Product Info** tab, type 1 in **Priority**.

4. Under Event Map, click **Add**.
   a. In the **Event** column, select **Payment Incentive Event**.
   b. In the **Measured By** column, select **Current Bill Total**.
   c. In the **Rate Plan Structure** column, select **Rate Plan Selector**.

5. Set up a rate plan for the 15% total bill reduction.
   a. Under **Rate Plan Selector**, type **15% Reduction** as the name for the payment incentive.
   b. Click **Edit Plans** and click **New**.
   c. Define the **Plan Details** and **Rate Plan Structure**.
   d. Define the **Quantity Discount Bracket** for the new rate as **Based on: Rate Dependent**.
   e. In the **Balance Impacts** tab, select **US Dollars [840]** as the **Resource ID** and type −0.15 in **Scaled Amount**. Because the value for **Scaled Amount** is negative, this results in a 15% reduction. (A positive number would result in a fee.)
   f. Click **OK**.

6. Set up a second rate plan for the $10 total bill reduction plus the 30 free minutes. This reduction is applied only if the total for the current bill is over $100.
   a. Under **Rate Plan Selector**, type **$10 Reduction + 30 Minutes** as the name for the payment incentive.
   b. Click **Edit Plans** and click **New**.
   c. Define the **Plan Details** and **Rate Plan Structure**.
   d. Define the **Quantity Discount Bracket** for the new rate as **Based on: Rate Dependent**.
   e. In the **Balance Impacts** tab, deselect **Minimum** and type 100 in the associated entry box.
   f. Select **US Dollars [840]** as the **Resource ID** and type −10 in **Fixed Amount**.

   **Note:** This defines payment fees for all customer accounts. To define fees only for certain accounts, create a **Subscription** product and purchase the product for the account.
g. Add a row to the balance impacts table that sets **Free Domestic Minutes** as the **Resource ID** and enter 30 in **Fixed Amount**.

h. Click OK.

7. Set up the rate plan selector.

   a. Click ...+ in the first column, select **Event**, choose **PIN_FLD_INCENTIVE.PIN_FLD_PAY_TYPE** from the attributes list, and click OK.

   b. Click ...+ in the next column, select **Event**, choose **PIN_FLD_INCENTIVE.PIN_FLD_CUSTOMER_SEGMENT** from the attributes list, and click OK.

   c. Click + in the row column to create a new row for each of the two payment method/customer segment combinations. The system product does not provide incentives to any customers who do not meet one of these two criteria.

   d. For the first row, type 10003 to define credit card as the payment method and **Platinum Subscriber** to define the customer segment. Select the **15% Reduction** rate plan.

   e. For the second row, type 10011 to define cash as the payment method and **Silver Subscriber** to define the customer segment. Select the **$10 Reduction + 30 Minutes** rate plan.

   f. Click OK and Apply.

---

**Customizing Payment Incentives**

You can configure BRM to grant payment incentives. See "Configuring Payment Incentives".

To customize payment incentives, read the following:

- To learn about the standard payment incentive opcodes, see "How Payment Incentives Work".

- To customize how payment incentives are triggered, for example, by date, use the **PCM_OP_PYMT_POL_PROVISION_INCENTIVE** policy opcode. See "Customizing How to Trigger Payment Incentives".

- To customize how to grant payment incentives, use the **PCM_OP_PYMT_POL_GRANT_INCENTIVE** policy opcode. See "Customizing How to Grant Payment Incentives".

- To manually reverse payment incentives, see "Manually Reversing a Payment Incentive".

**How Payment Incentives Work**

Payment incentives are implemented by calling **PCM_OP_PYMT_PROVISION_INCENTIVE** and **PCM_OP_PYMT_GRANT_INCENTIVE**.

**PCM_OP_PYMT_PROVISION_INCENTIVE** evaluates a payment to determine whether a payment incentive should be provisioned and, if so, sets the payment incentive trigger. If a payment incentive is triggered, the **PCM_OP_PYMT_GRANT_INCENTIVE** performs the incentive.

For more information, see "How Payment Incentives are Triggered" and "How Payment Incentives Are Granted".
How Payment Incentives are Triggered

PCM_OP_PYMT_PROVISION_INCENTIVE is called by PCM_OP_BILL_ITEM_TRANSFER immediately after payment allocation, provided BRM is configured for payment incentives.

PCM_OP_PYMT_PROVISION_INCENTIVE determines whether the payment resulted in an early, in-full settlement of the last bill. If so, the current bill may be eligible for a payment incentive and PCM_OP_PYMT_POL_PROVISION_INCENTIVE creates a trigger for payment incentive processing to apply an incentive.

PCM_OP_PYMT_PROVISION_INCENTIVE performs the following functions:

1. It calls the PCM_OP_PYMT_POL_PROVISION_INCENTIVE policy opcode to determine if the PIN_EFFECTIVE_T field or any other customizable field contains a payment timestamp. See "Customizing How to Trigger Payment Incentives".

2. It retrieves all of the bills and determines whether each of the bills is from the last billing cycle or a prior cycle. PCM_OP_PYMT_PROVISION_INCENTIVE is only concerned with the bills from the last billing cycles; none of the other bills qualify for early incentives.

3. It reads the Due and Due Time in each /bill object. PCM_OP_PYMT_PROVISION_INCENTIVE uses this information along with the timestamp in the PIN_FLD_END_T field from the /event/billing/payment object or a timestamp from the policy opcode to determine whether the payment for a given bill was allocated in full and early. PCM_OP_PYMT_PROVISION_INCENTIVE must find the following conditions:
   - PIN_FLD_DUE must be 0, indicating that there are no more payments due for the bill, and it was paid in full.
   - The timestamp in PIN_FLD_END_T or the timestamp provided by the policy opcode must be earlier than or the same as the one in PIN_FLD_DUE_T, indicating that the payment was allocated before or at the same time as the due time. BRM considers both of these conditions to be indications of an early payment.

BRM always attempts to use whatever timestamp the policy opcode provides. PCM_OP_PYMT_PROVISION_INCENTIVE only uses PIN_FLD_END_T if the policy opcode doesn’t return a timestamp or returns PIN_FLD_END_T instead of some other timestamp.

4. If the payment meets these conditions, PCM_OP_PYMT_PROVISION_INCENTIVE modifies the /billinfo object by setting the /billinfo object by setting the PIN_FLD_PAYMENT_EVENT_OBJ to the POID of the payment event that resulted in early, in-full payment. This acts as a trigger for granting the payment incentive during the billing run.

5. It returns a list of /billinfo objects to which it added the payment incentive.

Customizing How to Trigger Payment Incentives

You can configure the PCM_OP_PYMT_POL_PROVISION_INCENTIVE policy opcode to determine the payment date that should be considered when provisioning incentives.

This policy opcode is called by PCM_OP_PYMT_PROVISION_INCENTIVE to provide the payment date that will be used when determining whether the bill was paid on time. It receives the POID of the /event/billing/payment object in the input flist and reads this object to determine the payment date. If it finds a payment date it's
configured to provide, it returns the date to PCM_OP_PYMT_PROVISION_INCENTIVE, which performs the following tasks:

- Retrieves the bill for which the payment allocation was made.
- Determines if the bill was the last bill.
- Compares the timestamp provided by the policy opcode to the due date for the payment.

By default, PCM_OP_PYMT_POL_PROVISION_INCENTIVE reads the PIN_FLD_END_T field to obtain the timestamp.

You can customize PCM_OP_PYMT_POL_PROVISION_INCENTIVE to provide the timestamp from a field other than PIN_FLD_END_T (for example, PIN_FLD_EFFECTIVE_T) or to apply business logic that determines the payment date.

For example, you can customize this opcode to use the payment receipt date as the payment timestamp for all credit card payments, and 3 days after the payment receipt date for all check payments.

You can also create custom payment objects that use fields other than PIN_FLD_EFFECTIVE_T or its equivalent. In this case, you would customize this policy opcode to read these fields and provide them as output for PCM_OP_PYMT_PROVISION_INCENTIVE.

### How Payment Incentives Are Granted

PCM_OP_PYMT_GRANT_INCENTIVE is called by PCM_OP_BILL_MAKE_BILL as part of the billing run. PCM_OP_PYMT_GRANT_INCENTIVE grants payment incentives based on:

- Whether the account has purchased a payment incentive subscription product or the account is eligible for a system product that includes a payment incentive.
- Whether the payment incentive trigger is set in the /billinfo object.
- Conditions specified in the rate plan.
- Any additional conditions specified in the PCM_OP_PYMT_POL_GRANT_INCENTIVE policy opcode.

PCM_OP_PYMT_GRANT_INCENTIVE performs the following functions:

1. If the bill qualifies for a payment incentive, the opcode uses information from the /account and /event/billing/payment objects to enrich the input flist with the payment method, payment channel, and customer segment list. It also includes:
   - The total for the current bill calculated during the current billing run.
   - The total for the last bill, as determined from the /bill object for that bill.
   By default, both these totals are after-tax amounts.

2. It calls the PCM_OP_PYMT_POL_GRANT_INCENTIVE policy opcode to determine whether it needs to enrich the input flist with any extra fields, and validates the fields returned by the policy opcode. See "Customizing Payment Incentives".

3. It creates an /item/incentive object and an /event/billing/incentive object for the bill. In addition to the payment method, payment channel, and so forth, the /event/billing/incentive object contains any fields specified in the PCM_OP_PYMT_POL_GRANT_INCENTIVE policy opcode, provided the object has been suitably extended.
4. It sends the event to the rating opcodes to calculate the payment incentive for each bill. BRM applies the balance impact of the payment incentive event to the default balance group of the bill unit.

5. It clears the payment incentive trigger in the PIN_FLD_PAYMENT_EVENT_OBJ field for the /billinfo objects of each affected bill, returning this field to a null value.

Two other standard opcodes are used for payment incentives:

- **PCM_OP_PYMT_PROVISION_INCENTIVE** triggers payment incentives. See "How Payment Incentives are Triggered".
- **PCM_OP_PYMT_REVERSE_INCENTIVE** reverses payment incentives. See "How Payment Incentives Are Reversed".

### Customizing How to Grant Payment Incentives

By default, you can set up a rate plan so that BRM considers three attributes when determining whether to apply a payment incentive:

- Customer segment
- Payment channel
- Payment method.

You can broaden this scope by customizing the PCM_OP_PYMT_POL_GRANT_INCENTIVE policy opcode. When you customize this opcode, you must also extend the /event/billing/incentive storable class.

You implement additional attributes that BRM can work with by customizing the PCM_OP_PYMT_POL_GRANT_INCENTIVE policy opcode to include a broader set of account and service attributes: for example, service type, currency type, and so forth. PCM_OP_PYMT_POL_GRANT_INCENTIVE augments the input flist by adding fields that PCM_OP_PYMT_GRANT_INCENTIVE includes when creating the /event/billing/incentive object.

If you include additional attributes in PCM_OP_PYMT_POL_GRANT_INCENTIVE, you must also extend the /event/billing/incentive object by adding these fields so that the object records all the criteria considered for the payment incentive. The additional fields are then included in the choices you can make when creating columns in the Pricing Center rate plan selector. Therefore, you can use these additional fields as criteria when defining attribute combinations that result in payment incentives.

The contents of the /event/billing/invoice object determine:

- The list of attributes that can be considered when setting up a rate plan. The pricing expert associates combinations of these attributes with specific rate plans when creating the payment incentive product.

- The information BRM compares with the attribute combinations defined for the payment incentive product. If BRM finds a match, it rates the payment incentive event according to the rate plan for the matching combination.

For example, to award a payment incentive to premium customers who pay in a certain currency, you perform two tasks:

- Customize PCM_OP_PYMT_POL_GRANT_INCENTIVE by adding the PIN_FLD_SERVICE_OBJ and PIN_FLD_CURRENCY fields to the input flist. The opcode then includes the service type for the account and the currency type in the information sent to the real-time rating opcodes. As a result, the rating opcodes are able to consider the service type and currency along with the usual criteria when
determining whether to apply the incentive and when calculating the payment incentive.

- Extend the `/event/billing/incentive` class by adding the PIN_FLD_SERVICE_OBJ and PIN_FLD_CURRENCY fields. BRM then includes the service type and currency in the event object each time it’s created. When you create columns in the rate plan selector, you can select these two fields along with the default fields.

In addition to performing this type of customization, you can also customize the PCM_OP_PYMT_POL_GRANT_INCENTIVE policy opcode to do the following:

- Provide special types of incentives such as free gifts. In this case, you modify the opcode so that it calls the opcodes that process and control product purchases, for example PCM_OP_SUBSCRIPTION_PURCHASE_DEAL. Then, you create a deal that includes an item product that awards the free gift.
- Develop an aggregation application to count the number of subscription services for an account and customize PCM_OP_PYMT_POL_GRANT_INCENTIVE to include this information in the enriched flist.
- Customize which customer segment to use. You can also customize this opcode to select a different customer segment from PIN_FLD_CUSTOMER_SEGMENT_LIST. By default, PCM_OP_PYMT_POL_GRANT_INCENTIVE selects the first customer segment from the PIN_FLD_CUSTOMER_SEGMENT_LIST it gets from PCM_OP_PYMT_GRANT_INCENTIVE. The customer segment returned by PCM_OP_PYMT_POL_GRANT_INCENTIVE is the one that BRM uses as a filtering attribute during payment incentive calculation.

### How Payment Incentives Are Reversed

PCM_OP_PYMT_REVERSE_INCENTIVE reverses a payment incentive, provided that it has not yet been granted.

PCM_OP_PYMT_REVERSE_INCENTIVE is called by PCM_OP_BILL_REVERSE_PAYMENT as part of payment reversal, provided BRM is configured for payment incentives. It determines whether payment is being reversed for a bill that had a payment incentive either provisioned or granted. If so, it either deactivates the payment incentive trigger or, for payment incentives that have already been granted, issues warnings and flags the reversal event so that you can initiate manual processing.

PCM_OP_PYMT_REVERSE_INCENTIVE performs the following functions:

1. It retrieves all bills affected by the reversal and searches for the associated `/billinfo` objects.
2. For each `/billinfo` object, it checks the PIN_FLD_PAYMENT_EVENT_OBJ field to determine whether a payment incentive has been provisioned but not yet granted. If so, it resets this field, removing the POID of the payment event, which automatically reverses the payment incentive.
3. If the PIN_FLD_PAYMENT_EVENT_OBJ field does not indicate that a payment incentive has been provisioned, the opcode searches all `/event/billing/incentive` objects for the account to determine whether any of them are associated with the bill for which payment is being reversed.

The existence of this object for a given bill means that BRM already granted a payment incentive for the bill. In this case, the opcode generates a warning in the cm.pinlog file to indicate that manual adjustment is required.
Manually Reversing a Payment Incentive

When a payment is reversed, BRM reverses any payment incentive provisioning triggers created at payment time, provided the payment was for the last bill. If payment was reversed for an earlier bill, BRM has already applied the incentive and does not reverse it. In this case, you need to perform a manual account adjustment through your CRM client application. This type of adjustment debits account resources rather than crediting them.

BRM identifies payment incentives that need manual reversal in the cm.pinlog file, which lists any cases where a payment incentive reversal failed. Searching the cm.pinlog for this information can be time consuming. Therefore, you should consider customizing this process in one of the following ways:

■ Write your own reporting application that creates a list of all bills that meet the following two conditions:
  – The bill had a payment incentive that was not only provisioned, but also granted.
  – The payment reversal was for the bill prior to the one that had the payment incentive granted.

Operations personnel can use this report to identify the adjustments they need to perform.

■ Create your own custom opcode or customize the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode so that it checks the reporting conditions discussed above and alerts the CRM client application whenever both conditions are true. Then, enable event notification and add the following information to your system’s event notification list so that BRM calls PCM_OP_ACT_POL_EVENT_NOTIFY (opcode number 301) each time an /event/billing/reversal/* event is generated during payment reversal:

```
# comment, if any
301 0 /event/billing/reversal/cc
301 0 /event/billing/reversal/check
301 0 /event/billing/reversal/dd
301 0 /event/billing/reversal/payorder
301 0 /event/billing/reversal/postalorder
301 0 /event/billing/reversal/transfer
```

For more information, see "Using Event Notification" in BRM Developer's Guide.

In addition, you need to customize your middleware and CRM applications to process the notification correctly and issue appropriate messages to operations personnel.
This chapter provides an overview of Payment Suspense Manager and how your Oracle Communications Billing and Revenue Management (BRM) system handles suspended payments. It includes:

- A summary of the Payment Suspense Manager functionality.
- An overview of how BRM determines the status of a payment and how it suspends payments.
- Information on how to set up BRM for Payment Suspense Manager.

**Note:** Payment Suspense Manager is available only for externally initiated payments; for example, check payments that are sent from a bank. It does not apply to BRM-initiated payments because the problems that trigger payment suspense cannot occur in BRM-initiated payments.

For background information on payments and payment processing, see "About Payments". For information on the client applications that support Payment Suspense Manager, see:

- Payment Tool Help
- Payment Center Help

**About Payment Suspense Manager**

Payment Suspense Manager lets you more effectively handle payments that cannot be posted immediately to customer accounts due to insufficient information, account closure, or other criteria, and payments that were posted incorrectly to customer accounts. It is a two-way process between the payment suspense account and a customer account: you can apply suspended payments to customer accounts or suspend payments that have been posted in customer accounts.

With Payment Suspense Manager, BRM *automatically* suspends payments that exhibit certain problems instead of failing them or wrongly allocating them. It removes the payment from the payment processing stream, postponing it for later investigation. This enables the payment posting process to complete without requiring immediate intervention to fix the errors.

Payment Suspense Manager also enables you to perform the following payment processing tasks:
Manually suspend payments during payment processing. If you find a successful payment in a payment batch that you suspect contains incorrect data or requires special handling, you can manually suspend that payment so that it can be carefully examined before it is posted to the account. Such payments can be processed later, so you can avoid performing the allocations during normal payment collection.

Manually suspend payments after they have been posted to customer accounts. If a payment was posted incorrectly, you can suspend it and then repost it to the correct account: you no longer need to use Payment Tool or a custom CRM application to reverse a posted payment from the BRM database and then resubmit it as a new payment.

Allocate suspended payments to one or more accounts. See "About Distributing One Payment to Multiple Accounts".

Partially allocate a suspended payment so that an amount remains in the payment suspense account. If all or part of a suspended payment cannot be allocated, you can remove the suspended payment or unallocated amount from the BRM system so that you can recognize the unallocated revenue. This enables you to track the unrealized revenue in your general ledger (G/L) system.

Create financial reports on revenue that you have realized but that remains unallocated. Suspended payments are assigned to their own G/L segment, so you can develop reporting facilities that isolate suspended payments and quantify that type of revenue.

Allocate an account-level payment to multiple bill units of a customer’s account. See "Allocating Account-Level Payments to Multiple Bill Units".

Payments can remain suspended indefinitely. This gives you the flexibility of fixing and applying them to the correct accounts at any time. You can move payments back and forth between customer accounts and the payment suspense account any number of times.

Suspended Payment Processing Overview

The payment suspense process begins when you collect payments from a financial institution: whether you use payment clerks to manually post payments from batch files or use a third-party payment gateway to automatically post payments. It is a three-step process:

1. **Validate the payment.**
   Determines whether there is enough information for BRM to post the payment successfully or if the payment should be suspended.

2. **Submit the suspended payment to the payment suspense account.**
   Moves a suspended payment into the BRM payment suspense account, a special account you set up to store all suspended payments. You can also suspend payments that have already been validated and posted to customer accounts.

3. **Correct the suspended payment.**
   Corrects the problem and moves the payment from the payment suspense account to the accounts for which the payment should be made.

You use two client applications to work with suspended payments: Payment Tool and Payment Center. As a general rule, you use Payment Tool to validate incoming payments, manually suspend payments before they get posted to customer accounts, and submit payments to the BRM database. You use Payment Center to manually
suspend payments that were incorrectly posted to customer accounts and to correct suspended payments.

How you work with these applications depends on whether you receive the payment as a batch file from the bank or use a payment gateway that has been directly integrated with BRM payment services.

For details on the payment suspense process, see "About the Payment Suspension Process". For more information on how to work with Payment Tool and the payment gateway in relation to Payment Suspense Manager, see "About Payment Suspense Manager and Client Applications".

About Setting Up Payment Suspense Manager

Payment Suspense Manager is an optional BRM feature. By default, this feature is disabled. To enable payment suspense, you modify the appropriate /config/business_params storable object. For more information, see "Enabling Payment Suspense in BRM".

After you enable Payment Suspense Manager, you must perform the following tasks:

- **Set up a BRM payment suspense account.**
  
  The payment suspense account is an ownerless account that stores each suspended payment along with details such as credit card number, payment channel, payment method, and so forth.

  If, during validation, BRM determines that a payment should be suspended, it performs one of these actions:

  - If the payment was submitted through a third-party payment gateway, it automatically moves the payment to the payment suspense account.
  
  - If you are working with a payment in Payment Tool, it moves the payment to the payment suspense account when your payment clerk submits the payment batch to the BRM database.
  
  - If you are working with Payment Center, it moves the payment to the payment suspense account when your payment clerk performs an undo allocation operation on a posted payment.

  After a suspended payment is corrected, BRM moves the payment from the payment suspense account to the correct customer account.

  The payment suspense account is assigned to a unique G/L segment, and the payments in the payment suspense account will only belong to this G/L segment: not the G/L segment to which the customer accounts belong. Thus, while the items in the payment suspense account are excluded from normal financial reporting, billing, and collection, you can obtain information on the associated revenue by generating financial reports for the G/L segment to which the payment suspense account belongs.

  For more information, see "Creating the Payment Suspense Account".

- **Define suspense reason codes and action owner codes.**

  *Note:* By default, the payment suspense account is included in the billing run. For information on how to suspend billing on the payment suspense account, see "Suspending Billing of /billinfo Objects" in BRM Configuring and Running Billing.
The reason codes and action owner codes provide information on why a payment is suspended and who is assigned to correct the problem. To help payment clerks understand the nature of a suspended payment, BRM displays descriptions associated with reason codes and action owner codes in Payment Tool and Payment Center. In addition, BRM uses reason codes and action owner codes to populate selection lists in these tools.

**Note:** You can also create additional search fields in Payment Center and additional batch types in Payment Tool. For information on modifying payment options in these tools, see the Programmer’s Guide.

For more information, see "Working with Suspense Reason Codes and Action Owner Codes"

- **Define any custom suspense rules.**

  BRM validates a payment by determining whether the payment has certain mandatory information such as the correct account number and bill number. You can customize BRM such that it considers extra validation criteria like the customer segment. You can also define the conditions under which you submit a suspended payment to the payment suspense account: for example, a minimum payment amount.

  For more information, see "About Customizing Payment Suspense Manager".

---

**About the Payment Suspension Process**

Payment suspension begins when you collect payments from a financial institution: whether you use payment clerks to manually post payments from batch files or you use a third-party payment gateway to automatically post payments. When payments are received, BRM validates payments based on the following criteria to determine whether they should be suspended rather than posted immediately to customer accounts:

- The account number and bill number are missing or incorrect.
- The account number and bill number are correct, but the bill belongs to a different account.
- The account is closed.
- The payment attributes: for example, the customer segment or payment amount: do not comply with your custom BRM validation rules.

The payment processing phase involves three steps: validation, suspension, and correction. These steps are sequential and rely on the completion of the prior step.

1. **Validation:** BRM determines whether a payment meets the validation criteria and assigns a status of successful or “to be suspended.” BRM takes the following actions:
   - If the payment is successful, BRM posts the payment to the account.
   - If the payment does not meet the validation criteria but has enough information to qualify for suspense, BRM marks it as “to be suspended” and forwards it to the opcodes responsible for suspending the payment.

   BRM can suspend both successful and financially failed payments. For example, a payment batch includes two check payments, each with an
incorrect account number. The payment information indicates that one check has cleared and the other bounced.

Coming into BRM, the first payment would be considered successful and the second, failed. When BRM validates the payments, both would be marked for suspense because, regardless of the financial success or failure of the payment, neither payment can be posted to the correct account.

■ If the payment does not meet the validation criteria and also does not qualify for suspense, BRM informs you that the payment cannot be posted. You need to create an exception batch to handle payments that fall into this category.

Payment validation is initiated automatically through the payment gateway or manually by a payment clerk.

For details on payment validation, see "About Payment Validation".

2. Suspension: BRM moves the payment to the payment suspense account and creates the associated events and items to store information on the suspended payment.

There are two distinct situations in which payment suspense can occur: during payment processing, when a payment batch is submitted to the BRM database, and during account maintenance, after payments have been saved to the BRM database.

■ During payment processing, payment suspense is initiated automatically through a third-party payment gateway or manually by using Payment Tool. It is initiated in Payment Tool when you submit a payment batch that includes payments marked for suspense. Such payments can be successful payments that you manually mark for suspense because you suspect they have a problem or you know they require manual allocation. For details on suspending unposted payments, see "About Processing Suspended Payments in a Payment Batch".

■ During account maintenance, payment suspense is initiated manually by using Payment Center. Payment suspense is initiated when you undo the allocation of a payment from a customer account. For details on suspending posted payments, see "About Processing Suspended Payments in the BRM Database".

3. Correction: To correct a suspended payment, you use Payment Center to assign it to a correct account number or bill number and apply it to the customer account. You can also create a distribution list for a suspended payment, which enables you to apply the payment to multiple accounts.

After payment analysts correct suspended payments and assign them to one or more accounts, the payments must be validated again. If the payment validation is successful, BRM posts the payments to the correct accounts. If the suspended payment is allocated completely (an amount does not remain in suspense), BRM reverses the suspended payment, removing it from the payment suspense account. While performing this operation, BRM creates the required objects and events.

Note: Payment correction is always initiated by a payment clerk through Payment Center; this step is never automatic. If, during revalidation, the payment still meets the suspense criteria, BRM again assigns a status of suspended and the payment is resubmitted to suspense.
For details on payment correction, see "About Payment Correction".

Figure 8–1 shows the steps involved in payment suspension and the basic operations they perform:

**Figure 8–1 Basic Operations and Steps Involved in Payment Suspension**

1. **Validate the payment**
   - Check the payment to determine if it is valid.
   - Assign a status of successful, suspended, or failed-suspended.

2. **Suspend the payment**
   - Move the payment to the payment suspense account.
   - Create a payment event and item.

3. **Correct the payment**
   - Find and correct missing or erroneous information.
   - If the payment is valid, reverse the payment to the payment suspense account. Create the reversal event and item.

---

**About Payment Validation**

Before payments can be processed and posted by BRM, they must pass validation. BRM uses the PIN_FLD_STATUS value to verify that the PCM_OP_PYMT_VALIDATE_PAYMENT opcode and the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode successfully performed the payment validation and to determine how to handle the payment. When Payment Suspense Manager is enabled, BRM updates the PIN_FLD_STATUS value as follows:

- **Successful payments that pass the validation process**: Retains the original PIN_FLD_STATUS of PIN_PYMT_SUCCESS. These payments are loaded into BRM and allocated to the correct account according to the default BRM payment processing behavior.

- **Failed payments that pass the validation process**: Retains the original PIN_FLD_STATUS of PIN_PYMT_FAILED. These payments are loaded into BRM and posted to the correct account using the default BRM payment processing behavior.

- **Payments that fail the validation process**: Updates PIN_FLD_STATUS to PIN_PYMT_SUSPENSE, PIN_PYMT_FAILED_SUSPENSE (for failed payments that need to be suspended), or any other status in the suspense range. These payments require further investigation, which you can perform immediately or defer by submitting the payments to BRM. Upon submission, BRM passes all payments marked for suspense to the PCM_OP_PYMT_POL_SUSPEND_PAYMENT policy opcode, which is responsible for actually directing them to suspense.
For more information about payment status, see "About Payment Status".
For information on the suspense validation opcodes, see "Payment FM Standard Opcodes" and "Payment FM Policy Opcodes" in BRM Developer’s Reference.

About Processing Suspended Payments in a Payment Batch
If you are using Payment Tool and a payment has been marked for suspense, you can:

■ Investigate and correct the problem before submitting the payment batch to the BRM database. If you do so, the payment is not actually suspended. Instead, it is revalidated and posted to the correct account.

■ Defer the investigation until later by submitting the payment to the database by using Payment Tool. In this case, the payment is suspended, and BRM moves it to the payment suspense account.

Note: If you use a payment gateway to process your payments, all payments marked for suspense are automatically submitted to suspense.

Whether you use Payment Tool or a payment gateway, BRM calls the PCM_OP_PYMT_POL_SUSPEND_PAYMENT policy opcode to process any payment submitted with the status PIN_PYMT_SUSPENSE. The opcode places a payment in suspense by enriching the flist so that the payment can be directed to the payment suspense account. It then calls the opcodes that post the payment to this account and create objects to record the suspended payment. The event object contains all information about the payment and its suspense, including the account number, bill number, transaction ID, and any associated reason codes or action owner codes. This information can be used to investigate why the payment failed the validation process and who is responsible for resolving the problem.

About Processing Suspended Payments in the BRM Database
To suspend payments that are posted in customer accounts, you use Payment Center to perform an Undo Allocation operation and to assign a reason for suspense. The PCM_OP_PYMT_RECYCLE_PAYMENT opcode validates that the source account is the account to which the payment was posted and that the destination account is the payment suspense account. If both requirements are true, the payment is recycled to suspense. See "About Recycling Payments to Suspense".

Payment Center sets the payment status to PIN_PYMT_RETURNED_SUSPENSE, indicating that allocation has been undone for this payment at least once. BRM uses this flag to identify the payment when you search for a suspended payment in Payment Center. See "About Payment Status".

When you suspend a payment that was posted to a customer account, the entire payment is reversed from the customer account. You cannot suspend a partial payment amount. For example, a payment is allocated to two correct bills and one incorrect bill. The entire payment allocation must be undone to correct the error, not just the amount allocated to the incorrect bill. To undo the allocation, the payment is suspended. Then, it can be reallocated to the correct account or bills.
About Payment Suspense Manager

If you are resuspending a payment that was distributed to multiple accounts, you can resuspend any number of the distributed payments at one time.

You use Payment Center to suspend payments that were posted to customer accounts. For more information, see Payment Center Help.

You use Customer Center to perform adjustments, if necessary. For more information, see Customer Center Help.

About Payment Correction

If a payment is suspended, you must correct the problem. If you are working with a payment that was already submitted to the BRM database, you must also resubmit the payment after correcting it.

You investigate and correct suspended payments by using Payment Center. After you update a payment with the corrected information, the action you take depends on how you opened the payment you are working on: from a current payment batch by using Payment Tool, or from the BRM database by using Payment Center. If the payment is in a current payment batch in Payment Tool, you must return to Payment Tool to revalidate it and submit the batch.

However, if the payment was already in the BRM database, Payment Center uses PCM_OP_PYMT_RECYCLE_PAYMENT to validate and allocate the corrected payment to the proper account. For valid payments, BRM searches for the original payment details and moves the corrected payment from the payment suspense account to the correct account by:

- Posting the payment to the correct account and allocating the payment to the correct bill.
- Reversing the original suspended payment stored in the payment suspense account.

If the corrected payment does not pass revalidation, BRM again suspends the payment and assigns a new reason code to describe the problem.

In some cases, you may determine that a suspended payment cannot be fixed: for example, the payment has stayed in suspense after repeated correction attempts. Payments of this nature represent unrealized revenue. If you want to track revenue and report for these payment, you can remove them from suspense as unallocatable. When you do this, BRM uses the PCM_OP_BILL_REVERSE_PAYMENT opcode to reverse the payment in the payment suspense account and create the associated objects in the database. BRM assigns a G/L ID of 112 for the reversal, placing the payment amount in a special G/L bucket so that you can obtain information about how much unallocatable revenue you have.

About Distributing One Payment to Multiple Accounts

You can divide a suspended payment into any number of distributed payments and apply each one to a different customer account. Distributed payments are subpayments that together comprise one larger suspended payment.
In the example shown in Figure 8–2, a $3,000 suspended payment is portioned into three distributed payments of $1000, $450, and $650, each of which is posted to a different customer account:

**Figure 8–2  Distributing One Payment to Multiple Accounts**

![Diagram of distributed payments](image)

Each distributed payment can be allocated to specific bills, items, or both, in an account, or left unallocated at the account level. When distributing a suspended payment to a group of accounts, you can mix allocation levels so that some distributed payments are allocated to accounts and some to bills. For underpayments, you can also allocate the payment to specific items in an account.

For example, in the scenario shown in Figure 8–3, Suspended Payment 1 is distributed from the suspense account to three accounts. As indicated by the shaded boxes below, the distributed payment is left unallocated in Account A and allocated to specific bills in Account B. The underpayment to Account C is allocated to specific items.

**Figure 8–3  Allocation of Distributed Payments from a Suspended Payment**

![Diagram of distributed payments and allocations](image)
You can specify only one payment allocation level for each target account in the
distribution list. For example, a payment analyst cannot choose to apply a payment to
Account A and then try to allocate it to specific bills in Account A. Likewise, if a
payment analyst chooses bill-level allocation for Account B, he or she cannot later
apply that payment at the account level for Account B.

You use Payment Center to process distributed payments. The procedure involves:
1. Opening a suspended payment that requires distribution.
2. Creating a payment distribution list and identifying the customer account, bills, or
   items to which each distributed payment belongs.
3. Submitting the payment distribution list to BRM.

If you discover that one or more distributed payments were allocated incorrectly, you
can return them to suspense and redistribute them at any time.

For more information about Payment Center and distributed payments, see
"Managing Suspended Payments".

About Allocating an Account-Level Payment to Multiple Bill Units

For accounts having multiple bill units, BRM distributes the account-level payment to
multiple bill units of the account. Payment Suspense Manager is used to suspend the
original payment and recycle it to distribute to multiple bill units. The distribution
process is similar to distributing a single suspense payment to multiple accounts. Each
distributed payment has the original payment’s transaction ID as a subtransaction ID.
For accounts with multiple bill units, BRM calls the PCM_OP_PYMT_COLLECT
opcode to allocate the payment.

For more information on how BRM allocates payment to multiple bill units, see
"Allocating Account-Level Payments to Multiple Bill Units".

Understanding Payment Recycling

The payment suspension functionality involves payment recycling, which is the method
by which BRM reuses payment information. BRM uses payment recycling when
distributing suspended payments to customer accounts, returning a payment to
suspense, and placing a posted payment into suspense.

Payment recycling is a transactional process consisting of two sequential operations:
1. Reversing a payment from a source account.
2. Applying a payment to a target account.

Payment recycling occurs each time you move a payment from a customer account to
the payment suspense account or from the payment suspense account to a customer
account. When a payment is recycled, the current payment is reversed, and all of the
payment information, except for the original payment date, is transferred to the new
recycled payment. For more information on payment reversal, see "How Payment
Reversals Work with Suspense and Recycling".

BRM uses the PIN_FLD_STATUS field of a payment to validate payments before they
are recycled and submitted to an account. The status code is returned by Payment
Center and can be configured. For a complete list of default payment status codes and
descriptions, see "About Payment Status".

Payments can be recycled any number of times; however, each recycled payment can
be traced back to only one original payment. BRM uses the payment’s transaction ID
and subtransaction ID to track their origin and descendants. For more information, see
"About Original Payments" and "How BRM Tracks Suspended Payments".

When performing reversals during recycling, the PCM_OP_BILL_REVERSE opcode must be called by PCM_OP_PYMT_RECYCLE_PAYMENT. This ensures that only payments with a SUB_TRANS_ID value of NULL can be reversed directly.

For more information, see "How Payments Are Recycled to and from Suspense".

**About Original Payments**

An original payment is the first instance of a payment that is saved to BRM, either through Payment Tool or a payment processor. Original payments can be saved initially to the suspense account or to a customer account. The distinguishing characteristic of an original payment is that it has never been recycled.

Because an original payment is the first instance of a payment, it does not have a subtransaction ID, which is an ID used to trace a recycled payment back to its original payment.

An original payment can be the ancestor of one or more recycled payments, but a recycled payment can be traced back to only one original payment. Both suspended payments and those posted in customer accounts can be original payments.

For more information on subtransaction IDs and how payments are tracked in BRM, see "How BRM Tracks Suspended Payments".

**About Payment Transfer Direction and Verification**

When PCM_OP_PYMT_RECYCLE_PAYMENT receives a distribution list from Payment Center, it uses the information in the CHARGES array of the input flist to determine the direction of the payment transfer: from the payment suspense account to a customer account, or to the payment suspense account from a customer account. The CHARGES array contains two fields that determine the direction of the transfer:

- PIN_FLD_EVENT_OBJ: This field identifies the payment event. The event, in turn, identifies the **source account**, the account that owns the payment. This is the account from which the opcode transfers payments.

- PIN_FLD_ACCOUNT_OBJ: This field identifies the **target account**, the account to which the opcode transfers payments.

If a payment is being moved from suspense, the PIN_FLD_EVENT_OBJ field establishes the payment suspense account as the source account, and the PIN_FLD_ACCOUNT_OBJ field contains one or more POIDs of valid customer accounts. If a payment is being moved to suspense, the PIN_FLD_EVENT_OBJ field establishes a valid customer account as the source account, and the PIN_FLD_ACCOUNT_OBJ field contains the POID of the payment suspense account.

Before recycling a payment, BRM verifies that there are no conditions present that prevent the transfer. These conditions include:

- The currency for the source account and all target accounts is not the same.

- The source and target accounts are both customer accounts.

- Multiple suspended payments are recycled in the same operation.

- You attempt to partially suspend a payment originally posted to a customer account. You can only perform this action if you suspend the **entire** payment.

- You attempt to distribute a failed payment from the suspense account into a customer account.
You attempt to return a distributed payment from a customer account to the suspense account, but the suspended payment is also a failed payment. This can happen when you distribute a suspended payment to customer accounts and, afterwards, the payment fails for financial reasons: for example, the bank will not honor the check. In this case, you cannot return the distributed payment to suspense.

About Recycling Payments from Suspense
When you post a suspended payment to a customer account, the following operations occur if validation and transfer verification are successful:

1. The suspended payment is reversed (an /event/billing/reversal object is created).
2. The payment item for the suspended payment is closed.
3. A recycled payment is posted in the customer account. The payment status reflects that the payment was successfully recycled. See “About Payment Status”.
4. Bills, bill items, or both in the account might be closed, depending on the payment allocation level.
5. A new suspended payment is created for any unallocated amount (partially allocated payments).

You do not have to fully allocate a suspended payment. If you partially allocate a suspended payment and an amount remains in suspense, you can apply the remaining amount to any account at any time.

About Recycling Payments to Suspense
When you suspend a payment that has been posted in a customer account, the following operations occur if validation is successful:

1. The posted payment is reversed (an /event/billing/reversal object is created).
2. Any accounts receivable items (/item/payment) that were previously closed by the payment are reopened.
3. A recycled payment is created in the payment suspense account. The payment status reflects that the payment was successfully recycled. See “About Payment Status”.
4. A new billable item for the recycled suspended payment is created.

How Payment Reversals Work with Suspense and Recycling
Payments are reversed in BRM for a variety of reasons, the most common of which is that the funds for a payment are never actually deposited: for example, when a check does not clear. Payments also can be reversed as part of the suspense process. When a payment is reversed, two things happen:

- Any bills or items previously closed by the payment are reopened.
- The payment is deactivated in the BRM system.

There are three types of reversals in BRM. The first two, indirect reversal and reversal to remove an unallocatable suspended payment, are related to the suspense process. The third, direct reversal, is not.

- **Reversals that occur indirectly during the recycling process**
  
  *Indirect reversals* occur when you transfer a suspended payment to a customer account or from a customer account to suspense. With an indirect reversal, the
payment is removed from the source account and moved to the target account, resulting in a complete reversal of the payment in the source account so that the payment information can be transferred to the destination account as a recycled payment. If the payment being recycled had been posted in a customer account, any bills and items that were closed due to the payment are reopened.

---

**Note:** To reverse a batch of payments from the BRM system, use Payment Tool. For more information, see "About Directly Reversing Payments from BRM".

---

Indirect reversals are assigned a G/L ID of 113, placing the payment amount in a special G/L bucket so that you can keep a separate record of these reversals. For more information on G/L IDs and reversals, see "Working with Suspense Reason Codes and Action Owner Codes".

For more information about reversals due to recycling, see "Understanding Payment Recycling".

- **Reversals that occur when you remove a payment as unallocatable**

  If a suspended payment cannot be fixed but you want to track revenue for these payments and get reports on how much unallocatable revenue you have, you can remove them from suspense as unallocatable. When you do this, BRM reverses the payment in the payment suspense account and assigns it to a special G/L ID bucket. These reversals are rare in BRM. Even though they are part of suspense, they occur outside of the recycling process.

  For more information about removing payments as unallocatable, see "About Removing Unallocatable Payments from Suspense".

- **Reversals that you perform directly by using Payment Tool or a third-party application**

  *Direct reversals* occur when you use Payment Tool or a third-party application to reverse a payment that was recorded in the BRM database but never actually deposited. Direct reversals are the most frequent type of reversal in BRM, and they occur outside of the suspense and recycling processes.

  Direct reversals reverse an active payment completely and reopen any bills and items so the payment can be made again. Unlike reversals that occur during recycling, direct reversals are not initiated by the creation of a new recycled payment.

  For more information about direct reversals, see "About Directly Reversing Payments from BRM".

---

**How BRM Tracks Suspended Payments**

All payments and payment reversals contain a transaction ID (TRANS_ID field), which is a unique identifier that enables you to track each payment and reversal in BRM:

- For *payments*, the transaction ID identifies the payment transaction from the third-party payment processor. If a payment is submitted to BRM without a transaction ID, one is assigned to it.
- For *reversals*, the transaction ID is generated by BRM.

---
In addition to having a transaction ID, payments and reversals have another ID that is used to track all actions performed on a specific payment: a subtransaction ID (SUB_TRANS_ID field) for payments, and a paytransaction ID (PAYMENT_TRANS_ID field) for reversals. These ID values are used to find all payments and reversals that occur due to the recycling of an original payment.

- The SUB_TRANS_ID value of an original (never-recycled) payment is **NULL**.
- The SUB_TRANS_ID value of a recycled payment is equal to the transaction ID of its original payment.
- The PAYMENT_TRANS_ID value of a payment reversal is equal to the transaction ID of the payment it reversed.

In the example shown in **Figure 8–4**, a $3000 payment is suspended when it first arrives in BRM so that it can be posted to individual accounts within the corporation. The original payment is represented in white.

The payment analyst first partially distributes the original suspended payment to two accounts. The first distributed payment is for $1000 to Account A and the second is for $700 to Account B. The remaining $1300 is resuspended. The results of this operation are represented in Phase 1.

Later, the payment analyst realizes that the distributed payment made to Account B was erroneous and puts the $700 payment back into suspense, leaving $1000 in Account A and $2000 in the suspense account. The results of this operation are represented in Phase 2.
**Figure 8–4  Suspended Payment Distribution Example**

**Phase 1**
You distribute part of the original suspended payment to two customer accounts, resulting in Distributed Payment A and Distributed Payment B. The remainder stays in suspense.

<table>
<thead>
<tr>
<th>Original Payment</th>
<th>Suspended Payment 1</th>
<th>Suspended Payment 2</th>
<th>Suspended Payment 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3,000</td>
<td>$1,300</td>
<td>$1,300</td>
<td>$2,000</td>
</tr>
<tr>
<td>TRANS_ID: s1</td>
<td>TRANS_ID: s2</td>
<td>TRANS_ID: s3</td>
<td>TRANS_ID: s1</td>
</tr>
<tr>
<td>SUB_TRANS_ID: NULL</td>
<td>SUB_TRANS_ID: s1</td>
<td>SUB_TRANS_ID: s1</td>
<td>SUB_TRANS_ID: s1</td>
</tr>
</tbody>
</table>

**Phase 2**
You decide to resuspend Distributed Payment B. BRM reverses the payment from the customer account and recycles it back into suspense.

<table>
<thead>
<tr>
<th>Reversal 1</th>
<th>Reversal 2</th>
<th>Reversal 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$3,000</td>
<td>$700</td>
<td>$1,300</td>
</tr>
<tr>
<td>TRANS_ID: r1</td>
<td>TRANS_ID: r2</td>
<td>TRANS_ID: r3</td>
</tr>
<tr>
<td>PAY_TRANS_ID: s1</td>
<td>PAY_TRANS_ID: s2</td>
<td>PAY_TRANS_ID: s1</td>
</tr>
</tbody>
</table>

In this figure, all payments except the original payment are recycled payments. Notice that:

- Both distributed payments have the same subtransaction IDs (s1). The SUB_TRANS_ID value is the same as the TRANS_ID value of the original payment.

- All of the recycled payments have SUB_TRANS_ID values that match the original payment’s TRANS_ID values.

- Each reversal has a PAYMENT_TRANS_ID value that is equal to the TRANS_ID value of the payment it reversed.

When an original suspended payment is applied to a customer account, the following operations occur:

1. The suspended payment is reversed, and an `/event/billing/reversal` object is created. The reversal object’s PAYMENT_TRANS_ID value is equal to the TRANS_ID value of the suspended payment.

2. A new payment (`/event/billing/payment/pay_type` object) is applied to the customer account and contains the following information:

   - The original suspended payment’s details, including a value for any amount remaining in suspense.

   - A SUB_TRANS_ID value that is equal to the suspended payment’s TRANS_ID value.
3. If an amount remains in suspense, a new suspended payment is created for that amount; the original payment is not adjusted with a new amount. Instead, the new suspended payment receives a SUB_TRANS_ID value that is equal to the original payment’s TRANS_ID value.

Likewise, when a suspended payment is partially allocated to one or more accounts, and then an allocated payment is resuspended, the current suspended payment that was partially allocated is reversed and a new suspended payment containing the new amount is created.

**Note:** An original payment does not have to be a suspended payment. For example, if a payment was posted successfully to a customer account when it was received by BRM, it is an original payment. If it is later suspended and then reposted to customer accounts, both the suspended payment and the posted payment will have SUB_TRANS_ID values that match the TRANS_ID value of the original successful payment as seen in Figure 8–5:

---

**Figure 8–5  SUB_TRANS_ID of a Suspended Payment**

![Diagram showing SUB_TRANS_ID of a Suspended Payment](image)

- Original payment
  - Posted Payment A
    - $3000
    - TRANS_ID: pA
    - SUB_TRANS_ID: NULL

- Reversal 1
  - $3000
  - TRANS_ID: r1
  - PAY_TRANS_ID: pA

- Suspended Payment 1
  - $3000
  - TRANS_ID: s1
  - SUB_TRANS_ID: pA

- Distributed Payment B
  - $1000
  - TRANS_ID: pB
  - SUB_TRANS_ID: pA

- Reversal 2
  - $3000
  - TRANS_ID: r2
  - PAY_TRANS_ID: s1

- Distributed Payment C
  - $2000
  - TRANS_ID: pC
  - SUB_TRANS_ID: pA

- Payment Suspense Account

- Payment A and Suspended Payment 1 are no longer active.

---

**How Direct Reversals and Refunds Relate to Suspense**

BRM distinguishes between original payments, suspended payments, and recycled payments when processing direct reversals and refunds. Depending on the state of the payment, BRM may or may not be able to perform these activities.

**About Directly Reversing Payments from BRM**

Payment reversals are necessary when a payment is recorded in the BRM database but the payment is not deposited. You create payment reversal batches by using Payment Tool.
When you have Payment Suspense Manager enabled, payments may be reversed due to payment recycling, which is different from directly reversing the payment from the BRM database. Reversals that occur due to recycling happen automatically, and the funds are transferred from a source account to a target account: they are not removed completely from BRM.

The type of payment reversal discussed here is different from recycled payment reversals. Here, you manually reverse a payment to remove it from the system. A recycled payment reversal is a consequence of transferring a payment to a target account.

The following restrictions apply when reversing payments from BRM:

- You cannot directly reverse recycled payments: you can reverse only original (non-recycled) payments with a SUB_TRANS_ID value of **NULL**. Therefore, when creating the payment reversal batch for a payment that entered the database as suspended, you must identify the original suspended payment rather than the active recycled payment. When the reversal batch is submitted to BRM, all active recycled payments that are descended from the original payment will be internally reversed.

- When directly reversing a payment, you can reverse only original payments. If you reverse an original suspended payment, PCM_OP_BILL.Reverse reverses all of the active recycled payments that were distributed to customer accounts from that payment. See "About Directly Reversing Payments from BRM".

BRM uses PCM_OP_BILL.Reverse to process manual reversals and PCM_OP_PYMT_RECYCLE_PAYMENT to process payment reversals during recycling. For more information, see "How Payments Are Reversed".

For more information on direct reversals, see "How BRM Reverses Payments".

**About Refunding Payments**

You create payment refunds by using Customer Center or calling the PCM_OP_BILL_ITEM_REFUND opcode. You can only refund a payment that has been posted in a customer account; you cannot refund a suspended payment.

Before it refunds a payment, PCM_OP_BILL_ITEM_REFUND determines whether Payment Suspense Manager is enabled. If so, it checks the account POID in the input flist against the account POID in the /config/psm object to see if the account is the suspense account. If the two POIDs match, the opcode generates an error.

If you suspend a payment that has been refunded (the /item/refund object was closed), the due amount on the account is increased by the same amount that was removed by the refund adjustment.

For more information on payment refunds, see "About Refunds" in BRM Managing Accounts Receivable.

**About Removing Unallocatable Payments from Suspense**

In some cases, you may determine that a suspended payment cannot be allocated, and should be removed from the system. Payments of this nature represent unrealized revenue. To track revenue and report for these payments, you can remove them from the payment suspense account as unallocatable.
You use Payment Center to remove unallocatable payments from suspense. BRM assigns a G/L ID of 112 for the reversal, placing the payment amount in a special G/L bucket so that you can obtain information about how much unallocatable revenue you have. This amount was a credit that your company could not recognize toward a debit on any account. It is removed from the system and tracked for accounting purposes.

You can remove an original or recycled payment from suspense as unallocatable. Removing unallocatable payments from suspense does not generate any recycled payments.

In some cases, you may need to partially distribute a suspended payment and remove the remaining suspended amount as unallocatable. If you then resuspend one of the distributed payments, BRM creates a new suspended payment for the distributed payment’s amount, and you can later remove this new amount as unallocatable if necessary.

---

**Note:** If one or more distributed payments have been removed as unallocatable, you cannot directly reverse the original payment from the BRM database.

---

### About Payment Suspense Manager and Client Applications

The payment suspense process is initiated in one of three ways:

- Through original payments, suspended payments, and Payment Tool when payment analysts work with a payment batch.
- Through a payment gateway when it processes a payment file.

**Note:** For the full range of Payment Suspense Manager functionality to work with a payment gateway, the payment gateway must be directly integrated with BRM payment services.

- Through Payment Center when payment analysts work with payment batches that contain suspended payments or after payments have been posted in customer accounts.

There are two BRM client applications that are used in the payment suspense process: Payment Tool and Payment Center. Payment Tool is used to determine whether any payments should be suspended and Payment Center is used to investigate and correct suspended payments. Depending on how you initiate the payment suspense process, you use one or both of these applications.

- If a payment clerk loads payment batches into BRM, you use a combination of Payment Tool and Payment Center.
- If the payment gateway loads payment batches into BRM, you use only Payment Center.
If payments are already posted to customer accounts, you use only Payment Center.

How you use these client applications differs depending on how the payment process is initiated.

**Figure 8–6** shows how the payment suspense process works when you use Payment Tool to load payments:

**Figure 8–6  Payment Suspense Process Using Payment Tool**

![Diagram](image)

When you receive externally initiated payment batches, you perform all validation and suspense tasks by using Payment Tool and all correction tasks by using Payment Center.

Use Payment Tool for the following tasks:

- Validate a batch of payments.
- Manually suspend a payment that passed validation but requires special handling. Or, change the status of a manually suspended payment back to validated.
- Submit validated payments to BRM.

Use Payment Center for the following tasks:

- Investigate and correct suspended payments.
- Apply corrected payments to the appropriate account.
- Remove a payment from suspense if you cannot correct it within a reasonable time.

Typically, when you use Payment Tool to process a batch of payments, you import the batch and validate the payments. The results of validation show the status of payment, indicating whether the payment was successful or suspended.

You can then do one of three things:

- Submit the batch to BRM, which posts all successful payments to the correct accounts and posts any suspended payments to the payment suspense account. In this case, you would later open Payment Center to investigate and correct the suspended payments as well as resubmit the corrected payments to BRM.
Correct the suspended payments before submitting the batch. In this case, you would immediately launch Payment Center from Payment Tool and correct the payments. Then, you must return to Payment Tool to revalidate the payments and submit the payment batch to BRM.

Save the payment batch as a PMT file for later processing. In this case, you would open the PMT file in Payment Tool and begin the validation process again.

When you use automated payment processing, like that provided by a payment gateway, there is no need for payment personnel to handle a payment batch, validate payments, or submit payments to BRM. These steps are all performed automatically by the payment gateway working in concert with BRM.

Figure 8–7 shows how the payment suspense process works if you use a payment gateway to process payments.

---

**Summary of Payment Suspension Guidelines and Restrictions**

This section provides a review of guidelines and restrictions that apply to Payment Suspense Manager.

- **General Guidelines**
- **Suspended Payment Guidelines**
- **Distributed Payment Guidelines**

**General Guidelines**
The following guidelines and restrictions apply to suspended payment processing.

- Only externally initiated payments can be suspended and managed by using Payment Center.
- The currency of a recycled payment must match the currency of its original payment.
■ Payments can be recycled any number of times, but a recycled payment can have only one original payment.

■ You cannot change the properties of a payment after it has been directly reversed, removed as unallocatable, or recycled completely.

■ You cannot directly reverse a suspended payment if any portion of the payment has been removed from suspense as unallocatable.

■ You cannot distribute failed payments. These payments are stored in BRM as /event/billing/payment/failed objects and have a status of PIN_PYMT_FAILED. They are created to handle unconfirmed payment processing. For more information, see “Handling Failed Unconfirmed Payments”.

■ To directly reverse a payment outside of the recycling process, you must reverse the original payment. If you reverse a suspended payment that was applied to one or more customer accounts, all posted payments will be reversed before the suspended payment is reversed.

**Suspended Payment Guidelines**
The following guidelines apply to suspended payments.

■ You can process only one suspended payment at a time; you cannot apply multiple suspended payments to customer accounts in the same allocation. Similarly, you cannot return two distributed payments that originated from different suspended payments in the same operation.

■ If you change the properties (for example, the action owner) of a suspended payment, it will be reversed and a new payment event is created to contain the updated information.

■ You cannot change the action owner or any other properties of a suspended payment after it has been completely distributed to customer accounts. However, if you return any of the distributed payments to suspense, you can change the properties of the resulting suspended payment.

■ You cannot refund a suspended payment; you can refund only a payment that has been applied to a customer account. You create payment refunds by using Customer Center or by calling PCM_OP_BILL_REFUND_ITEM.

■ If you suspend a payment that was previously refunded (the /item/refund object was closed), the due amount on the account is increased by the same amount that was removed by the refund adjustment. For more information on adjustments, see “About Adjustments” in *BRM Managing Accounts Receivable*.

**Distributed Payment Guidelines**
The following guidelines apply to distributed payment processing.

■ If the entire list of distributed payments does not pass validation, it is rolled back to suspense.

■ You cannot recycle a payment directly from one customer account to another customer account; first you must suspend the payment and then apply it to the target account.

■ When recycling a distributed payment to suspense, the entire payment is recycled; you cannot recycle a partial payment amount.

■ If one or more distributed payments have been removed as unallocatable, you cannot reverse the original payment from the BRM database.
Configuring BRM for Payment Suspense Manager

To set up BRM for Payment Suspense Manager, you complete three tasks:

- Enabling Payment Suspense in BRM
- Creating the Payment Suspense Account
- Working with Suspense Reason Codes and Action Owner Codes

Enabling Payment Suspense in BRM

By default, Payment Suspense Manager is disabled in BRM. You can enable this feature by modifying a field in the ar instance /config/business_params object created during BRM installation.

- With the feature enabled, BRM categorizes a payment as successful, suspended, or failed whenever it validates a payment.
- With this feature disabled, BRM determines only whether a payment is successful or failed; it does not determine whether a payment should be suspended. Any payment that would normally be suspended and moved to the payment suspense account for investigation is, instead, considered a failed payment.

You modify the /config/business_params object using the pin_bus_params utility. For information on this utility, see pin_bus_params.

To enable Payment Suspense Manager:

1. Use the following command to create an editable XML file for the BusParamsAR parameter class:
   
   ```
   pin_bus_params -r BusParamsAR bus_params_AR.xml
   ```

   This command creates the XML file named bus_params_AR.xml.out in your working directory. If you do not want this file in your working directory, specify the full path as part of the file name.

2. Search the XML file for following line:
   
   ```
   <PaymentSuspense>disabled</PaymentSuspense>
   ```

3. Change disabled to enabled.

   **Caution:** BRM uses the XML in this file to overwrite the existing /config/business_params object for the ar instance. If you delete or modify any other parameters in the file, these changes affect the associated aspects of BRM’s billing configuration.

4. Use the following command to load the change into the /config/business_params object:
   
   ```
   pin_bus_params bus_params_AR.xml
   ```

   You should execute this command from the BRM_Homesys/data/config directory, which includes support files used by the utility. To execute it from a different directory, see pin_bus_params.

5. Read the object with the testnap utility or Object Browser to verify that all fields are correct.
See "Using Testnap" in *BRM Developer’s Guide* for general instructions on using testnap. See "Reading Objects by Using Object Browser" in *BRM Developer’s Guide* for information on how to use Object Browser.

6. Stop and restart the Connection Manager (CM). For more information, see "Starting and Stopping the BRM System" in *BRM System Administrator’s Guide*.

7. For multiple databases, run the `pin_multidb` script with the `-R CONFIG` parameter. For more information on this script, see `pin_multidb`.

Creating the Payment Suspense Account

When BRM determines that a payment should be suspended, it stores the suspended payment and all information available for the payment in the payment suspense account.

By default, you can have only one payment suspense account in BRM. You create the payment suspense account by using Customer Center and base it on the default CSR plan.

To create the payment suspense account:

1. Start Customer Center and choose *File - New - Account Type* (Business or Consumer) to activate the Account Creation wizard.

2. On the Contact page, enter payment for First Name and suspense for Last Name. This information is not case sensitive.

3. On the Plan page, select the CSR plan for your BRM system.

   **Important:** The CSR plan you select must comply with these rules:
   - The admin_client service should have been used when setting up the plan.
   - There can be absolutely no deals or charges attached to the plan.

4. On the Payment page, select Undefined for Payment Method.

5. For all other required fields in the Account Creation wizard, select the defaults.

6. Click Finish to create the account.

When BRM creates the payment suspense account, the PCM_OP_CUST_POL_PREP_NAMEINFO policy opcode first prepares the contact information for validation and then determines whether Payment Suspense Manager is enabled. If it is enabled and the account being checked is the payment suspense account, this opcode retrieves the POID of the `/config/psm` object, provided that object exists. BRM references the `/config/psm` object to locate the payment suspense account whenever it suspends a payment or you correct a payment.

The payment suspense account POID is then passed to the PCM_OP_CUST_SET_NAMEINFO opcode. This opcode stores the account POID in the `/config/psm` object.

Working with Suspense Reason Codes and Action Owner Codes

Suspense reason codes explain why a payment was moved into or out of suspense or why an unallocatable payment is removed from the system. Action owner codes indicate who is responsible for correcting the problem or taking other action on the payment. For example, a reason code could indicate that an account number is wrong or that a payment needs to be removed from suspense because it could not be
allocated within a reasonable amount of time. An action owner code could indicate that Sally Brown is responsible for investigating the suspended payment.

---

**Note:** Action owner codes are typically used to assign payment analysts to particular suspended payments and allow them to search for all the payments they need to correct. However, you can customize action owner codes to provide different types of information such as an action that you took in regard to a suspended payment.

---

Reason codes and action owner codes are used in various ways by the different tools you use to process payments:

- **Payment Tool:** Provides reason lists that payment personnel can choose from as they suspend a payment.

- **Payment Center:** Provides action owner lists that payment personnel can choose from when assigning a person to correct a payment or use as a criteria when searching for a suspended payment. It also provides reason lists that payment personnel choose from when correcting a payment, suspending a payment, or removing an unallocatable payment from the system.

- **Payment Gateway:** Automatically assigns reasons to payments processed through a payment gateway provided you implement a preprocessing application to map reason codes in the payment file to reason codes you have created in BRM.

To ensure that BRM can assign the full range of reason codes and action owner codes suitable for your business needs, you customize BRM by:

- Creating and loading a `reasons.locale` file that lists each reason code and action owner code.

- Associating each reason code and action owner code with the appropriate Payment Suspense Manager reason code domain.

**About the Reasons.locale File**

The `reasons.locale` file defines each reason code domain, the reason codes or action owner codes that belong to the domain, and the event G/L ID. A reason code domain is a unique identifier, or version, used to organize reason codes according to the activities they are used for. For example, all reason codes that describe why you are removing an unallocatable payment from suspense would be defined within the reason code domain dedicated to that purpose. The domain and reason code information is used to build the `/strings` object and the event G/L ID is used to build the `/config/map_glid` object.

Payment suspense reason codes and action owner codes use the following domains:

- **Payment suspense reason codes:** “Reason codes-Payment Suspense Management” version 14.

- **Action owner codes:** “Reason codes-Payment Suspense Management Action Owner reason” version 15.

- **Reason codes for reversals due to recycling and removing unallocatable payments from suspense:** “Reason codes-Payment Suspense Management, Reversal Reason” version 16.

The following ranges are reserved for default BRM reason codes related to payment suspense and payment status:
0: Default reason code.

1 to 1000: Reason codes for successful payments.

1001 to 2000: Reason codes for failed payments.

2001 to 3000: Reason codes for suspended payments.

3001 to 4000: Action owner codes.

4001 to 5000: Reason codes for reversals generated when a payment is moved from a source account to a target account during recycling and for removing unallocatable payments from suspense.

If you want to add reason codes of your own, use values above 100,000.

You must assign G/L IDs for all reason codes you create for the following payment processes:

■ Removing unallocatable payments from suspense.

This enables BRM to map these payments to the G/L bucket used to store a record of payments that were removed from suspense because they were not correctable. You can then create reports and applications to help you track this form of unrealized revenue. The G/L ID assigned to the /event/billing/reversal event, which occurs when payments are removed from BRM as unallocatable, is 112.

■ Recycling payments to or from suspense. You can use information in this bucket to help determine how much revenue is recovered from suspense. This G/L bucket is reserved for distributed payments, distributed payments returned to suspense, and original payments to a customer account that are manually suspended, is 113. G/L ID bucket 113 stores both the recycled payment and its corresponding payment reversal. Storing both the payment and reversal in the same G/L ID bucket ensures the correct balance of debits to credits when generating reports.

Note: You should not assign G/L IDs for action owner codes, and there is no need to assign G/L IDs for the reason codes for suspended payments.

The following example shows a reasons.locale file segment defining a payment suspense reason code domain. Note that some of the reason codes are BRM defaults and some are defined by a user (ID >= 100,000).

```
LOCALE = en_US
DOMAIN = "Reason Codes - Payment Suspense Management"
STR
  ID = 2001;
  VERSION = 14;
  STRING = "Account No not found."
  HELPSTR = "Account Number not found in database"
END

DOMAIN = "Reason Codes - Payment Suspense Action Owner reason"
STR
  ID = 102,001;
  VERSION = 15;
```
STRING = "Alaya Baker";
HELPSTR = "Payments Processing department"
STR
ID = 102,002;
VERSION = 15;
STRING = "Micheal Orden";
HELPSTR = "Payments Processing department"
END

DOMAIN = "Reason Codes - Payment Suspense Management Reversal Reason";
STR
ID = 4999;
VERSION = 16;
STRING = "Unable to correct payment";
HELPSTR = "Unable to correct payment."
EVENT-GLID
"/event/billing/reversal" 112;
EVENT-GLID END
STR
ID = 110,000;
VERSION = 16;
STRING = "Payment recycled to suspense";
HELPSTR = "Payment moved from wrong customer account to payment suspense account"
EVENT-GLID
"/event/billing/reversal" 113;
EVENT-GLID END
STR
ID = 110,001;
VERSION = 16;
STRING = "Distributed Payment allocation";
HELPSTR = "Suspended payment applied to multiple accounts"
EVENT-GLID
"/event/billing/reversal" 113;
EVENT-GLID END

For more information on the reasons.locale file and assigning G/L IDs, see "Assigning G/L IDs to Nonrated Events" in BRM Collecting General Ledger Data.

Loading Reason Codes into the BRM Database

To define reason codes and action owner codes for Payment Suspense Manager, you edit the reasons.en_US sample file in the BRM_Home/sys/msgs/reasoncodes directory. You then use the load_localized_strings utility to load the contents of the file into the /strings and /config/map_glid objects.

When you run the load_localized_strings utility, use this command:

load_localized_strings reasons.locale

Note: If you are loading a localized version of this file, use the correct file extension for your locale. For a list of file extensions, see Locale names.
Caution: The load_localized_strings utility overwrites the /config/map_glid object. If you are updating this object, you cannot load new G/L ID maps only. You must load complete sets of data each time you run the load_localized_strings utility. This is also true when using the /strings object, but only if you specify the -f parameter. Otherwise, the load_localized_strings utility appends the new data to the object.

For information on loading the reasons.locale file, see "Loading Localized or Customized Strings" in BRM Developer’s Guide. For information on creating new strings for this file, see "Creating New Strings and Customizing Existing Strings" in BRM Developer’s Guide.

Setting Up Permissions for Payment Center

You use Permissioning Center to give payment analysts access to Payment Center and to restrict them from performing certain activities.

1. Log in to Permissioning Center as the root user.
2. From the application list, select Payment Center.
3. Create one or more roles.
4. Set permissions for each role.
5. Assign each payment expert to one or more roles.
6. From the application list, select Customer Center.
7. Ensure that all payment analysts have Read Only or Read/Write access to Customer Center.

Note: If you skip this step, some Payment Center functions may not work correctly.

For detailed instructions on how to set permissions, see Permissioning Center Help.

About Customizing Payment Suspense Manager

You can customize Payment Suspense Manager in several ways, such as:

- Applying suspended payments to one or more customer accounts
- Returning one or more distributed payments to the suspense account
- Suspending a payment originally posted to a customer account
- Removing a payment from the suspense account as unallocatable

For an overview of how to perform these and other customizations, read these sections:

- How Payments Are Suspended During Payment Processing
- How Payments Are Recycled to and from Suspense
- How Recycled Payments Are Retrieved
- How Payments Are Reversed
How Payments Are Suspended During Payment Processing

During payment processing, the PCM_OP_PYMT_COLLECT opcode calls the PCM_OP_PYMT_VALIDATE_PAYMENT opcode to determine the status of the payment records. (For information about PCM_OP_PYMT_COLLECT, see "How BRM Collects Payments").

When PCM_OP_PYMT_VALIDATE_PAYMENT receives a payment to validate, it determines whether the payment should be suspended and prepares it for posting by enriching the flist with any missing information.

It begins by evaluating the input flist to determine whether it contains the channel ID for the payment. If not, it gets the channel ID from the /config/ach object and adds it to the flist.

Then, PCM_OP_PYMT_VALIDATE_PAYMENT calls the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode to validate the payment and further enrich the flist with any missing payment information.

After PCM_OP_PYMT_VALIDATE_PAYMENT receives the results from the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode, it returns the flist to PCM_OP_PYMT_COLLECT for use when the payment is posted.

Flags are not used directly by PCM_OP_PYMT_VALIDATE_PAYMENT. They are passed in from PCM_OP_PYMT_COLLECT for the PCM_OP_PYMT_SELECT_ITEMS opcode. For example, Payment Tool can set the PCM_BILLFLG_DEFER_ALLOCATION flag to indicate which payments should be left unallocated.

---

**Note:** When you submit payments by using Payment Tool, payments that have a valid account number but a missing bill number are not marked for suspense by default. To enable Payment Tool to return such payments as suspended, see "Customizing Suspense Criteria for Payment Tool".

---

How Payments Are Recycled to and from Suspense

The PCM_OP_PYMT_RECYCLE_PAYMENT opcode removes a payment from a source account and posts it to a target account. It is called when one or more payments in Payment Center are submitted to the BRM database.

---

**Caution:** It is not recommended to have more than one payment suspense account in the /config/psm object.

---

When a payment is recycled from suspense to a customer account, the input flist generated for the submission includes the corrected account number, corrected bill number, and the transaction ID of the original payment. It can also contain other corrected information if you customized BRM to include additional validation criteria.
When a transaction is opened, PCM_OP_PYMT_RECYCLE_PAYMENT handles all of the activities required to move each payment in the CHARGES array from the source account to the target account. The transaction ID is recorded in all the event objects created when this opcode processes a payment.

In general, to recycle the payment, PCM_OP_PYMT_RECYCLE_PAYMENT uses the source account referenced in the input flist’s PIN_FLD_EVENT_OBJ field and the destination account POID in the PIN_FLD_ACCOUNT_OBJ field to determine the operation it will perform.

For example, if the account referenced in PIN_FLD_EVENT_OBJ is the suspense account and the account referenced in PIN_FLD_ACCOUNT_OBJ is a customer account, the opcode distributes all or part of the suspended payment to the customer account.

The source and target accounts, combined with the number of payments in the CHARGES array, determine the operation that PCM_OP_PYMT_RECYCLE_PAYMENT performs, as shown in the following table.

**Recycled Payment Results**

PCM_OP_PYMT_RECYCLE_PAYMENT performs the following operations when recycling payments:

1. Opens a transaction.

2. Determines the direction of the payment recycling: to or from the payment suspense account. The transfer direction is determined by the active payment’s account POID and the target account POID.

   - Determines if multiple distributed payments are being handled.
   - Checks that the currency is the same for every charges element.
   - If it is returning an entire distribution to suspense, calls PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH to retrieve the active suspended payment information, including the amount remaining in the suspended payment after recycling, and adds it to the CHARGES array.

3. Chooses a course of action based on the source account/target account combination plus the information in the CHARGES array.

4. Calls PCM_OP_PYMT_COLLECT to apply the payments to the target accounts, prepares the payment batch, and enriches the input flist of PCM_OP_PYMT_COLLECT with the following information:

   - The total charge amount.
   - Total reversal amount, for posted payments that are being suspended.
   - Total original payment amount, for payments that are being posted from the payment suspense account to a customer account.

   This opcode, in turn, calls PCM_OP_PYMT_VALIDATE_PAYMENT to validate the information in the CHARGES array. If an invalid scenario exists, the payment is guided to suspense.

   **Note:** For payments guided to suspense due to validation failures, PCM_OP_PYMT_RECYCLE_PAYMENT rolls back the entire transaction after PCM_OP_PYMT_COLLECT completes.

5. Prepares the reversal batch, creating new elements and setting a new reason code.
If PCM_OP_PYMT_RECYCLE_PAYMENT is being called for a distribution and the total of the charges in the input flist does not equal the amount to be distributed, the opcode determines whether the input flist specifies an amount to be placed in suspense.

- If the input flist does not specify an amount to be placed in suspense, the opcode posts the remaining amount to the suspense account.
- If the input flist specifies an amount to be placed in suspense, the opcode generates an error.

6. Calls PCM_OP_PYMT_COLLECT to:
   - Create the payment batch.
   - Create a new suspended payment if an amount remained in the suspended payment after it was applied to one or more accounts.
   - Update the /event/billing/payment/pay_type object with the SUB_TRANS_ID value specified in the PIN_FLD_CHARGES array of the flist.

7. If the payment batch was successful, updates the input flist for the PCM_OP_BILL_REVERSE opcode with the following information:
   - The reason code for the reversal
   - The PAYMENT_TRANS_ID value

8. Calls PCM_OP_BILL_REVERSE to create the reversal batch.

   PCM_OP_BILL_REVERSE updates the /event/billing/reversal/pay_type object with the PAY_TRANS_ID, the POID of the payment suspense account, and the reversal total.

   If PCM_OP_PYMT_COLLECT fails or if any information is invalid, PCM_OP_PYMT_RECYCLE_PAYMENT rolls back the entire transaction, leaving the suspended payment in its original state.

9. Records the success or failure of the entire operation in the PIN_FLD_RESULTS field of the output flist.

### How Recycled Payments Are Retrieved

The PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH opcode retrieves payment information such as the payment amount, transaction ID, subtransaction ID, account and bill number, payment types, and payment status. If you specify that more information be returned, this opcode can return additional payment information such as the action owner and the date suspended.

This opcode is called by:

- Payment Center to provide additional information on the payments returned by a search: allocated amount and unallocated amount, for example.
- PCM_OP_BILL_REVERSE when it reverses a payment that was posted to the suspense account when it entered the BRM database. This opcode uses PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH to find all active payments associated with the original suspended payment so that those payments also can be reversed. For more information, see "How Payments Are Reversed".
- PCM_OP_PYMT_RECYCLE_PAYMENT when it returns an entire set of distributed payments to the suspense account.
As input, this opcode receives the transaction ID of the payment it is to search for. It also receives an optional PIN_FLD_RESULTS array, which, in conjunction with the PCM_OP_FLAG_READ_RESULT flag, determines the type of information returned by the opcode.

PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH performs the following operations when searching for payments:

1. Verifies that the transaction ID in the PIN_FLD_TRANS_ID field is for an original suspended payment. To do this, the opcode checks the PIN_FLD_SUB_TRANS_ID field. If this field is not NULL, the payment is not an original payment, and the opcode returns an error.

2. Searches for all payments whose subtransaction ID matches the PIN_FLD_TRANS_ID field of the payment in the input flist.

3. Checks to see if the PIN_FLD_RESULTS array in the input flist and the PCM_OP_FLAG_READ_RESULT flag are present. The opcode returns payment information based on these presence of the flag and array, as follows:
   - If neither the flag nor the array is present, the opcode returns only the fields in the output flist.
   - If the flag is present, the opcode returns all the fields in the /event/billing/payment object.
   - If the flag is not present but the array is, the opcode returns the fields in specified in the PIN_FLD_RESULTS array plus the payment information fields normally included in the output flist.
   - If the both the flag and array are present, the opcode ignores the array and returns all the fields in the /event/billing/payment object.

PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH does not retrieve payment states for payments that have been reversed and are no longer active. If the search results are NULL, PCM_OP_PYMT_RECYCLED_PAYMENTS_SEARCH searches for reversal events related to a suspended payment. If a reversal event is found, the suspended record is filtered and not returned.

How Payments Are Reversed

The PCM_OP_BILL_REVERSE opcode prepares information for PCM_OP_BILL_REVERSE_PAYMENT to perform reversals. PCM_OP_BILL_REVERSE reverses payments during the following payment suspense operations:

- A payment is recycled to or from suspense.
- A suspended payment is removed from BRM as unallocatable.

Payments can also be reversed directly from BRM by submitting a payment reversal batch from Payment Tool. For information on how direct reversals are performed, see "How BRM Reverses Payments".

How Payments Are Reversed During Recycling

If PCM_OP_BILL_REVERSE is called from PCM_OP_PYMT_RECYCLE_PAYMENT, the reversal is due to recycling, and PCM_OP_BILL_REVERSE performs the following operations:

1. Assigns the reversal TRANS_ID value for each recycled payment.

2. Opens a transaction and checks the appropriate /config/business_params object to verify that Payment Suspense Manager is enabled.
3. Calls PCM_OP_BILL_REVERSE_PAYMENT to perform the reversal and create the associated objects in the database. For each /event/billing/reversal object created, PIN_FLD_PAYMENT_TRANS_ID is set to the transaction ID of the recycled payment. This opcode populates the reversal flist with each recycled payment’s TRANS_ID value.

4. Populates the payment batch with the sum of the reversal flist.

5. If the reversal was performed as part of recycling multiple distributed payments back into suspense, returns the recycled payments’ reversal information in the PIN_FLD_MULTI_RESULTS array.

How Payments Are Removed As Unallocatable

PCM_OP_BILL.Reverse performs the following operations to remove payments from BRM as unallocatable:

1. Assigns the reversal event a TRANS_ID value.

2. Opens a transaction and checks the appropriate /config/business_params object to verify that Payment Suspense Manager is enabled.

3. Checks to see if the following criteria are met:
   - The payment is a suspended payment. To do so, it compares the account POID in the flist with the account POID of the payment suspense account in the /config/psm object.
   - The payment is active.
   - The value of the PIN_FLD_FLAGS field is set to PIN_REVERSE_FLAG_REVERSE_AS_UNALLOCATED (1).

4. Calls PCM_OP_BILL_REVERSE_PAYMENT to reverse the payment in the payment suspense account and create the associated objects in the database. For the /event/billing/reversal object, PIN_FLD_PAYMENT_TRANS_ID is set to the transaction ID of the suspended payment. This opcode populates the reversal flist with the payment’s TRANS_ID value.

5. Populates the payment batch with the sum of the reversal flist.

6. Validates the reverse payment operation and creates the reversal batch event in the database.

The result of the reversal is returned in the PIN_FLD_RESULTS field of PCM_OP_BILL.Reverse. The reversal will not be allowed if either of the following conditions is true:

- The payment is not a suspended payment.
- The payment is not an active payment.

Customizing Payment Suspense Validation

The PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode validates payments to determine whether they can be successfully posted or whether a failed, unconfirmed payment needs reversal. If automatic write-off reversals are enabled, it also determines whether BRM should perform a write-off reversal.

In default implementations, BRM considers two questions when determining whether a payment should be suspended:

- Is the account number present and valid?
■ Is the account closed?

By default, the bill number is not initially used to suspend payments. It is used only when the account number is missing.

A payment is suspended in the following situations:
■ The account is closed.
■ The account number is missing and the bill number is missing.
■ The account number and the bill number are invalid.
■ The account number does not match the account number from the bill.

A payment is posted if the account is not closed and the following is true:
■ The account number is present and valid. If the bill number is missing, the payment is posted at the account level.
■ The account number is missing but the bill number is present and valid. Such payments are posted at the bill level.

To have Payment Tool mark these payments as suspended, see "Customizing Suspense Criteria for Payment Tool".

You can broaden this scope by customizing the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode to perform validation using additional criteria: for example, customer segment, payment amount, and so forth. BRM uses these criteria to determine whether the payment validation logic should mark a payment for suspense.

Note: If you include additional criteria in the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode that is not already part of the objects created by payment processing, you must also extend these objects by adding these fields so that the object records all the criteria considered by payment validation.

If neither the bill number nor bill POID was submitted with the payment, you can configure BRM to find the bill based on the due amount. See "Finding Bills by Due Amount".

Customization Example: Suspending Large Payments

As an example, if you want a payment analyst to always examine suspiciously large cash payments or cash payments that appear to arrive from the Internet, you can customize BRM to suspend any payments that meet these conditions. If a cash payment is suspiciously large, there may be a customer error or recording error that needs to be investigated. Because the Internet is not a likely source of cash payments, you may want to obtain extra confirmation on how this payment was made.

This type of customization includes the following tasks:
■ Modify the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode to include logic that checks the PIN_FLD_PAYMENT_TYPE field to determine if this is a cash payment and also checks PIN_FLD_CHANNEL_ID to see whether the payment was made over the Internet. If both conditions are met, have the opcode set PIN_FLD_STATUS to PIN_PYMT_SUSPENSE.

■ Also modify the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode to include logic that checks the PIN_FLD_AMOUNT field to see if the payment amount is greater than the amount you establish as the threshold for suspiciously large payments.
large payments: for example, $10,000. If this is a cash payment and the payment amount exceeds the threshold, have the opcode set PIN_FLD_STATUS to PIN_PYMT_SUSPENSE.

Customization Example: Threshold for Suspending Payments
You can customize the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode to enforce business practices related to suspense. For example, if you find that having payment analysts review suspended payments for very small amounts costs you more than the payments are worth, you can customize BRM so that it does not move any payment of less than a certain amount into the payment suspense account.

To create a suspense threshold, you customize the opcode to check the PIN_FLD_AMOUNT field to see if the payment amount is less than the threshold amount: for example, $1. You check this condition for any payment whose status is set to PIN_PYMT_SUSPENSE. For each payment that meets these conditions, you set the account POID to a special account you set up for this type of unallocatable payment. You also set PIN_FLD_STATUS to PIN_PYMT_SUCCESSFUL so that PCM_OP_PYMT_COLLECT can post the payment.

**Note:** When you implement any of the customizations just discussed, you modify the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode. This opcode validates only externally initiated payments; it does not validate BRM-initiated payments. Therefore, none of the customizations you implement through this opcode affect BRM-initiated payments.

Customization Example: Finding Unconfirmed Payments
In the case of failed unconfirmed payment, if the payment processor is not able to send a transaction ID with each payment, you can modify the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode to find the original unconfirmed payment by using other payment properties that are sent from the payment gateway, such as the original unconfirmed payment amount. When the original unconfirmed payment is found, the item object to which the payment was applied can be loaded into the input list for the PCM_OP_PYMT_APPLY_FEES opcode.

**Note:** If you used failed payment properties to locate the payment, you must also update the /event/billing/payment/failed object to contain the same properties that you are using to locate the original unconfirmed payment.

Customization Example: Error Handling
You can modify the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode to handle errors that occur because the original unconfirmed payment cannot be found or the transaction IDs do not match.

Default Payment Validation Process
The PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode processes payments in three phases:

1. **Payment suspense phase:** Receives a list of payments from PCM_OP_PYMT_VALIDATE_PAYMENT and checks the /config/business_params object to determine whether Payment Suspense Manager is enabled. If so, it checks the
payments to determine whether any payments need to be suspended and updates
the PIN_FLD_STATUS field accordingly.

2. **Failed unconfirmed payment phase**: For all payments with a PIN_FLD_STATUS
in the failed range, uses the payment method value to determine whether a failed
payment has an associated unconfirmed successful payment or if it is a failed
confirmed payment.

   The input flist contains the POID of the original payment item, the transaction ID,
   and the amount of the original payment in order to properly handle unconfirmed
   failed payments.

3. **Write-off reversal phase**: Checks the /config/business_params object to determine
whether automatic write-off reversals are enabled. If so, it determines whether the
payment is for an account, bill, or bill item that has been written off. If so, it sets
PIN_FLD_STATUS accordingly.

See "About Payment Status" for information on PIN_FLD_STATUS values and ranges.

**Payment Suspense Phase**

In this phase, the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode first
checks the PIN_FLD_STATUS field to determine whether the payment has a status in
the suspense range, indicating that the payment has already been marked for
suspense. In this case, the opcode passes the output flist and associated status back to
PCM_OP_PYMT_VALIDATE_PAYMENT. BRM will then use PCM_OP_PYMT_COLLECT
to direct the payment to the payment suspense account.

If the payment is not already marked for suspense, the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode does the following:

1. Validates the account number specified by the PIN_FLD_ACCOUNT_NO or PIN_FLD_ACCOUNT_OBJ field in the input flist or searches for the corresponding account POID.

2. Validates the bill number specified by the PIN_FLD_BILL_NO field in the input flist and searches for the corresponding bill POID, /billinfo POID, and account POID.

3. If neither a bill POID nor a bill number was submitted with the payment, BRM uses the bill amount to find the bill.

   If the account and bill numbers supplied in the input flist are both invalid or the
   PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode cannot find the
   account or bill POIDs (or bill amount), it marks the payment for suspense. It also
   compares the account POID from the /bill object with the account POID found in
   step 1. If they do not match, the opcode marks the payment for suspense.

4. Checks the PIN_FLD_STATUS field in the /account object to determine whether
the account is closed. If the account is closed, the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode marks the payment for suspense.

5. Checks any custom validation criteria and marks the payment for suspense if
appropriate.

6. Writes the status of the payment in the PIN_FLD_STATUS field, and includes this
field in the output flist. If the payment needs to be suspended, it sets this field to
one of the values in the suspense range, as appropriate.
Note:
- When an /event/billing object is created for a suspended payment, it stores the original reason code associated with a failed payment that has been flagged for suspense. This ensures that the reason initially associated with the failed payment is not lost if BRM places the payment in suspense.
- When the /event/billing/payment object is created, it stores the original account number provided for the payment being suspended, the original bill number, and the original transaction ID.

Failed Unconfirmed Payments Phase
In this phase, the opcode considers only payments whose status is marked as failed: those whose PIN_FLD_STATUS value is in the financially failed range. These payments are ones that the opcode was able to validate, but were marked by the payment processor as failing for financial reasons. In default implementations, the opcode requires the transaction ID and result of each payment to prepare the failed unconfirmed payments for reversal.

If the failed payment is an unconfirmed payment, PCM_OP_PYMT_VALIDATE_PAYMENT:

1. Searches the /event/billing/payment object for an unconfirmed payment with the transaction ID passed in with the failed payment.

2. Does one of the following:
   - If an unconfirmed payment is found, sets PIN_FLD_RESULT to PIN_PAY_TYPE_SUCCESS.
   - If the transaction ID of the unconfirmed payment is not found, it checks the /config/business_params object to determine whether Payment Suspense Manager is enabled.
     - If so, it sets PIN_FLD_STATUS to PIN_FLD_FAILED_SUSPENSE and BRM posts the payment to the payment suspense account.
     - If not, it sets PIN_FLD_RESULT to PIN_FLD_PAYMENT_RESULT_FAIL, and a reversal does not occur. The subsequent steps do not occur and manual allocation is required.

3. Loads the following unconfirmed payment information from the /event/billing/payment storable class into the PIN_FLD_FAILED_PAYMENT_FEE substruct in the PIN_FLD_EXTENDED_INFO substruct of the output flist:
   - Payment channel ID
   - Payment method
   - Transaction ID
   - Original payment amount
   - Customer segment

Note: For unconfirmed payments, the customer segment value is also retrieved from PIN_FLIST_CUSTOMER_SEGMENT_LIST in the input flist of this policy.
4. Passes the POID of the successful unconfirmed payment in the output flist to PCM_OP_BILL_REVERSE_PAYMENT so it can be reversed.

The output flist sends the array of reversal events and tax events (if created) that were passed in by the PCM_OP_AR_WRITEOFF_REVERSAL opcode.

**Write-off Reversal Phase**

In this phase, PCM_OP_PYMT_VALIDATE_PAYMENT considers only payments whose status is marked as successful: those whose PIN_FLD_STATUS value is in the successful range. The opcode determines which of these payments is for an account, bill, or bill item that has been written off. It performs the following operations:

- Checks the /config/business_params object to determine if automatic write-off reversal functionality for payment processing is enabled.
- If this is enabled, checks the /profile/writeoff object to verify that the write-off flag is set for the account.
- If both checks are successful, sets the PIN_FLD_STATUS field in the output flist to PIN_PYMT_WRITEOFF_SUCCESS.

**Payment Validation Flags**

Flags are not used directly by the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode. They are passed in from PCM_OP_PYMT_COLLECT for PCM_OP_PYMT_SELECT_ITEMS. For example, Payment Tool can set the PCM_BILLFLG_DEFER_ALLOCATION flag to indicate which payments should be left unallocated.

**Customizing Payment Guidance to Suspense**

The PCM_OP_PYMT_POL_SUSPEND_PAYMENT policy opcode enables you to define custom rules for handling payments that are being guided into suspense. For example, you can modify this policy opcode to track the number of times the same payment is guided back to suspense after being applied to the wrong customer account.

In addition, if your company processes distributed payments, you can implement rules to distribute a payment directly to the proper accounts, without first saving the payment to the suspense account. For example, by using the account number associated with the original payment, the original payment amount, and the account number associated with each subpayment amount, you can automatically divide the payment into distributed payments and allocate them to the designated accounts.

**Customizing Payment Failure Reason Codes**

To customize payment failure reason codes, use the PCM_OP_PYMT_POL_CHARGE policy opcode. This policy opcode provides the ability to map the online and offline payment result to the payment status and the reason IDs defined in the /strings object.

In the output flist PIN_FLD_PAYMENT_REASONS array, the array of PIN_FLD_REASON_ID fields contains the failure reasons sent by the payment processor. You can configure this policy opcode to apply fees for failed credit card and direct debit transactions based on the reason for failure.

The input flist contains a results array for a payment batch, including reasons for payment failures.

The output flist contains the payment method, transaction ID, and an array of reason IDs for failed payments.
Customizing Payment Tool

The procedures in this section describe how to add a cash reversal batch to Payment Tool and how to extend Payment Tool to automatically suspend payments that have a missing bill number.

For more information on handling externally initiated payments by using Payment Tool, see "Managing Externally Initiated Payments".

Adding a Cash Reversal Batch

Before you begin, you should be familiar with the rules for modifying the Payment Tool configuration object. See "Rules for Modifying Payment and Reversal Fields".

You can add a cash reversal batch to Payment Tool to handle any cash payments that were posted incorrectly and must be reversed from your BRM system. For example, if the currency a customer used to make a payment was later found to be counterfeit, you can remove the payment. This restores the customer’s previous account balance and removes the payment from your company’s G/L system.

The /event/billing/reversal/cash storable class exists in the BRM database; therefore, there is no need to create one. You only need to add the cash reversal entry to the /config/paymenttool object.

When you install BRM, a default /config/paymenttool object is created from data in the init_objects.source file.

1. Use the PCM_OP_WRITE_FLDS opcode to add the cash reversal entry to the /config/paymenttool class. Call the opcode using flag 32. For example:

   0 PIN_FLD_POID POID [0] 0.0.0.1 /config/paymenttool 13748
   0
   0 PIN_FLD_ACCOUNT_OBJ POID [0] 0.0.0.1 /account 1 0
   0 PIN_FLD_DESCR STR [0] "US_EN paymentTool pymnt type configuration locale"
   0 PIN_FLD_HOSTNAME STR [0] "-"
   0 PIN_FLD_NAME STR [0] "PaymentTool payment Types: Default"
   0 PIN_FLD_PROGRAM_NAME STR [0] "" 0 PIN_FLD_VERSION STR [0] ""
   0 PIN_FLD_PAY_TYPE ARRAY [10011] allocated 2, used 2
   1 PIN_FLD_NAME STR [0] "Cash" 1 PIN_FLD_PAYMNTTOOL_FIELDS ARRAY [0] allocated 4, used 4
   2 PIN_FLD_BATCH_TYPE INT [0] 1
   2 PIN_FLD_COLUMN_NAME STR [0] "Receipt Date" 2 PIN_FLD_FIELD_NAME STR [0] "PIN_FLD_EFFECTIVE_T"
   2 PIN_FLD_PURPOSE INT [0] 0
   1 PIN_FLD_PAYMNTTOOL_FIELDS ARRAY [1] allocated 4, used 4
   2 PIN_FLD_BATCH_TYPE INT [0] 1
   2 PIN_FLD_COLUMN_NAME STR [0] "Reason Code" 2 PIN_FLD_FIELD_NAME STR [0] "PIN_FLD_REASON_CODE"
   2 PIN_FLD_PURPOSE INT [0] 0
About Customizing Payment Suspense Manager

Configuring Payment Suspense Manager

2. Stop and restart the CM. See “Starting and Stopping the BRM System” in BRM System Administrator’s Guide.

All changes you make in `/config/paymenttool` are reflected in the Payment Tool UI when you restart BRM.

Customizing Suspense Criteria for Payment Tool

When you submit payments by using Payment Tool, payments that have a valid account number and a missing bill number are not marked for suspense by default.

To enable Payment Tool to return such payments as suspended, customize the `fm_pymt_pol_validate_payment_bill` function in the PCM_OP_PYMT_VALIDATE_PAYMENT opcode. Use the `validation_result` value to set the PIN_FLD_STATUS value.

For more information on managing payments with Payment Tool, see “Managing Externally Initiated Payments”.

Handling Custom Payment Methods

If you create custom payment methods for your BRM system, you must customize Payment Suspense Center to handle them. This overview procedure describes how to create custom classes and fields and enable Payment Center to handle them.

For detailed information on creating custom classes and fields, see "Creating Client Applications by Using Java PCM" and "Creating Custom Fields and Storable Classes" in BRM Developer’s Guide. For information on Storable Class Editor, see Storable Class Editor Help.

1. Complete the following tasks by using Storable Class Editor:
   a. Create your storable classes and fields in the Java PCM package.
   b. Create source files for your custom fields.

   **Note:** Storable Class Editor creates a C header file called `cust_flds.h`, a Java properties file called `InfranetPropertiesAdditions.properties`, and a Java source file for each custom field.

2. In the directory where Storable Class Editor created the Java source files, compile the source files:

   `javac -d . *.java`
3. Package the new class files into a JAR file. For example:
   
   ```bash
   jar cvf customfields.jar *.class
   ```

4. Copy the contents of the `InfranetPropertiesAdditions.properties` file and paste it into the Payment Suspense Center `Infranet.properties` file. By default, this file is located in:
   
   `C:\Program Files\Portal Software\PyntSuspCenter\PaymentCenter`

5. Append the location of the JAR file to the PAYCTRCP environment variable path. For example:
   
   `;.;C:\Program Files\Portal Software\PyntSuspCenter\customfields.jar;`

---

**Adding Multidatabase and Multischema Support in Payment Processing**

You can use Payment Center and Payment Tool in a multidatabase or multischema environment.

The multidatabase and multischema support for payment processing enables you to:

- Create and submit a single payment batch containing payments made to customer accounts residing on multiple databases.
- Create and submit a single reverse or refund payment batch containing payments made to customer accounts residing on multiple databases.
- Move the suspended payments to the payment suspense account set up for the connected database.
- Recycle the suspended payments to one or more customer accounts residing on multiple databases or schemas by using Payment Center.

---

**Important:** The multidatabase and multischema support for payment processing is applicable only for Oracle DM and is not applicable for Oracle IMDB Cache DM.

---

To add multidatabase and multischema support for payment processing:

- Create separate payment suspense accounts for each database or schema and map them to the respective `/config/psm` objects. See "Creating the Payment Suspense Account".
- Set up database links in a multidatabase system. See "Setting Up Database Links".
- Set up schema qualifications in a multischema system. See "Setting Up Schema Qualifications".

---

**Setting Up Database Links**

To add multidatabase support for payment processing, after the BRM multidatabase system is setup and running, you must set up database links for primary and secondary DMs.

To set up database links:

1. On the primary database in your BRM system, do the following:
   
   a. Open the `BRM_Home/sys/dm/pin.conf` configuration file in a text editor.
   
   b. Add the following entries:
Adding Multidatabase and Multischema Support in Payment Processing

Configuring Payment Suspense Manager

1. On the primary database in your BRM system, do the following:
   a. Open the BRM_Home/sys/dm/pin.conf configuration file in a text editor.
   b. Add the following database link entries:

      - \( \text{dm db link } \text{db_no primary } \text{primary_link_name} \)
      - \( \text{dm db link } \text{db_no backup } \text{backup_link_name} \)

      where:
      - \( \text{db_no} \) is the number of the primary database.
      - \( \text{primary_link_name} \) is the name of the primary database link.
      - \( \text{backup_link_name} \) is the name of the backup primary database link.

      For example:
      - \( \text{dm db link 0.0.0.2 primary MD_SECONDARY} \)
      - \( \text{dm db link 0.0.0.2 backup MD_SECONDARY_BACKUP} \)

   c. Set the sequence cache size:

      - \( \text{dm sequence_cache_size db_no 10000} \)

   d. Save and close the file.
   e. Stop and restart the primary DM.

2. On each secondary database in your BRM system, do the following:
   a. Open the BRM_Home/sys/dm/pin.conf configuration file in a text editor.
   b. Add the following database link entries:

      - \( \text{dm db link } \text{db_no primary } \text{primary_link_name} \)
      - \( \text{dm db link } \text{db_no backup } \text{backup_link_name} \)

      where:
      - \( \text{db_no} \) is the number of the secondary database.
      - \( \text{primary_link_name} \) is the name of the secondary database link.
      - \( \text{backup_link_name} \) is the name of the backup secondary database link.

      For example:
      - \( \text{dm db link 0.0.0.2 primary MD_SECONDARY} \)
      - \( \text{dm db link 0.0.0.2 backup MD_SECONDARY_BACKUP} \)

   c. Set the sequence cache size:

      - \( \text{dm sequence_cache_size db_no 10000} \)

   d. Save and close the file.
   e. Stop and restart the secondary DM.

Setting Up Schema Qualifications

To add multischema support for payment processing, after the BRM multischema system is set up and running, you must set up schema qualifications for all the DMs in the system.

To set up schema qualifications:

1. On the primary database in your BRM system, do the following:
   a. Open the BRM_Home/sys/dm/pin.conf configuration file in a text editor.
   b. Add the following entry:
Adding Multidatabase and Multischema Support in Payment Processing

- `dm schema db_no schemaName`

where `db_no` is the database number and `schemaName` is the name of the schema.

For example:
- `dm schema 0.0.0.2 pin02`

c. Set the sequence cache size:
- `dm dm_sequence_cache_size db_no 10000`

d. Save and close the file.

e. On the primary DM machine, run the `grant_permissions_oracle.plb` script from the UNIX prompt:

```
% cd BRM_Home/sys/dm_oracle/data
% sqlplus login/password@ORACLE_SID < grant_permissions_oracle.plb
% call grant_permissions('secondary_schema_name');
```

where `ORACLE_SID` is the BRM database alias and `secondary_schema_name` is the name of the secondary schema.

f. Stop and restart the primary DM.

2. On each secondary database in your BRM system, do the following:

a. Open the `BRM_Home/sys/dm/pin.conf` configuration file in a text editor for the secondary DM.

b. Add the following schema entry:

- `dm schema db_no schemaName`

For example:
- `dm schema 0.0.0.1 pin01`

c. Set the sequence cache size:
- `dm dm_sequence_cache_size db_no 10000`

d. Save and close the file.

e. On the secondary DM machine, run the `grant_permissions_oracle.plb` script from the UNIX prompt:

```
% cd BRM_Home/sys/dm_oracle/data
% sqlplus login/password@ORACLE_SID < grant_permissions_oracle.plb
% call grant_permissions('primary_schema_name');
```

where `ORACLE_SID` is the BRM database alias and `primary_schema_name` is the name of the primary schema.

f. Stop and restart the secondary DM.
This chapter describes how to implement the Oracle Communications Billing and Revenue Management (BRM) top-up features.

Before reading this chapter, you should be familiar with the following topics:

- "About Managing Customers" in BRM Concepts
- "About Accounts Receivable" in BRM Managing Accounts Receivable

About Topping Up Accounts

The BRM top-up features enable your customers to top up: add currency and non-currency resources to balances in their own accounts or in other customer accounts.

BRM supports two types of top-ups:

- **Standard top-ups**: Top-ups that customers make to their own accounts. See "About Standard Top-ups".
- **Sponsored top-ups**: Top-ups that are made from one customer's account to another customer's account. See "About Sponsored Top-ups".

About Standard Top-ups

A standard top-up is a top-up that a customer makes to his or her own account. For example, a customer can top up her account balance with a $50 payment from her credit card.

BRM supports two types of standard top-ups:

- **Manual standard top-ups**
  
  Manual standard top-ups are initiated by a customer service representative (CSR) using a client application or by your customers using a self-care application. For example, $100 can be added to an account balance by a CSR using Customer Center or by a customer using Self-Care Manager.

  Manual top-ups can occur at any time and can be performed on any account (no special account configuration is required). They can be used to add resources to credit balances or to debit balances. For example, a customer can add $50 to the balance associated with his prepaid mobile phone service to extend the life of the service, or he can use the top-up feature to lower the balance due on his next bill by making an early, instant payment to his account. For more information, see "Topping Up Accounts in Customer Center and Self-Care Manager".

- **Automatic standard top-ups**
**Automatic standard top-ups** are initiated by BRM, not by a CSR or a customer. They occur when an account balance falls below a specified threshold amount. When BRM rates a usage event and updates the account balance, it checks whether the balance dropped below the threshold. If it did, BRM automatically tops up the balance. For example, whenever a customer’s balance falls below $20, BRM can charge her credit card $50 and credit the balance with that amount.

To receive automatic standard top-ups, an account must have one or more services that are configured for top-ups. In addition, an automatic standard top-up payment method, amount, and cap must be set for the account. For more information, see "Implementing Automatic Standard Top-ups".

**Standard Top-up Payment Methods**

Customers can use the following payment methods for standard top-ups:

- **Credit card** or **direct debit** (manual and automatic standard top-ups)

  When a customer charges a top-up to her credit card or bank account, BRM automatically contacts a payment service for authorization via an application such as Paymentech. BRM then debits the credit card or bank account and tops up the customer’s BRM account.

- **Voucher** (manual standard top-ups only)

  When a customer uses a voucher, such as a prepaid phone card, to top up his account, the BRM API interacts with a voucher management system to validate the voucher and payment amount. The voucher management system can be Voucher Manager or a third-party voucher manager:

  - For information about Voucher Manager, see "About Managing Voucher Inventory" in *BRM Telco Integration*.

  - To configure BRM to interact with a third-party voucher system, you must customize the PCM_OP_PYMT_POL_VALID_VOUCHER policy opcode. See "Customizing Voucher Validation" in *BRM Telco Integration*.

A voucher can be used to top up one or more resources in a specified balance group (you cannot allocate a voucher’s resources to multiple balance groups). The resources can include one currency resource and an unlimited number of non-currency resources. Top-ups for currency resources are added to the existing currency sub-balance, which maintains its original validity period. Top-ups for non-currency resources are added to sub-balances according to their validity period.

For example, suppose on February 20 you apply a voucher with $50 and 100 free minutes to a balance group with a prepaid currency sub-balance of $10 that expires on March 30 and a free-minutes sub-balance of 100 with a validity period from February 1 to March 1. The balance group would now have:

- One currency sub-balance of $60 that expires on March 30.
- One free-minutes sub-balance of 100 with a validity period from February 1 to March 1.
- One free-minutes sub-balance with a validity period from February 10 to the date specified in Pricing Center.
About Topping Up Accounts

Configuring Top-ups

About Sponsored Top-ups

A sponsored top-up is a top-up that is performed by transferring resources from a balance group in one account to a balance group in another account. For example, a mother can top up her teenage son’s account with a $50 payment from her account. Resources can be transferred from a debit balance to a credit balance or a debit balance.

BRM supports two types of sponsored top-ups:

- Manual sponsored top-ups

  Manual sponsored top-ups are initiated by a CSR using a custom client application or by your customers using a custom self-care application. For example, $50 might be transferred from a mother’s account to her son’s account. Like manual standard top-ups, manual sponsored top-ups can occur at any time.

  To receive manual sponsored top-ups, an account must be a member of a sponsored top-up group. For more information, see “About Sponsored Top-up Groups” and "Implementing Manual Sponsored Top-ups".

- Automatic sponsored top-ups

  Automatic sponsored top-ups are initiated by the pin_balance_transfer utility at intervals (such as daily, weekly, or monthly) and in amounts that you specify.

  To receive automatic sponsored top-ups, an account must be a member of a sponsored top-up group. In addition, an automatic sponsored top-up amount must be specified for the group, and an automatic sponsored top-up frequency must be specified for the member account. For more information, see "About Sponsored Top-up Groups" and "Implementing Automatic Sponsored Top-ups".

About Sponsored Top-up Groups

To top up other accounts, an account must own a sponsored top-up group (/group/topup object). An account can own multiple sponsored top-up groups.

To receive top-ups from a group owner account, an account must be a member of one of the owner’s sponsored top-up groups. To be a member of a sponsored top-up group, an account must have a /topup object, and that object must be linked to the appropriate /group/topup object.

Note: The PIN_FLD_GROUP_OBJ field in a receiving account’s /topup object specifies the /group/topup object with which the receiving account is associated. If the value is NULL, the account is not associated with a /group/topup object. In such cases, the account’s /topup object is used only to enable the account to receive standard top-ups.

An account can be a member of only one sponsored top-up group at a time.
About Member Status

All accounts that belong to a sponsored top-up group have one of member statuses shown in Table 9–1:

Table 9–1  Sponsored Top-Up Group Member Statuses

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Member account can receive top-ups from the group owner account.</td>
</tr>
<tr>
<td>Inactive</td>
<td>Member account’s top-ups from the owner account are suspended, but member cannot join another sponsored top-up group.</td>
</tr>
<tr>
<td>Closed</td>
<td>Member account no longer receives top-ups from the group owner account. Member can join another sponsored top-up group.</td>
</tr>
</tbody>
</table>

Only member accounts whose member status is active can receive sponsored top-ups. For more information about member status, see these topics:

- Setting an Account’s Sponsored Top-up Member Status and PIN
- Canceling Top-ups

About member top-up PINs

Each member account in a sponsored top-up group can be assigned a top-up PIN (personal identification number). A top-up PIN is required to authorize all manual sponsored top-ups requested by the member. For more information about top-up PINs, see "Setting an Account’s Sponsored Top-up Member Status and PIN".

About Sponsored Top-up Credit Limits

Sponsored top-ups are subject to the following credit limits. When either credit limit is reached, the account cannot make any more sponsored top-ups until the credit balance is reduced.

- Credit limit of owner account’s paying balance group
  
  This credit limit controls the amount of currency and non-currency debits that can accumulate in the owner’s paying balance group.

- Credit limit of group resource
  
  Each resource supported by a sponsored top-up group has a group top-up cap. The cap specifies the maximum amount of the resource that the owner account can transfer to its members during each of the owner account’s accounting cycles.

  The cap applies to the sum of all top-ups associated with the group, not to an individual member’s top-ups. For example, if a sponsored top-up group with three members has a $90 cap for US dollars, the cap can be reached as follows:
– All three members receive $30.
– Member A receives $50, and member B receives $40.
– Member A receives $90.

To set this credit limit, see "Implementing Manual Sponsored Top-ups".

---

**Note:** Member accounts do not have *individual* sponsored top-up credit limits.

---

**Sponsored Top-up Limitations**

Sponsored top-ups cannot be made between the following accounts:

- Accounts with different primary currencies
- Accounts in different databases in a BRM multidatabase system
- Accounts in different brands

---

**About Top-up Discount Incentives**

You can offer discount incentives to customers whenever they top up an account balance by a specified amount. For example, you can offer 60 free off-peak minutes for every $100 top-up.

For more information, see "Offering Discount Incentives with Top-ups".

---

**Implementing Top-ups in Custom Client Applications**

To implement top-ups in your custom client application, see these topics:

- Implementing Manual Standard Top-ups
- Implementing Automatic Standard Top-ups
- Implementing Manual Sponsored Top-ups
- Implementing Automatic Sponsored Top-ups

---

**Note:** By default, Customer Center and Self-Care Manager are configured to perform manual standard top-ups. See "Topping Up Accounts in Customer Center and Self-Care Manager".

---

**Implementing Manual Standard Top-ups**

To implement the manual standard top-up feature, configure your custom client application to accept the following top-up information and pass it in the input flist to the PCM_OP_PYMT_TOPUP opcode:

- For voucher top-ups, pass the following information in the PIN_FLD_VOUCHERS_INFO array:
  - Voucher serial number
  - Voucher PIN number
  - Bill unit to top up
  - Balance group to top up
Implementing Top-ups in Custom Client Applications

- Service to top up

**Note:** To apply the voucher top up to a service-level balance group, you must pass the /service object POID in the PIN_FLD_SERVICE_OBJ field of the PIN_FLD_VOUCHERS_INFO array.

- For credit card top ups, pass the following information in the PIN_FLD_TOPUP_INFO array:
  - Bill unit to top up
  - Balance group to top up
  - Credit card number
  - Credit card expiration date
  - Credit card owner’s name and address information.

- For direct debit top ups, pass the following information in the PIN_FLD_TOPUP_INFO array:
  - Bill unit to top up
  - Balance group to top up
  - Bank routing number
  - Bank account number
  - Direct debit owner’s name and address information.

For more information, see "How PCM_OP_PYMT_TOPUP Handles Manual Standard Top-ups".

**Implementing Automatic Standard Top-ups**

To implement the automatic standard top-up feature:

1. In the Pricing Center, set a credit floor and credit threshold for any plan that contains deals whose service balances you want your customers to top up. For example, to trigger an automatic standard top-up when an account balance falls below $30, set the credit floor to -100 and the threshold to 70%.

   For more information, see "About Credit Thresholds and Credit Floors" in BRM Setting Up Pricing and Rating.

2. Configure your custom client application to accept the following top-up information:

   - **Payment method**
     To specify the payment method for automatic standard top-ups, set the appropriate value in the PIN_FLD_PAYINFO_OBJ field in the PIN_FLD_TOPUP_INFO array of the called opcode’s input flist (see step 3 for opcode names).

     See "Standard Top-up Payment Methods".

   - **Automatic top-up amount**
     To specify the payment amount of each automatic standard top-up, set the appropriate value in the PIN_FLD_TOPUP_AMT field in the PIN_FLD_TOPUP_INFO array of the called opcode’s input flist.
Implementing Top-ups in Custom Client Applications

- **Automatic top-up cap**
  To specify the aggregate amount of automatic standard top-ups that the account can receive during an accounting cycle, set the appropriate value in the PIN_FLD_TOPUP_CAP field in the PIN_FLD_TOPUP_INFO array of the called opcode’s input flist.

- **Bill unit to top up**
  To specify the bill unit (/billinfo object) that contains the balance to top up, set the appropriate value in the PIN_FLD_BILLINFO field in the PIN_FLD_TOPUP_INFO array of the called opcode’s input flist.

3. Pass the information to one of these opcodes:
   - PCM_OP_CUST_COMMIT_CUSTOMER when creating accounts
   - PCM_OP_CUST_UPDATE_CUSTOMER when modifying accounts

---

**Implementing Manual Sponsored Top-ups**

To implement the manual sponsored top-up feature:

1. Configure your custom client application to accept the following top-up information:
   - The POID of the account that *initiates* the creation or modification of the top-up configuration. Set this value in the level-one PIN_FLD_POID field of the called opcode’s input flist (see step 2 for opcode names).
   - The POID of the account that will receive the top-ups (the member account). Set this value in the PIN_FLD_ACCOUNT_OBJ field in the PIN_FLD_TOPUP_INFO array of the called opcode’s input flist.
   - The POID of the /topup object associated with the account that will receive the top-ups. If the account does not have a /topup object, specify the POID type. Set this value in the PIN_FLD_POID field in the PIN_FLD_TOPUP_INFO array of the called opcode’s input flist.
   - **Group to add member to**
     To specify the sponsored top-up group to add the member account to, set the POID of the appropriate /group/topup object in the PIN_FLD_POID field in the PIN_FLD_GROUP_TOPUP_INFO array of the called opcode’s input flist.
     If the object does not exist, pass the POID type.
     For information about how BRM searches for existing groups if the POID is not specified, see "Finding Sponsored Top-up Groups".
   - **Group name**
     To specify the name of the sponsored top-up group, set the appropriate value in the PIN_FLD_NAME field in the PIN_FLD_GROUP_TOPUP_INFO array of the called opcode’s input flist.

---

**Note:** Both of these opcodes call PCM_OP_CUST_SET_TOPUP, which is a wrapper opcode that calls other standard opcodes to set up or modify top-up information.

For more information, see "How BRM Sets Up Top-up Information for an Account".
If you do not specify a group name, the group will be named default. See “Finding Sponsored Top-up Groups”.

- **Group owner**
  To specify the sponsored top-up group owner account, set the POID of the owner account in the PIN_FLD_PARENT field in the PIN_FLD_GROUP_TOPUP_INFO array of the called opcode’s input flist.

- **Paying balance group**
  To specify the owner account’s balance group to transfer top-up resources from, set the POID of the balance group in the PIN_FLD_BAL_GRP_OBJ field in the PIN_FLD_GROUP_TOPUP_INFO array of the called opcode’s input flist.

- **Resources to top up**
  To specify the resources to be topped up in the member account, set the IDs of the appropriate currency and non-currency resources in the PIN_FLD_RESOURCE_ID fields in the PIN_FLD_GROUP_TOPUP_LIMITS array of the called opcode’s input flist.

- **Resource top-up cap**
  To specify the maximum amount of a resource that the owner can transfer to its members during the owner’s accounting cycle, set the appropriate value in the resource’s PIN_FLD_TOPUP_CAP field in the PIN_FLD_GROUP TOPUP_LIMITS array.

  **Important:** This cap applies to the sum of all top-ups in the group, not to an individual member’s top-ups. The cap should not exceed the credit limit of the paying balance group. See "About Sponsored Top-up Credit Limits”.

- **Receiving balance group**
  To specify the member account’s balance group to transfer sponsored top-ups to, set the POID of the balance group in the member’s PIN_FLD_BAL_GRP_OBJ field in the PIN_FLD_GROUP TOPUP_MEMBERS array of the called opcode’s input flist.

- **Top-up PIN**
  To specify the member’s sponsored top-up PIN, set the appropriate value in the PIN_FLD_PIN field in the PIN_FLD_GROUP TOPUP_MEMBERS array of the called opcode’s input flist. See "About member top-up PINs”.

- **Top-up status**
  To specify the member’s sponsored top-up status, set the appropriate value in the PIN_FLD_STATUS field in the PIN_FLD_GROUP TOPUP_MEMBERS array of the called opcode’s input flist. See "About Member Status”.

2. Pass the information to one of these opcodes:

- PCM_OP_CUST_COMMIT_CUSTOMER when creating accounts
- PCM_OP_CUST_UPDATE_CUSTOMER when modifying accounts
Implementing Automatic Sponsored Top-ups

To implement the automatic sponsored top-up feature:

1. Configure your custom client application to accept the following top-up information:
   - All top-up information listed in "Implementing Manual Sponsored Top-ups".
   - Automatic top-up amount
     To specify the amount of a resource to transfer from the owner to the member during each automatic sponsored top-up, set the appropriate value in the resource’s PIN_FLD_TOPUP_AMT field in the PIN_FLD_GROUP_TOPUP_LIMITS array.
     This amount applies to every member in the group.
   - Automatic top-up frequency
     To specify the number of days in the member’s automatic sponsored top-up cycle, set the appropriate value in the member’s PIN_FLD_TOPUP_INTERVAL field in the PIN_FLD_GROUP_TOPUP_MEMBERS array of the called opcode’s input flist (see step 2 for opcode names).
     This interval applies only to the member account.

2. Pass the information to one of these opcodes:
   - PCM_OP_CUST_COMMIT_CUSTOMER when creating accounts
   - PCM_OP_CUST_UPDATE_CUSTOMER when modifying accounts

   Note: Both of these opcodes call PCM_OP_CUST_SET_TOPUP, which is a wrapper opcode that calls other standard opcodes to set up or modify top-up information.

For more information, see "How BRM Sets Up Top-up Information for an Account".

How BRM Sets Up Top-up Information for an Account

After your custom client application passes the required top-up information to PCM_OP_CUST_COMMIT_CUSTOMER or PCM_OP_CUST_UPDATE_CUSTOMER, that opcode calls PCM_OP_CUST_SET_TOPUP.

To create or modify an account’s top-up information, PCM_OP_CUST_SET_TOPUP calls one of the following opcodes:

- PCM_OP_CUST_CREATE_TOPUP
- PCM_OP_CUST_MODIFY_TOPUP

See "Creating or Modifying an Account’s Top-up Information".
How BRM Sets Up Top-up Information for an Account

If successful, the PCM_OP_CUST_SET_TOPUP output flist contains the PIN_FLD_POID set to the POID of the created or modified /topup object.

If unsuccessful, the PCM_OP_CUST_SET_TOPUP output flist contains the following:

- PIN_FLD_FIELD_NUM set to the field that failed.
- PIN_FLD_TYPE set to the type of field that failed.
- PIN_FLD_RESULT set to the validation error code.

For standard top-ups, an account’s top-up information is stored in one object (/topup), but for sponsored top-ups, an account’s top-up information is stored in two objects (/topup and /group/topup). Sometimes, one of the objects must be created and the other modified. For example, to add a member account to a sponsored top-up group, BRM might need to create a /topup object for the member and associate it with an existing /group/topup object. To determine which opcode to call in such cases, PCM_OP_CUST_SET_TOPUP uses these rules:

- If at least one of the objects is being created, PCM_OP_CUST_SET_TOPUP calls PCM_OP_CUST_CREATE_TOPUP. This is true even if the other object is being modified.
- If neither object is being created, PCM_OP_CUST_SET_TOPUP calls PCM_OP_CUST_MODIFY_TOPUP.

Preparing an Account’s Top-up Information

To prepare top-up information for an account, PCM_OP_CUST_CREATE_TOPUP and PCM_OP_CUST_MODIFY_TOPUP call the PCM_OP_CUST_POL_PREP_TOPUP policy opcode. The policy opcode prepares information required to perform one of these tasks:

- **Create a standalone /topup object for standard top-ups.** This occurs when the following information is not passed to PCM_OP_CUST_POL_PREP_TOPUP:
  - A /topup object POID
  - A sponsored top-up group owner account POID

- **Modify a standalone /topup object for standard top-ups.** This occurs when the following information is passed to PCM_OP_CUST_POL_PREP_TOPUP:
  - A /topup object POID

But this information is not passed to it:
  - A sponsored top-up group owner account POID
  - A /group/topup POID

- **Create one or both objects (/topup and /group/topup) for sponsored top-ups.**
  - A sponsored top-up group owner account POID

But this information is not passed to it:
  - A /topup object POID
  - A /group/topup POID
Modify both objects (/topup and /group/topup) for sponsored top-ups. This occurs when the following information is passed to PCM_OP_CUST_POL_PREP_TOPUP:

- A /topup object POID
- A /group/topup POID

Additional Preparation for Sponsored Top-ups

For sponsored top-ups, PCM_OP_CUST_POL_PREP_TOPUP also prepares this information:

- If the group owner account did not initiate the object creation or modification, the policy opcode sets the following values in its output flist:
  - PIN_FLD_STATUS (member account’s group membership status) = the value associated with the PIN_STATUS_INACTIVE status in the BRM_Home/include/ops/pcm.h header file
  - PIN_FLD_PIN (member account’s top-up PIN) = NULL

- If the group owner account did initiate the object creation or modification, the policy opcode does the following:
  - If the status of the member account’s group membership is not specified in the input flist, sets it to the value associated with the PIN_STATUS_ACTIVE in the BRM_Home/include/ops/pcm.h header file
    - (Creation only) If the group name is not specified in the input flist, sets the name to default

For more information, see "Setting an Account’s Sponsored Top-up Member Status and PIN".

Validating an Account’s Top-up Information

To validate the top-up information prepared for an account (see "Preparing an Account’s Top-up Information"), PCM_OP_CUST_CREATE_TOPUP and PCM_OP_CUST_MODIFY_TOPUP call the PCM_OP_CUST_POL_VALID_TOPUP policy opcode. The policy opcode performs these tasks:

- Verifies that the status of the account to be debited for each top-up (the paying account) is active.
- Verifies that the standard or sponsored top-up amount is less than or equal to the corresponding top-up cap.
- (Sponsored top-ups only) Verifies that the member is not trying to join a group that it owns.
- (Sponsored top-ups only) Verifies that the prospective member’s account is not closed.
- (Sponsored top-ups only) Verifies that the prospective member is not a member of any other sponsored top-up group.
You can customize `PCM_OP_CUST_POL_VALID_TOPUP` to change the way it validates the output flist of the `PCM_OP_CUST_POL_PREP_TOPUP` policy opcode.

In its own output flist, `PCM_OP_CUST_POL_VALID_TOPUP` returns a `PIN_FLD_RESULT` value that is associated with one of the following values:

- **PIN_RESULT_PASS** (validation succeeded)
- **PIN_RESULT_FAIL** (validation failed)

The associated values are defined in the `pcm.h` header file. For more information, see "Header Files" in *BRM Developer’s Guide*.

### Creating or Modifying an Account’s Top-up Information

If validation succeeds (see "Validating an Account’s Top-up Information"), the validated information is used to perform one of the following tasks:

- Creating Top-up Information
- Modifying Top-up Information

### Creating Top-up Information

`PCM_OP_CUST_CREATE_TOPUP` uses the validated information to perform one of these operations:

- If the information does not include the POID of a sponsored top-up group owner account, the opcode creates a standalone `/topup` storable object that contains information for automatic standard top-ups.

- If the information includes the POID of a sponsored top-up group owner account and `/group/topup` and `/topup` POID types, the opcode creates a sponsored top-up relationship as follows:
  1. Creates a `/group/topup` object for the owner account
  2. Creates a `/topup` object for the member account
  3. Associates the new `/topup` object with the new `/group/topup` object

- If the information includes the POID of a sponsored top-up group owner account, the POID of an existing `/group/topup` object, and a `/topup` POID type, `PCM_OP_CUST_CREATE_TOPUP` creates a sponsored top-up relationship as follows:
  1. Creates a `/topup` object for the member account
  2. Associates the new `/topup` object with the existing `/group/topup` object

- If the information includes the POID of a sponsored top-up group owner account, a `/group/topup` POID type, and an existing `/topup` object, `PCM_OP_CUST_CREATE_TOPUP` creates a sponsored top-up as follows:
  1. Creates a `/group/topup` object for the owner account
  2. Associates the existing `/topup` object with the new `/group/topup` object

If successful, the `PCM_OP_CUST_CREATE_TOPUP` output flist contains the following:

- `PIN_FLD_POID` set to the POID of the `/topup` object created

If unsuccessful, the output flist contains the following:

- `PIN_FLD_FIELD_NUM` set to the field that failed
- `PIN_FLD_TYPE` set to the type of field that failed
Modify Top-up Information

PCM_OP_CUST_MODIFY_TOPUP uses the validated information to perform one of these operations:

- If the information does not include the POID of an existing /group/topup object, the opcode modifies the automatic standard top-up information in a standalone /topup object.
- If the information includes the POID of an existing /group/topup object, the opcode modifies the sponsored top-up information in the /group/topup and /topup objects.

If successful, the PCM_OP_CUST_MODIFY_TOPUP output flist contains the following:

- PIN_FLD_POID set to the POID of the /topup object modified

If unsuccessful, the PCM_OP_CUST_MODIFY_TOPUP output flist contains the following:

- PIN_FLD_FIELD_NUM set to the field that failed
- PIN_FLD_TYPE set to the type of field that failed
- PIN_FLD_RESULT set to the validation error code

Setting an Account’s Sponsored Top-up Member Status and PIN

If an account is a member of a sponsored top-up group, its group status and top-up PIN are set as follows:

- When an owner account initiates adding a member to the group, the member’s group status and PIN are set according to the information in the PCM_OP_CUST_SET_TOPUP input flist. If member status is not specified in the flist, it is set to active.
- When a member account initiates adding itself to a group, the member’s group status is set to inactive and the PIN is set to NULL no matter what values are provided for those items in the input flist. After the member is added to the group, the group owner must activate the member and sets its top-up PIN. See "Activating Sponsored Top-up Group Members" and "Setting Sponsored Top-up Member PINs".
- When a member account initiates a modification of its sponsored top-up settings after being added to a group, its group status is reset to inactive. This occurs even if the modification is not related to member status. The group owner must then reactivate the member. See "Activating Sponsored Top-up Group Members".

For related information, see these topics:

- Inactivating Sponsored Top-up Group Members
- Reinstating Sponsored Top-ups

Activating Sponsored Top-up Group Members

To receive sponsored top-ups, a member’s group status must be active (see "About Member Status"). By default, only groups owners can activate a member. To enable members to activate themselves, customize the PCM_OP_CUST_POL_PREP_TOPUP policy opcode.
To activate members whose group status is *inactive* or *closed*:

1. Use your custom client application to call PCM_OP_CUST_SET_TOPUP.
2. Set the member’s PIN_FLD_STATUS field in the MEMBERS array of the opcode’s input flist to the value associated with the PIN_STATUS_ACTIVE status in the `BRM_Home/include/ops/pcm.h` header file.

**Inactivating Sponsored Top-up Group Members**

A member whose group status is inactive can use any outstanding topped-up credit in its topped-up balance group, but it cannot receive any more top-ups from the group until its member status is reactivated. In addition, it cannot join another sponsored top-up group.

To inactivate a member’s group status:

1. Use your custom client application to call PCM_OP_CUST_SET_TOPUP.
2. Set the member’s PIN_FLD_STATUS field in the MEMBERS array of the opcode’s input flist to the value associated with the PIN_STATUS_INACTIVE status in the `BRM_Home/include/ops/pcm.h` header file.

---

**Note:** You can also inactivate sponsored top-ups by changing the status of the member *account* to inactive. To change the status of an account, see "Changing Account and Service Status" in *BRM Managing Customers*.

---

To cancel a sponsored top-up relationship, see "Canceling Top-ups".

For more information about member status, see "About Member Status".

**Inactivating all the members in a sponsored top-up group**

To inactivate the group status of every member in a group, change the status of the sponsored top-up group owner *account* to inactive.

When a group owner account is inactive, the members of its sponsored top-up groups can use any outstanding topped-up credit in their topped-up balance groups, but they cannot receive any more top-ups from the group until the owner account is reactivated, and they cannot join another sponsored top-up group.

To change the status of an account, see "Changing Account and Service Status" in *BRM Managing Customers*.

**Setting Sponsored Top-up Member PINs**

By default, only groups owners can set a member’s PIN.

To assign a top-up PIN to a member:

1. Use your custom client application to call PCM_OP_CUST_SET_TOPUP.
2. Set the member’s PIN_FLD_PIN field in the MEMBERS array of the opcode’s input flist to the appropriate string value.
Finding Sponsored Top-up Groups

When setting up sponsored top-ups, the PCM_OP_CUST_POL_PREP_TOPUP policy opcode uses the following information from the PCM_OP_CUST_SET_TOPUP input flist to determine whether the prospective member account can be added to an existing group:

- The group owner account POID (PIN_FLD_PARENT)
- The name of the group (PIN_FLD_NAME)

Note: Each group owned by the same account must have a unique name. Groups owned by different accounts can have the same name.

If a group name is not provided, the policy opcode searches for a group by owner account POID and the ID of the resource or resources that you want to top-up in the member account (PIN_FLD_RESOURCE_ID in the LIMITS array). The search has the following results:

- If the policy opcode finds the group by name but a resource is specified in the input flist that the group does not support, the following occurs:
  - If the group owner account initiated the transaction, the resource is added to the group.
  - If the member account initiated the transaction, an error is returned.

- If the policy opcode finds the group by resource and the search returns multiple groups, the groups are listed alphabetically by PIN_FLD_NAME value and the member is added to the group at the top of the list.

- If the policy opcode fails to find a group by name or by resource, the following occurs:
  - If the group owner account has a group named default, the member is added to that group.
  - If the group owner account does not have a group named default, such a group is created based on the information in the input flist. (Each group owner can have only one sponsored top-up group named default.)

Note: To change the way the search is performed, customize the PCM_OP_CUST_POL_PREP_TOPUP policy opcode.
About Tracking Sponsored Top-up Adjustments

To differentiate sponsored top-up adjustments (/event/billing/adjustment/account objects) from other types of account adjustments, the following reason codes and domain IDs have been added to the reasons.locale file:

- **Sponsored top-up debit reason code 4 and domain ID 1**

  ```
  DOMAIN = "Reason Codes-Debit Reasons" ;
  STR
  ID = 4 ;
  VERSION = 1 ;
  STRING = "Sponsored Topup. Sponsor Debit" ;
  EVENT-GLID
  "/event/billing/adjustment/account" 105 ;
  EVENT-GLID-END
  END
  ```

- **Sponsored top-up credit reason code 5 and domain ID 8**

  ```
  DOMAIN = "Reason Codes-Credit Reasons" ;
  STR
  ID = 5 ;
  VERSION = 8 ;
  STRING = "Sponsored Topup. Sponsoree Credit" ;
  EVENT-GLID
  "/event/billing/adjustment/account" 105 ;
  EVENT-GLID-END
  END
  ```

The following definitions for these new reason codes and domain IDs have been added to the pin_pynt.h file in the BRM_Home/include directory:

- **Sponsored top-up reason code definitions**

  ```
  #define PIN_REASON_ID_TOPUP_CREDIT 5
  #define PIN_REASON_ID_TOPUP_DEBIT 4
  ```

- **Sponsored top-up reason domain ID definitions**

  ```
  #define PIN_PYMT_TOPUP_CREDIT_REASON_DOMAIN_ID 8
  #define PIN_PYMT_TOPUP_DEBIT_REASON_DOMAIN_ID 1
  ```

The new reason codes and domain IDs are used by the following opcodes:

- PCM_OP_PYMT_TOPUP
- PCM_OP_BILL_TRANSFER_BALANCE
- PCM_OP_AR_ACCOUNT_ADJUSTMENT

### Customizing and Loading Sponsored Top-up Reason Codes

You can customize the default reason codes used for sponsored top-up adjustments as follows:

- Change the G/L ID event mapping. (If you change the G/L ID mapping, be sure the G/L IDs you define in the reasons.locale and pin_glid files match.)
- Change the reason code domain identifier (version number).
- Change the reason string.
To customize the default reason codes, edit the reasons.en_US sample file in the BRM_Home/sysmsgs/reasoncodes directory.

To load the contents of the customized reasons.en_US file into the /strings and /config/map_glid objects, use the load_localized_strings utility.

To run the load_localized_strings utility, use this command:
```
load_localized_strings reasons.locale
```

Note: If you load a localized version of this file, use the correct file extension for your locale. For a list of file extensions, see Locale names.

Caution: The load_localized_strings utility overwrites the /config/map_glid object. If you are updating this object, you cannot load new G/L ID maps only. You must load complete sets of data each time you run the load_localized_strings utility. This is also true if you specify the -f parameter when updating the /strings object. Otherwise, the load_localized_strings utility appends the new data to the /strings object.

For more information about loading the reasons.locale file, see "Loading Localized or Customized Strings" in BRM Developer’s Guide.

For information about creating new strings for this file, see "Creating New Strings and Customizing Existing Strings" in BRM Developer’s Guide.

Offering Discount Incentives with Top-ups

You can offer customers discount incentives whenever they top up an account balance by a specified amount. For example, you can offer 20 free peak minutes whenever a customer makes a $50 top-up.

To offer top-up discount incentives:

1. Set up the discount in Pricing Center. See "Using Pricing Center to Configure Discounts" in BRM Configuring Pipeline Rating and Discounting.

2. Configure the PCM_OP_PYMT_POL_PURCHASE_DEAL policy opcode to validate the discount.

   By default, this policy opcode is an empty hook provided to facilitate customization. You can customize it to apply discounts to account balances when they are topped up. For example, this policy opcode might grant 60 free minutes of usage for every $50 top-up.

   PCM_OP_PYMT_POL_PURCHASE_DEAL is called by PCM_OP_PYMT_TOPUP. See "How BRM Performs Top-ups".

How BRM Performs Top-ups

All top-ups are performed by PCM_OP_PYMT_TOPUP. This section describes how that opcode works.
Triggering PCM_OP_PYMT_TOPUP

PCM_OP_PYMT_TOPUP is triggered as follows:

- **Manual top-ups**: When a customer or CSR uses a client application to top up a balance in an account (a manual top-up), the opcode is called by the client application.
  - To use Customer Center or Self-Care Manager to top up accounts, see "Topping Up Accounts in Customer Center and Self-Care Manager".
  - To use a custom client application to top up accounts, see "Implementing Top-ups in Custom Client Applications".

- **Automatic standard top-ups**: When a balance in an account configured for automatic standard top-ups falls below a specified threshold, PCM_OP_PYMT_TOPUP is called by the PCM_OP_ACT_POL_EVENT_NOTIFY policy opcode. To configure accounts to receive automatic standard top-ups, see "Implementing Automatic Standard Top-ups".

- **Automatic sponsored top-ups**: When the next automatic top-up date of an account configured for automatic sponsored top-ups is within the time range specified in the "pin_balance_transfer" utility’s command-line parameters, PCM_OP_PYMT_TOPUP is called by the utility. See "Performing Automatic Sponsored Top-ups".
  To configure accounts to receive automatic sponsored top-ups, see "Implementing Automatic Sponsored Top-ups".

Performing Top-ups with PCM_OP_PYMT_TOPUP

For information about how PCM_OP_PYMT_TOPUP performs top-ups, see these topics:

- How PCM_OP_PYMT_TOPUP Handles Automatic Standard Top-ups
- How PCM_OP_PYMT_TOPUP Handles Manual Sponsored Top-ups
- How PCM_OP_PYMT_TOPUP Handles Automatic Sponsored Top-ups

How PCM_OP_PYMT_TOPUP Handles Manual Standard Top-ups

PCM_OP_PYMT_TOPUP performs manual standard top-ups as follows:

1. Receives the top-up amount from a client application.

2. For top-ups paid with a voucher, calls the PCM_OP_PYMT_POL_VALID_VOUCHER policy opcode to use Voucher Manager or a third-party voucher management system to perform these operations:
   a. Validate the voucher.
   b. Associate the voucher with the account.
   c. Retrieve the balance impacts of the voucher’s resources.

After the voucher is validated, the PCM_OP_PYMT_POL_VALID_VOUCHER policy opcode performs these operations:

a. Determines whether the voucher has a currency resource.

b. Uses the resource with the earliest validity start date and the resource with the latest validity end date to determine the validity period of the voucher.
c. Returns the preceding information and the voucher’s balance impacts to PCM_OP_PYMT_TOPUP.

PCM_OP_PYMT_TOPUP then performs these operations:

a. If the validated voucher does not have a currency resource, creates an /event/billing/vouchertopup event to record the top-up balance impact.

b. If the voucher has a currency resource, calls PCM_OP_PYMT_COLLECT, which passes the balance group impacted by the top up and the top-up resource information and in its input flist to PCM_OP_BILL_RCV_PAYMENT.

If there are non-currency balance impacts in the top-up resource information, PCM_OP_BILL_RCV_PAYMENT adds them to the input flist of PCM_OP_ACT_USAGE, which records them in the /event/billing/payment/voucher object that it generates to record the top-up balance impact.

Note: When called by PCM_OP_PYMT_TOPUP, PCM_OP_PYMT_COLLECT does not call the payment suspense manager PCM_OP_PYMT_VALIDATE_PAYMENT opcode.

c. Posts the top-up as an unallocated payment.

3. For top-ups paid by credit card or direct debit, collects payment from the credit card agency or direct debit company, and then updates the specified balance.

4. For all top-ups, applies any top-up discount incentives to the account by calling the PCM_OP_PYMT_POL_PURCHASE_DEAL policy opcode. See "Offering Discount Incentives with Top-ups".

Note: The PIN_FLD_DESCR field in the input flist contains a description of the topup. This value is stored in the /event/billing/payment/cc object and is used by Customer Center when making a “One time credit card” or “Charge credit card now” payment.

How PCM_OP_PYMT_TOPUP Handles Automatic Standard Top-ups

PCM_OP_PYMT_TOPUP performs automatic standard top-ups as follows:

1. Retrieves the top-up amount from the specified /topup object.

2. Verifies that the top-up amount will not cause the sum of all automatic standard top-ups received during the current accounting cycle to exceed the account’s automatic standard top-up cap.

3. Applies any top-up discount incentives to the account by calling the PCM_OP_PYMT_POL_PURCHASE_DEAL policy opcode. See "Offering Discount Incentives with Top-ups".

How PCM_OP_PYMT_TOPUP Handles Manual Sponsored Top-ups

PCM_OP_PYMT_TOPUP performs manual sponsored top-ups as follows:

1. Receives the top-up amount from a client application.

2. Verifies the following:
   - The status of the member is active. See "About Member Status".
How BRM Performs Top-ups

1. (Member-initiated top-ups only) The top-up PIN is valid. See "About member top-up PINs".

2. The top-up amount will not cause the total amount of credit charged to the owner account’s balance group to exceed the credit limit of the associated resource in that balance group. See "About Sponsored Top-up Credit Limits".

3. The top-up amount will not cause the sum of all top-ups received during the current accounting cycle of the owner account to exceed the group’s top-up cap.

3. Performs these operations:
   - Calls PCM_OP_BILL_TRANSFER_BALANCE to transfer the top-up resources from the paying balance group to the receiving balance group. See "About Transferring Sponsored Top-ups from Debit Balances".
   - Passes the reason ID and the reason domain ID used to differentiate sponsored top-up adjustments from other types of adjustments to PCM_OP_BILL_TRANSFER_BALANCE, which passes them to PCM_OP_AR_ACCOUNT_ADJUSTMENT. See "About Tracking Sponsored Top-up Adjustments".
   - Updates the sum of all sponsored top-ups credited to members of the group during the group owner account’s current accounting cycle. This value is stored in the PIN_FLD_CYCLE_TOPPED_AMT field of the LIMITS array in the /group/topup object.
     - If the last sponsored top-up occurred in the owner account’s current accounting cycle, PCM_OP_PYMT_TOPUP adds the amount of the current top-up to the value already in this field.
     - If the last sponsored top-up occurred in the owner account’s previous accounting cycle, the opcode sets this field to the amount of the current top-up.

4. Applies any top-up discount incentives to the account by calling the PCM_OP_PYMT_POL_PURCHASE_DEAL policy opcode. See "Offering Discount Incentives with Top-ups".

How PCM_OP_PYMT_TOPUP Handles Automatic Sponsored Top-ups

PCM_OP_PYMT_TOPUP performs automatic sponsored top-ups as follows:

1. Retrieves the top-up amount from the specified /group/topup object.

2. Verifies the following:
   - The status of the member is active. See "About Member Status".
   - The top-up amount will not cause the total amount of credit charged to the owner account’s balance group to exceed the credit limit of the associated resource in that balance group. See "About Sponsored Top-up Credit Limits".
   - The top-up amount will not cause the sum of all top-ups received during the current accounting cycle of the owner account to exceed the group’s top-up cap.

3. Performs these operations:
   - Calls PCM_OP_BILL_TRANSFER_BALANCE to transfer the top-up resources from the paying balance group to the receiving balance group. See "About Transferring Sponsored Top-ups from Debit Balances".
How BRM Performs Top-ups

- Passes the reason ID and the reason domain ID used to differentiate sponsored top-up adjustments from other types of adjustments to PCM_OP_BILL_TRANSFER_BALANCE, which passes them to PCM_OP_AR_ACCOUNT_ADJUSTMENT. See "About Tracking Sponsored Top-up Adjustments".

- Updates the time that the member’s last automatic sponsored top-up occurred to the current time.

  This value is stored in the PIN_FLD_LAST_TOPUP_T field of the LIMITS array in the /group/topup object. PCM_OP_PYMT_TOPUP uses this value to determine when to execute the member’s next automatic sponsored top-up.

- Calculates the time that the member’s next automatic sponsored top-up will occur. This value is stored in the PIN_FLD_NEXT_TOPUP_T field of the LIMITS array in the /group/topup object.

- Updates the sum of all sponsored top-ups credited to members of the group during the group owner account’s current accounting cycle. This value is stored in the PIN_FLD_CYCLE_TOPPED_AMT field of the LIMITS array in the /group/topup object.

  If the last sponsored top-up occurred in the owner account’s current accounting cycle, PCM_OP_PYMT_TOPUP adds the amount of the current top-up to the value already in this field.

  If the last sponsored top-up occurred in the owner account’s previous accounting cycle, the opcode sets this field to the amount of the current top-up.

4. Applies any top-up discount incentives to the account by calling the PCM_OP_PYMT_POL_PURCHASE_DEAL policy opcode. See "Offering Discount Incentives with Top-ups".

About Transferring Sponsored Top-ups from Debit Balances

To perform a sponsored top-up, PCM_OP_BILL_TRANSFER_BALANCE must transfer resources from a debit balance in a group owner account to a debit or credit balance in a member account. By default, however, this opcode transfers resources only from credit balances. To enable the opcode to transfer resources from a debit balance, the PIN_FLD_VERIFY_BALANCE field in its input flist is set to PIN_BOOLEAN_FALSE by PCM_OP_PYMT_TOPUP.

Note: If this field is not set (default) or is set to PIN_BOOLEAN_TRUE, the opcode cannot transfer resources from debit balances.

About Retrieving Balance Impact Information for Voucher Top-ups

To retrieve the balance impacts associated with a voucher top-up, such as the tax payment and original top-up amount, use Event Browser to retrieve the balance impacts from the PIN_FLD_BAL_IMPACTS array in the /event/billing/payment/voucher object. The /event/billing/payment/voucher object contains the net amount and tax amount in separate PIN_FLD_BAL_IMPACTS arrays.

About Taxes Applied During Voucher Topups

By default, when you apply a voucher with tax to an account, BRM applies a negative balance impact to the account balance.
When you apply a voucher with tax to an account, you must set the tax to a negative value. For example, if a voucher grants $100 with -10% tax on the amount granted, BRM applies a balance impact of -100 for the voucher and +10 for the tax to the account balance. In this case, the final balance is 0 - (-100) - (+10) = $90.

**Topping Up Accounts in Customer Center and Self-Care Manager**

Manual standard top-ups are performed by a CSR using a client application such as Customer Center or by a customer using a self-care application such as Self-Care Manager.

To perform manual standard top-ups in Customer Center and Self-Care Manager, see the following topics:

- Performing Top-ups in Customer Center
- Performing Top-ups in Self-Care Manager

**Performing Top-ups in Customer Center**

To perform manual standard top-ups in Customer Center, see “Topping up accounts in Customer Center and Self-Care Manager” in *BRM Configuring and Collecting Payments*.

**Changing the default top-up payment method**

The default top-up payment method in Customer Center is voucher. To change this default, add the following parameter to the CCSDK_home/CustomerCareSDK/CustCntr/custom/Customized.properties file:

```properties
customized.default.topup.payment.method = payment_method
```

Where payment_method is one of these values:

- ONFILE (Payment method on file)
- ONETIME (One-time credit card)
- VOUCHER (Voucher)

**Note:** If this parameter is not included in the file, voucher is the default payment method.

For information about modifying the Customized.properties file, see "Modifying Behaviors Defined by the Default Properties Files" in *BRM Developer’s Guide*.

For information about standard top-up payment methods, see "Standard Top-up Payment Methods".

**Turning off “Top-up completed” message**

By default, Customer Center displays the message “Top-up completed” after you complete a top-up. If you typically perform multiple top-ups in a row and don’t want to close this message after each of them, you can prevent the message from appearing. To do so, set the following parameter in the CCSDK_home/CustomerCareSDK/CustCntr/custom/Customized.properties file to true:

```properties
customized.turn.off.topup.completed.msg = true
```

By default, this parameter is set to false.

For information about modifying the Customized.properties file, see "Modifying Behaviors Defined by the Default Properties Files" in *BRM Developer’s Guide*. 
About voucher top-up error handling

When an error occurs during a voucher top-up in Customer Center, the PCM_OP_VOUCHER_ASSOCIATE_VOUCHER opcode creates an EBufException. The EBufException includes the error type and the name of the field associated with the error in the error buffer (ebuf). Customer Center uses this information to determine which error message to display to the user.

Table 9–2 lists the default error messages that are displayed in Customer Center when errors associated with the corresponding error type and field name occur:

<table>
<thead>
<tr>
<th>Error message</th>
<th>Error type</th>
<th>Field name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voucher has already been used</td>
<td>ERR_NOT_FOUND</td>
<td>PIN_FLD_EXTENDED_INFO</td>
</tr>
<tr>
<td>Invalid voucher ID/PIN combination</td>
<td>ERR_NOT_FOUND</td>
<td>PIN_FLD_POID</td>
</tr>
<tr>
<td>Voucher has already been used or has expired</td>
<td>ERR_BAD_VALUE</td>
<td>PIN_FLD_STATE_ID</td>
</tr>
<tr>
<td>Voucher has expired</td>
<td>ERR_BAD_VALUE</td>
<td>PIN_FLD_EXPIRATION_T</td>
</tr>
<tr>
<td>Voucher and account brands do not match</td>
<td>ERR_PERMISSION_DENIED</td>
<td>PIN_FLD_BRAND_OBJ</td>
</tr>
<tr>
<td>Invalid voucher ID/PIN combination</td>
<td>ERR_BAD_ARG</td>
<td>PIN_FLD_VOUCHER_PIN</td>
</tr>
<tr>
<td>Voucher has already been used or has expired</td>
<td>ERR_BAD_ARG</td>
<td>PIN_FLD_STATE_ID</td>
</tr>
</tbody>
</table>

These error messages are stored in the CustomerCenterResources.properties file. To modify them, see "Modifying the Customer Center Properties Files" in BRM Developer’s Guide.

Performing Top-ups in Self-Care Manager

To perform manual standard top-ups in Self-Care Manager, see "Applying Voucher Top-ups" in BRM Managing Customers.

Performing Automatic Sponsored Top-ups

Automatic sponsored top-ups are initiated by the pin_balance_transfer utility. The utility triggers such top-ups for all member accounts in your system whose next automatic top-up date is within the time range specified in the utility’s command-line parameters. To perform the top-ups, the utility calls PCM_OP_PYMT_TOPUP.

For more information, see these topics:

- For an overview of automatic sponsored top-ups, see "About Sponsored Top-ups".
- To set up automatic sponsored top-ups for an account, see "Implementing Automatic Sponsored Top-ups".
- For information about how PCM_OP_PYMT_TOPUP implements top-ups, see "How BRM Performs Top-ups".

Running the pin_balance_transfer utility

To run the pin_balance_transfer utility, use a cron job with a crontab entry. The following crontab entry runs pin_balance_transfer at 1:00 a.m. daily:

```
0 1 * * * /opt/portal/7.4/bin/pin_balance_transfer &
```
About Reversing Voucher Top-ups

To reverse voucher top-ups, see these topics:

- Reversing Vouchers That Have Only Non-Currency Resources
- Reversing Vouchers That Have Currency and Non-Currency Resources

Caution: When a voucher is associated with an account balance, its state becomes used and it cannot be associated with another account or balance group. Thus, although its impact on the balance to which it was applied can be reversed, its resources cannot be reapplied to another account or balance group.

Reversing Vouchers That Have Only Non-Currency Resources

If a voucher has only non-currency resources, an /event/billing/vouchertopup event is generated when the voucher is associated with an account. To reverse the balance impact of this event, you must perform an adjustment. See "About Adjustments" in BRM Managing Accounts Receivable.

Reversing Vouchers That Have Currency and Non-Currency Resources

If a voucher has currency and non-currency resources, an /event/billing/payment/voucher event is generated when the voucher is associated with an account. To reverse the balance impact of this event, you must use testnap to perform a payment reversal (see "Using Testnap" in BRM Developer’s Guide). The payment reversal will be recorded in an /event/billing/reversal/voucher event by PCM_OP_BILL_REVERSE_PAYMENT.

About Vouchers Having Non-Currency Resources With a Positive Impact

By default, when you apply a voucher with non-currency resources to an account, a negative balance impact is applied to the account balance. For example, if a voucher grants 30 free minutes, a balance impact of -30 is applied to the customer’s account balance. As the customer uses the free minutes, the account balance approaches 0. For example, if the customer uses 20 of the 30 free minutes, the account balance becomes -10. In this case, the non-currency resource has a credit limit of 0 by default, or it can be changed to a negative value.

To have vouchers with non-currency resources apply a positive balance impact to account balances, you must set the resource’s credit limit to a positive non-zero value. For example, you must set the free minutes resource to +2.

To set the credit limit for non-currency resources to a positive value, perform one of the following:

- Specify the credit limit in your plan.
- Specify the credit limit in an account.

Viewing Sponsored Top-up History

To display information about sponsored top-up transactions, configure your custom client application to call PCM_OP_PYMT_FIND_TOPUP_EVENTS to retrieve balance transfer events (/event/billing/adjustment/account objects) in which top-up transactions are recorded.
Historic sponsored top-up information can be displayed for both sponsored top-up owner accounts and member accounts:

- Member accounts can view only the sponsored top-ups that they received.
- Owner accounts can view all sponsored top-ups associated with their sponsored top-up groups.

**Displaying All Sponsored Top-ups Associated with an Account**

To retrieve all sponsored top-up events associated with a group owner account or a member account, include the following values in the PCM_OP_PYMT_FIND_TOPUP_EVENTS input flist:

- Account’s POID in the PIN_FLD_POID field

**Note:** This field stores the POID of the account that initiates the search. If a POID type rather than an actual POID is provided, an actual balance group POID must be set in the flist’s PIN_FLD_BAL_GRP_OBJ field.

- No value in these fields:
  - PIN_FLD_BAL_GRP_OBJ

**Note:** If no account POID is provided, a balance group POID must be provided. In such cases, the opcode retrieves top-up events associated only with the specified balance group.

- PIN_FLD_REASON_ID
- PIN_FLD_REASON_DOMAIN_ID

**Displaying Sponsored Top-ups Associated with Only One Group**

If an owner account has multiple sponsored top-up groups, retrieve sponsored top-up events related only to one group by including the following values in the PCM_OP_PYMT_FIND_TOPUP_EVENTS input flist:

- Group owner account’s POID in the PIN_FLD_POID field
- Paying balance group’s POID in the PIN_FLD_BAL_GRP_OBJ field
- No value in these fields:
  - PIN_FLD_REASON_ID
  - PIN_FLD_REASON_DOMAIN_ID

**Displaying Only Sponsored Top-up Credits or Debits**

By default, PCM_OP_PYMT_FIND_TOPUP_EVENTS retrieves all sponsored top-up debit and credit adjustment events associated with the initiating account. Thus, if a sponsored top-up group owner account initiates the search, the opcode retrieves two events for each top-up:

- An event for debiting the top-up amount from the owner’s paying balance group
- An event for crediting the top-up amount to a member’s receiving balance group
To limit the search to debit or credit adjustment events, include the appropriate reason ID and reason domain ID in the PIN_FLD_REASON_ID and PIN_FLD_REASON_DOMAIN_ID fields of the opcode’s input flist. For more information, see "About Tracking Sponsored Top-up Adjustments".

---

**Note:** If you create custom reason and reason domain IDs for sponsored top-up events, PCM_OP_PYMT_FIND_TOPUP_EVENTS returns events associated with all the IDs unless you limit the search to specified IDs.

---

**Canceling Top-ups**

To cancel top-ups, see these topics:

- Canceling Sponsored Top-ups
- Deleting Accounts That Are Sponsored Top-up Owners or Members

**Canceling Sponsored Top-ups**

Sponsored top-ups can be canceled for a single member or for an entire group. For more information, see these topics:

- Canceling a Single Member’s Sponsored Top-ups
- Canceling an Entire Group’s Sponsored Top-ups

**Canceling a Single Member’s Sponsored Top-ups**

To cancel a member account’s sponsored top-ups, see these topics:

- Canceling top-ups by changing a member’s group status
- Canceling top-ups by closing a member account

To stop sponsored top-ups temporarily, see "Inactivating Sponsored Top-up Group Members".

**Canceling top-ups by changing a member’s group status**

To cancel a member account’s sponsored top-ups, change the member’s group status to `closed`. When the member’s group status is closed, the account can use any outstanding topped-up credit in its topped-up balance group, but it can no longer receive sponsored top-ups from the group. It can, however, join another sponsored top-up group.

By default, only the group owner can change a member’s group status to closed. To enable members to close their group status themselves, customize the PCM_OP_CUST_POL_PREP_TOPUP policy opcode.

To change a member’s group status to closed:

1. Use your custom client application to call PCM_OP_CUST_SET_TOPUP.
2. Set the member’s PIN_FLD_STATUS field in the PIN_FLD_GROUP_TOPUP_MEMBERS array of the opcode’s input flist to the value associated with the PIN_STATUS_CLOSED status in the BRM_Home/include/ops/pcm.h header file.

---

**Note:** This changes only the member’s group status. It does not change the member’s account status.
Canceling Top-ups

Canceling Top-ups by closing a member account
You can also cancel a member account’s sponsored top-ups by changing the account status of the member to closed. By default, when a member account is closed, its sponsored top-up group member status is set to closed. To change the status of an account, see “Changing Account and Service Status” in BRM Managing Customers.

Caution: When a member account is closed, any outstanding topped-up credit that it has is forfeited, not transferred back to the group owner account or refunded to either the owner or the member. Even if the member account’s sponsored top-ups are reactivated, the forfeited credit is not reinstated.

Canceling an Entire Group’s Sponsored Top-ups
To cancel the sponsored top-ups of every member in a group, change the account status of the sponsored top-up group owner to closed. By default, when the owner account is closed, the member status of its member accounts is set to closed.

To change the status of an account, see “Changing Account and Service Status” in BRM Managing Customers.

Reinstating Sponsored Top-ups
When an account’s sponsored top-up group member status is set to closed, its array element is not removed from the PIN_FLD_GROUP_TOPUP_MEMBERS array in the /group/topup object with which it was associated.

If you later reactivate the member’s status and want to use its old MEMBERS array element, the client application must pass the called opcode the same receiving balance group POID that was used the last time the member belonged to the group (see "Implementing Top-ups in Custom Client Applications"). Otherwise, a new array element will be created for the member account.

Note: If a lot of members have multiple MEMBERS array elements, your system’s performance may be affected.

Deleting Accounts That Are Sponsored Top-up Owners or Members
This section describes what happens to /group/topup and /topup objects when the accounts with which they are associated are deleted.

For general information about deleting accounts, see the following topics:

- "Removing Accounts by Using the sample_del.c Program" in BRM Developer’s Reference
- PCM_OP_CUST_DELETE_ACCT

Caution: Do not delete accounts in a production system.

About Deleting Owner Accounts
When a sponsored top-up group owner account is deleted, the sponsored top-ups for all its member accounts end. In addition, the following occurs:

- If a /topup object is associated with the owner account, it is deleted.
Canceling Top-ups

- All /group/topup objects associated with the owner account are deleted.
- The following fields in all /topup objects of accounts that were members of the deleted /group/topup objects are set to NULL:
  - PIN_FLD_GROUP_OBJ
  - PIN_FLD_GROUP_INDEX

About Deleting Member Accounts
When a sponsored top-up group member account is deleted, PCM_OP_CUST_DELETE_ACCT calls PCM_OP_CUST_DELETE_TOPUP, which performs these operations:
- Deletes the /topup object associated with the member account.
- Removes the member’s array element from the MEMBERS array of the /group/topup object with which the /topup object was associated.

**Important:** PCM_OP_CUST_DELETE_TOPUP should not be used to cancel an account’s membership in a sponsored top-up group. See "Canceling Top-ups".

If successful, the PCM_OP_CUST_DELETE_TOPUP output flist contains the following:
- PIN_FLD_POID set to the POID of the deleted /topup object

If unsuccessful, the PCM_OP_CUST_DELETE_TOPUP output flist contains the following:
- PIN_FLD_FIELD_NUM set to the field that failed
- PIN_FLD_TYPE set to the type of field that failed
- PIN_FLD_RESULT set to the validation error code
Handling Atypical Payments

This chapter describes how to handle payments that aren’t tailored to your normal payment processing.

For more information on payments, see the following documents:

- About Payments
- About BRM-Initiated Payment Processing

Handling Overpayments and Underpayments

If a customer pays too much or too little, your Oracle Communications Billing and Revenue Management (BRM) business policies may require payment allocation. See "About Allocating Payments".

- For underpayments, choose which bills or items to allocate the payment to.
- For overpayments, pay all items and generate a credit balance.

By default, BRM requires overpayments to be allocated.

To change the default BRM behavior for overpayments and underpayments, customize the PCM_OP_PYMT_POL_UNDER_PAYMENT or PCM_OP_PYMT_POL_OVERPAYMENT policy opcodes.

If the money received is more or less than the sum of the total due of all the open items selected by PCM_OP_PYMT_SELECT_ITEMS, PCM_OP_PYMT_SELECT_ITEMS calls PCM_OP_PYMT_POL_OVERPAYMENT or PCM_OP_PYMT_POL_UNDER_PAYMENT, respectively. PCM_OP_PYMT_SELECT_ITEMS does not call these policy opcodes if:

- The amount is equal to the sum of the total due.
- The flag PIN_BILLFLG_SELECT_FINAL is passed in.
- The flag PIN_BILLFLG_DEFER_ALLOCATION is passed in.

By default, PCM_OP_PYMT_POL_OVERPAYMENT returns the amount overpaid on the output flist. The excess amount remains in the bill unit specified on the input flist (or the default bill unit if none was specified) until they are manually redistributed by using Customer Center. You can customize PCM_OP_PYMT_POL_OVERPAYMENT to perform as a hook for an application that would search for and settle all overpaid payment items.

By default, PCM_OP_PYMT_POL_UNDER_PAYMENT pays the billed items in order they are listed on the input flist (item[0] first, then item[1], item[2], etc.). It then returns the items paid on the output flist. Items that are partially paid are returned with a new amount due. Items not paid are not returned.
Handling Late or Missed Payments

You can specify how BRM handles late or missed payments, for example, change the account status to inactive, or charge a late fee.

To change the account status, customize the PCM_OP_PYMT_POL_COLLECT policy opcode.

To charge a late fee, customize the PCM_OP_PYMT_POL_APPLY_FEE policy opcode.

Handling Multiple Payments to the Same Account

By default, you cannot use Payment Tool to apply more than one payment from a batch to a single account. However, if the account uses open item accounting, you might need to apply more than one payment to more than one bill.

To enable Payment Tool to apply multiple payments from the same batch to a single account, you edit the PaymentTool.ini file.

See "Applying Multiple Payments to the Same Account".

Applying Multiple Payments to an Account Via Payment Gateways

The PCM_OP_PYMT_COLLECT opcode enables a payment gateway to apply multiple payments from the same batch to a single account. Payment gateways need no additional configuration to enable this feature.

To avoid payment allocation errors when a batch contains multiple payments for the same account, the PCM_OP_PYMT_COLLECT opcode processes them sequentially: that is, it performs all operations on one payment before moving on to the next payment. For example, a batch contains two payments for the same account: payment 1 = $10 and payment 2 = $20. The account has three open bill items: item 1 = $5, item 2 = $3, and item 3 = $22. Hence, the total payment amount ($30) is equal to the total due amount ($30) of all the items. The PCM_OP_PYMT_COLLECT opcode allocates the payments as shown in Table 10–1:

Table 10–1  PCM_OP_PYMT_COLLECT Payment Allocation

<table>
<thead>
<tr>
<th>Operation/Remaining Total Due for Open Bill Items</th>
<th>Item 1</th>
<th>Item 2</th>
<th>Item 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Select items for payment 1 ($10). Beginning with the oldest open item (item 1), BRM selects items in chronological order until the payment is completely spent.</td>
<td>$5</td>
<td>$3</td>
<td>$22</td>
</tr>
<tr>
<td>2. Apply payment 1 ($10): $5 to Item 1 $3 to Item 2 $2 to Item 3</td>
<td>$0</td>
<td>$0</td>
<td>$20</td>
</tr>
<tr>
<td>3. Select items for payment 2 ($20). Because payment 1 has been applied, the two oldest items selected in step 1 are now paid and closed, and the amount due for item 3 has been reduced to $20.</td>
<td>$0</td>
<td>$0</td>
<td>$20</td>
</tr>
</tbody>
</table>
Handling Failed Unconfirmed Payments

An unconfirmed payment is an BRM-initiated payment that is posted as successful in BRM before receiving a confirmation from the bank or payment processor, for example an automated clearing house.

If the bank or payment processor later sends a failure notification for a BRM-initiated payment, for example, due to insufficient funds or an expired credit card, BRM reverses the initially-successful payments and posts the failed payments.

Failed unconfirmed payments can be submitted to BRM by using a payment gateway, or by using Payment Tool or another third-party payment application.

Figure 10–1 shows how failed unconfirmed payments are processed with an automated clearing house (ACH). Each step in the process is described below.

![Figure 10–1 ACH Processing of Failed Unconfirmed Payments](image)

1. The Connection Manager (CM) collects the balances due on accounts and sends the charges to the DM for processing.

   /event/billing/charge/pay_type events are recorded in BRM.

2. The Data Manager (DM) processes the charges and sends them to the ACH.

3. Before charges are confirmed by the ACH, the DM automatically sends them back to the CM as successful payments.

   /event/billing/payment/pay_type events are recorded in BRM.

For information about payment gateways, see "About Payment Gateways".
4. The ACH returns any failed unconfirmed payments in a failed payment file. The failed payment records must include the transaction ID, payment failure reason code, and result.

5. The failed payment file is loaded into Payment Tool or a third-party payment application.

6. The transaction ID, reason code, and result are validated, the failed payments are recorded, and the unconfirmed successful payments are reversed.

   Failed unconfirmed payments are recorded as /event/billing/payment/failed events, and the reversals of the initially-successful payments are recorded as /event/billing/reverse/pay_type events.

   **Note:** Both the /event/billing/payment/failed event and the /event/billing/reversal event contain the same transaction ID of the original unconfirmed payment.

When a unconfirmed payment fails, the failed payment is recorded in BRM with a balance impact of 0, and the successful payment amount is re-applied to the account balance.

To charge fees for failed unconfirmed payments, see "Configuring Payment Fees".

**Submitting Failed Unconfirmed Payments with Payment Tool**

Failed unconfirmed payment records are sent to BRM in a batch containing only failed payments. When you open the failed payment batch in Payment Tool, payments are displayed with a status “Failed Payment”.

You process failed payments the same way you process other payment batches. See "Processing a Batch of Payments by Using Payment Tool".

**Requirements for Posting Unconfirmed Payments**

Failed unconfirmed payments are processed by the PCM_OP_PYMT_COLLECT opcode. The interface you use to load failed payments into the BRM database must be configured to send the following information with each failed payment, in order to be validated in BRM:

- Transaction ID
- Failed payment status
- Failure reason ID

The transaction ID of the failed payment is compared against the transaction ID in the /event/billing/payment/pay_type object of the initial unconfirmed payment that was posted successfully in BRM.

If the transaction IDs are the same, the original payment is reversed and the failed payment is recorded. If the transaction ID is missing or incorrect, or the successful payment can’t be located in the database, the PCM_OP_PYMT_VALIDATE_PAYMENT opcode returns a value of PIN_PYMT_FAILED in the PIN_FLD_RESULT field, and an error is displayed. The payment is not reversed and the failed payment is not allocated. You must manually fix the transaction ID to resubmit the payment. Or, you can configure BRM to identify the original payment by using other payment attributes. See "Customizing Payment Fees".
Customizing Unconfirmed Payment Processing

When a failed unconfirmed payment is received, the original successful payment is identified by using its transaction ID.

The PIN_FLD_PAYMENT_TRANS_ID field in the /event/billing/payment/failed event must be equal to the PIN_FLD_TRANS_ID of the successful unconfirmed payment item. If so, the unconfirmed payment is reversed and the failed payment posted to the proper account.

If the payment processor is not able to send a transaction ID with each payment, or if the transaction ID is not a reliable means of identifying a payment, you can configure PCM_OP_PYMT_POL_VALIDATE_PAYMENT to find the original unconfirmed payment by using other payment properties. For example, you can use a combination of the payment amount, account number, and invoice number.

When the payment is received, BRM compares these values in the failed payment with those of the original unconfirmed payment, and if the values match, it will reverse the unconfirmed payment and post the failed payment.

---

**Note:** The result of the validation is the POID of the unconfirmed successful payment item that was recorded in BRM. If the item is not available, the reversal can’t occur. By default, this policy opcode retrieves the item by finding the payment event with corresponding transaction ID.
Managing Externally Initiated Payments

This chapter describes how to use Oracle Communications Billing and Revenue Management (BRM) to handle externally initiated payments, such as check or cash payments.

For information about handling BRM-initiated payments, see "About BRM-Initiated Payment Processing".

About Externally Initiated Payments

With externally initiated payments, such as payment by check or cash, an invoice is sent to the customer, who responds by sending the payment. Typically, payments are sent to a bank, and the bank sends you a list of payments that have been received and deposited. You then use Payment Tool to update the customer’s account and close outstanding payments. For more information, see "Processing a Batch of Payments by Using Payment Tool".

Figure 11–1 shows how BRM collects externally initiated payments:

Supported Externally Initiated Payment Methods

You can submit the following externally initiated payments to BRM by using Payment Tool:

- Check
- Cash
- Wire transfer
- Postal order
Processing a Batch of Payments by Using Payment Tool

Processing payments with Payment Tool typically follows this procedure. This example shows how to process a batch of payments. Processing payment reversals and refunds is a nearly identical procedure.

1. You receive a batch of check payments. (A batch of check payments is typically a record of checks received and deposited in your bank.) Review the batch of payments to determine the number of payments and the total of all payments in the batch.

   1. Open Payment Tool.
   2. If you have branding enabled, choose the brand containing the accounts you need to process. (You can work in only one brand at a time.)
   3. Choose the type of payment batch to create; for example, check payments.
   4. Enter the following batch properties:
      - The number of payments that you are processing.
      - (Optional) The projected total for the payments in the batch. You use this total to validate that you have entered all the payments.
      - (Optional) The currency.
      - (Optional) The payment channel.
      - (Optional) The effective date of the batch. You can backdate a batch, but not prior to the date of the last posted general ledger transaction report.
      - (Optional) The Lockbox ID and Lockbox date.
      - The allocation level. You can allocate payments to specific items or to an entire bill.
   5. Record the data for each payment in the batch window, as shown in Figure 11–2.

   **Note:** You can submit batches that contain 20,000 payments. When a batch contains 20,000 payments, the undo buffer sometimes overflows, causing the Microsoft Foundation Class (MFC) framework to generate a warning. If you continue, the batch is submitted successfully despite the warning.

   To avoid performance issues, submit only 5,000 payments in a batch. If you receive a batch that contains more than 5,000 payments, split it into multiple batches before using Payment Tool to process it.

   **Note:** If unconfirmed payment processing is enabled, you can submit a batch of failed payments for unconfirmed payments that later fail. For more information, see "About Unconfirmed Payment Processing".

   To add more payment methods, see "Rules for Modifying Payment and Reversal Fields".
7. After each entry is complete, Payment Tool displays the updated batch total in the **Calculated Total** field. When you finish entering all the data in a batch, compare the entry in the **Calculated Total** field with the entry in the **Specified Total** box. If you expect the entries to match and they do not, check the values in the **Payment Amount** column.

---

**Note:** If Payment Suspense Manager is enabled, the **Suspended Total** box may contain an entry. This entry displays the total for all payments that cannot be submitted to the correct accounts and are therefore submitted to the payment suspense account.

---

For information about Payment Suspense Manager, see "About Payment Suspense Manager". For information about handling suspended payments, see Payment Tool Help.

8. When all the payments are recorded, click **Validate** to validate the batch of payments. If you entered a projected total when you created the batch and the calculated total of the payments in the batch does not equal the projected total, Payment Tool asks if you want to continue. You can change the payment amounts prior to submitting the batch.

   If a customer paid too much or too little, an entry might not be validated (depending on your BRM configuration). You can create a batch that includes only payments that are not valid. You can use that batch to process the payments that are not valid, usually by allocating payments. See "About Allocating Payments".

9. When all payments are validated, submit the batch to BRM.

   For information about manually suspending payments, see Payment Tool Help.

### Who uses Payment Tool?

Payment Tool is used mostly by data entry personnel. Batches of payments that cannot be validated because they require allocation are typically processed by a senior customer service representative (CSR) or an accountant.
About Allocating Payments

When customers pay more or less than they owe, you can allocate payments. For example, if a customer pays too little, you can specify which items on the bill to apply the payment to. If your batch supports bill-level allocation, you can also allocate a payment to a specific bill when there are multiple unpaid bills for an account.

When you validate data, if you see a message in the status bar that says something similar to “Payment allocation required,” you need to allocate payments.

You can also allocate an account-level payment when the payment is applied to an account with multiple bill units.

When applying payments to an account, you can allocate the payments before or after you validate them.

For the steps to allocate payments, see Payment Tool Help.

About Required and Suggested Allocations

Payment allocations can be required or suggested. If an allocation is required, you must make the payment allocation before the payment can be submitted. If an allocation is suggested, your business policy should recommend that you allocate the payment, but allocation is not required.

About Allocating Multiple Payments for the Same Bill

When a payment clerk submits a payment batch that contains multiple payments for the same bill, BRM views each payment portion as an underpayment and displays a message requiring the payment to be allocated manually.

By default, BRM views each payment as an underpayment and prompts the payment clerk to manually allocate it. To disable the underpayment validation so underpayments are not returned with an error, set the NoManualAllocation flag in the PaymentTool.ini configuration file to 1. This also disables the Allocate button in Payment Tool. When the batch is submitted to BRM, the payments are allocated correctly.

Allocating Payments to Bills and Items

You can allocate a payment to a specific bill or to individual charges (items) on a bill. A payment batch can contain either bill-level allocations or item-level allocations, but not both. You must choose the allocation level before you create a new payment batch.

If you enter a bill number and a payment amount when creating a batch, and the payment amount matches the bill amount, the bill is closed automatically when the payment is submitted. If the payment amount does not equal the amount due for the bill, your business policies should handle the underpayment or overpayment.

For example, an account has an unpaid bill of $30 and the customer pays $35. If the CSR records the payment and indicates the bill number in the payment batch, the bill is closed automatically and the overpayment is allocated to another open bill or recorded at the account level, according to your business policies.
For more information on configuring overpayment and underpayment rules, see "Handling Overpayments and Underpayments".

**Allocating an Account-Level Payment to Multiple Bill Units**

You can allocate an account-level payment to multiple bill units of an account. Payment Tool allocates the payment as per defined business policies given in the PCM_OP_PYMT_POL_MBI_DISTRIBUTE policy opcode. However, you can manually allocate the payment to override the default payment distribution.

Default payment distribution follows these rules:

- Bills having older due dates receive the payment amount first.
- If all the bills have the same due date, the bills with the higher due amounts are considered first.
- In the case of overpayment, the excess payment remains unallocated to the default bill unit of the account.
- In the case of underpayment, bills with later due dates or low due amounts do not get any payment amount.

---

**Note:** The Payment Suspense Management feature must be enabled in your BRM system for you to allocate payments to accounts with multiple bill units. For more information, see "Enabling Payment Suspense in BRM".

---

Payment Tool performs the following steps to allocate an account-level payment to multiple bill units:

1. When you validate a payment, Payment Tool calls the PCM_OP_PYMT_VALIDATE_PAYMENT opcode, which is wrapper opcode for the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode. The PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode validates that the account-level payment requires distribution to multiple bill units. This validation opcode adds the reason ID as PIN_REASON_ID_MBI_DISTRIBUTION_REQD and domain ID as PIN_PYMT_SUSPENSE_REASON_DOMAIN_ID to the PIN_FLD_PAYMENT_REASONS array in the return flist if:
   - The payment is made to an account with multiple bill units.
   - The payment is not suspended or failed.

2. After validating the payment, Payment Tool distributes the payment.
   - Payment Tool calls the PCM_OP_PYMT_MBI_DISTRIBUTE opcode to get the payment distribution across multiple bill units. PCM_OP_PYMT_MBI_DISTRIBUTE invokes the PCM_OP_PYMT_POL_MBI_DISTRIBUTE policy opcode. The PCM_OP_PYMT_POL_MBI_DISTRIBUTE policy opcode distributes the payment as per default distribution logic.
   - Payment Tool calls the PCM_OP_PYMT_SELECT_ITEMS opcode by giving the output of PCM_OP_PYMT_MBI_DISTRIBUTE (bill unit-level distribution) as input. PCM_OP_PYMT_SELECT_ITEMS identifies the item-level distribution.
   - Payment Tool calls the PCM_OP_PYMT_MBI_ITEM_SEARCH opcode to retrieve the bills/item across multiple bill units of the account.
You can override the default distribution by manually allocating the payment. After manual allocation of payment, revalidate the payment. For more information on how to manually allocate payment to multiple bill units, see Payment Tool Help.

3. After distributing the payment, Payment Tool calls the PCM_OP_PYMT_COLLECT opcode. For more information on how PCM_OP_PYMT_COLLECT allocates the payment to multiple bill units, see "Allocating Account-Level Payments to Multiple Bill Units".

---

**Note:** For manually allocated payments, the input flist contains the status as PIN_SELECT_STATUS_MBI_DISTRIBUTED. PCM_OP_PYMT_VALIDATE_PAYMENT is already called from Payment Tool. So, PCM_OP_PYMT_VALIDATE_PAYMENT does not perform previous actions again when PCM_OP_PYMT_COLLECT calls it.

---

### Allocating Payments Later

You can create a batch for only those payments that need allocations. See "Managing Non-Validated Batch Entries".

As an alternative, you can apply a payment to an account and allocate it later by using Customer Center. When you apply a payment at the account level, the account balance is reduced, but items and bills are not closed.

### Allocating Payments in More Than One Currency

Each batch accepts payments in only one currency. If your customers pay bills in an EMU national currency and the euro, be sure to create a separate batch for each currency.

---

**Important:** For countries that joined the EMU before February, 2002, the euro is the only legal currency. Countries that joined the EMU after 2/28/2002 can continue to use their national currency during the crossover period; however, the accounts should also use the euro, not the EMU currency, as the primary account currency. If you are unsure whether you should create an additional batch for the EMU currency, ask your supervisor.

---

### Allocating Externally Initiated Payments by Due Amount

When allocating an externally initiated payment, such as a payment made by check or cash, that is associated with a valid account number but not a valid bill POID, BRM uses the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode to find the appropriate bill as follows:

1. If the element associated with the payment in the PIN_FLD_CHARGES array of the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode input flist has a bills array (PIN_FLD_BILLS) and the array contains a bill number (PIN_FLD_BILL_NO), the policy opcode searches for the bill POID associated with the bill number.

2. If the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode finds the POID, it adds it to the bills array in its output flist. The information in the output flist is passed to the PCM_OP_PYMT_SELECT_ITEMS opcode by Payment Tool or by the PCM_OP_PYMT_COLLECT opcode.
3. If the payment is not associated with a valid bill number, the opcode searches for a bill whose total due amount matches the payment amount. The search is restricted to bills linked to the account with which the payment is associated.

**Note:** By default, this search is disabled. To enable it, see "Finding Bills by Due Amount".

---

**Finding Bills by Due Amount**

If the payment element in the PIN_FLD_CHARGES array does not have a bills array or if the bills array does not contain a valid bill POID or bill number, the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode searches for a bill whose total due amount matches a specified payment amount. BRM uses this search to find a bill to allocate a payment to when the bill POID and bill number associated with the payment are unknown. The search is restricted to bills that belong to the account with which the payment is associated. By default, this search is disabled.

If the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode finds the bill, it adds its POID to the bills array in its output flist. If the bills array is missing, the opcode creates the array before adding the POID.

To enable the search, you modify a field in the ar instance /config/business_params object created during the BRM installation. Use the `pin_bus_params` utility or perform this modification.

To enable this search:

1. Use the following command to create an editable XML file for the BusParamsAR parameter class:
   ```
   pin_bus_params -r BusParamsAR bus_params_AR.xml
   ```
   This command creates the XML file named `bus_params_AR.xml.out` in your working directory. If you do not want this file in your working directory, specify the full path as part of the file name.

2. Search the XML file for following line:
   ```
   <SearchBillAmount>disabled</SearchBillAmount>
   ```

3. Change `disabled` to `enabled`.

**Caution:** BRM uses the XML in this file to overwrite the existing /config/business_params object for the ar instance. If you delete or modify any other parameters in the file, these changes affect the associated aspects of BRM’s billing configuration.

4. Use the following command to load the change into the /config/business_params object:
   ```
   pin_bus_params bus_params_AR.xml
   ```
   You should execute this command from the BRM_Homesys/data/config directory, which includes support files used by the utility. To execute it from a different directory, see `pin_bus_params`.

5. Read the object with the testnap utility or Object Browser to verify that all fields are correct.

6. Stop and restart the Connection Manager (CM). For more information, see "Starting and Stopping the BRM System" in BRM System Administrator’s Guide.

7. For multiple databases, run the pin_multidb script with the -R CONFIG parameter. For more information on this script, see pin_multidb.

---

**Caution:** If an account that uses balance forward accounting receives an underpayment when this search is enabled, some bill items in the account may remain open even after funds to cover the underpayment are submitted and the account balance (amount due for all bills) returns to zero. As the open items age, they may trigger debt management programs such as BRM Collections Manager.

For example, Account A has a monthly billing cycle. At the end of May:

- Total due amount for Bill 1 is $10 (Item 1a = $5, Item 1b = $5).
- Customer makes a $5 payment.
- Item 1a is closed, Item 1b remains open, and the total due for the account is set to $5.

At the end of June:

- Total due for Bill 2 is $15 (Item 2a = $5, Item 2b = $5, and the amount due from previous bills = $5).
- Customer makes a $15 payment.
- Neither a bill POID nor a bill number was submitted with the payment, so BRM uses the payment amount to find a bill to apply the payment to. It finds Bill 2 because the total due amount for Bill 2 matches the payment amount.
- Items 2a and 2b are closed, $5 is left unallocated at the account level (it remains in the open payment item), and the total due for the account is set to $0.
- Item 1b remains open.

To close bill items such as 1b:

- Use a database reporting tool such as BRM Reports to find all the open payment items in your BRM database.
- In Customer Center, manually allocate the amount in each open payment item to the corresponding open bill item.

---

**About Reversing Payments**

Payment reversals are necessary when a payment is recorded in the BRM database, but the payment is not deposited. For example, you could record a check payment for a check that does not clear. To reopen the bill so the payment can be made again, you reverse the payment. Reversing the payment allows BRM to treat the payment as if it never happened.
You use Payment Tool to process batches of payment reversals. For more information on how BRM reverses payments, see "How BRM Reverses Payments".

---

**Note:**

- To reverse failed unconfirmed payments, create a failed payment batch.

- The reversals discussed in this section are referred to as direct reversals. They differ from indirect reversals that occur due to payment recycling during suspended payment processing. For information on reversals and how they relate to payment recycling, see "Understanding Payment Recycling".

- To reverse account-level payment made to an account having multiple bill units, you pass the original payment’s transaction ID to Payment Tool. When you submit a reversal batch, the reversal batch reverses all the sub-payments created during the account-level payment allocation to multiple bill units.

---

**Supported Payment Reversal Types**

You can directly reverse the following types of payments:

- Check
- Credit card
- Direct debit
- Inter-bank transfers
- Postal order
- Wire transfer

---

**Important:** Cash reversals enable cash payments to be recycled during payment suspense processing. They are not intended to directly reverse cash payments from the BRM database. For more information on payment reversals and recycling, see "How Payments Are Reversed".

---

**Processing a Batch of Payment Reversals by Using Payment Tool**

To process payment reversals, follow the basic procedure described in "Processing a Batch of Payments by Using Payment Tool".

- Instead of entering the bill number, you enter the payment ID. To find the payment ID, search for the account to see a list of payments.

- (Optional) Enter the original transaction date in the batch window. See "About the Columns in Batch Windows".

---

**About Externally Initiated Refunds**

To make refunds to customers with invoice accounts, first create the refund items, either manually or by using the `pin_mass_refund` utility. See "About Refunds" in BRM Managing Accounts Receivable. You then make the refund payments by check or other externally initiated payment, and record those payments by using Payment Tool. See
"Processing a Batch of Refunds by Using Payment Tool".

- You cannot refund suspended payments. For information on suspended payment processing, see "Configuring Payment Suspense Manager".
- You cannot reverse a refund. If you refund a customer account by mistake, adjust the account for the refunded amount. See "Performing Adjustments With Your Custom Application" in BRM Managing Accounts Receivable.

Supported Batch Refund Types

Payment Tool supports the following types of refunds:
- Check
- Cash
- Wire transfer
- Postal order
- Inter-bank transfer

Note: Payment Tool does not support batches of credit card refunds. BRM-initiated refunds are handled by the pin_refund utility.

Processing a Batch of Refunds by Using Payment Tool

To process refunds, you follow the same procedure described in "Processing a Batch of Payments by Using Payment Tool". The only difference is that you do not allocate payments for refunds.

Managing Refunds with Your Custom Application

For background information, see "About Refunds" in BRM Managing Accounts Receivable.

To create a refund, use the PCM_OP_BILL_ITEM_REFUND opcode. This opcode creates a refund item for a /bill or /billinfo object.

POIDs passed in to either the PIN_FLD_BILL_OBJ field or the PIN_FLD_BILLINFO_OBJ field specify the billing entity to receive the refund.

There are two ways to run PCM_OP_BILL_ITEM_REFUND: calculate-only mode or regular mode.

In calculate-only mode, PCM_OP_BILL_ITEM_REFUND:
- Computes the total refund amount without actually creating the refund item or committing any changes to the database.
- It shows the amount that a CSR can refund, based on both credit and debit items.

In the regular mode, PCM_OP_BILL_ITEM_REFUND does the following:
- Creates a refund item for the account if it has an open credit. Otherwise the refund fails and an error message is returned.
- Transfers amounts from any open credit items to the new refund item, and closes the credit items.
Transfers amounts from the /billinfo or /bill object to any open debit items. Each transfer from the new refund item to debit items decreases the value of the new refund item.

- Creates the /item/refund object, which contains the total credit amount.
- Closes these credit and debit items.
- Returns the refundable amount in the /item/refund object.

If you use a custom program to perform refunds, you can put the program name in the input flist optional field PIN_FLD_PROGRAM_NAME. If this field is used to contain the name of the program refunding a customer’s account, the program name is recorded in the events associated with an item refund. If the program name is not specified, the default value, Refund Opcode, is used.

**Note:**
- You cannot refund suspended payments. For more information, see “How Direct Reversals and Refunds Relate to Suspense”.
- You cannot reverse a refund. If you refund a customer account by mistake, adjust the account for the refunded amount. See "Performing Adjustments With Your Custom Application" in BRM Managing Accounts Receivable.

### Managing Non-Validated Batch Entries

While processing a payment, refund, or reversal batch, you might have some entries that cannot be validated easily. You can create a new batch that includes only those entries with validation errors. This allows you to submit the entries that can be validated. If Payment Suspense Manager is enabled, you can manually suspend any invalid payments and continue with submitting the batch.

To create a batch of non-validated entries, first validate the batch, then choose **Tools - Create Exception Batch**. A new batch is created that includes only non-valid batches. The non-valid batches are removed from the validated batch.

### Processing Lockbox Batches

*Lockbox processing* is a typical way to handle externally initiated payments, reversals, and refunds. With lockbox processing, the bank sends you a record of the data, which you enter into the BRM database by using Payment Tool.

Most banks that perform lockbox processing can format a text file according to your specifications, which might include:

- Which data is included.
- The format (fixed width or delimited).
- The order of the entries.
- A batch header or footer. The batch header and footer can contain information common to all payments in the payment batch, and information specific to the
batch, including the lockbox number, date, number of checks, and total payment amount. If a payment is missing information, the batch data is used.

You can have the bank deliver the file electronically, and you can use Payment Tool to import data directly from the file. See “Importing Batch Data into Payment Tool”.

---

**Note:**

- You might need to create a utility to retrieve the file.
- If the bank creates the file with the EBCDIC character set, you need to create a utility to convert it to ASCII.
- Payment Tool does not support the EDI 823 format.

---

**About the Columns in Batch Windows**

This table describes the columns in the Payment Tool batch window. Most types of batches use the same columns. Depending on the type of payment batch, and the functionality that is enabled, some columns might not be displayed. For example, the **Receipt No.** column appears only for cash payment batches, and the **Suspense Description** column appears only if Payment Suspense Manager is enabled.

The only required columns are the **Bill Number** and **Account Number** columns, but you only need one of them. For example, if you enter the account number, the bill number is not required.

You can configure custom columns for each type of batch. For example, a cash payment batch might include a column for entering the name of the person who signed the receipt.

Entering data in optional columns makes it easier for a BRM administrator to find payment information as shown in Table 11–1.

<table>
<thead>
<tr>
<th>Table 11–1 Payment Data</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Column</strong></td>
</tr>
<tr>
<td>Status</td>
</tr>
<tr>
<td>Payment ID</td>
</tr>
<tr>
<td>Allocation</td>
</tr>
<tr>
<td>*</td>
</tr>
<tr>
<td>Column</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>Bill Number</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Account Number</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Payment Transaction ID</td>
</tr>
<tr>
<td>Check Date</td>
</tr>
<tr>
<td>Receipt Date</td>
</tr>
<tr>
<td>Original transaction date</td>
</tr>
<tr>
<td>Chargeback date</td>
</tr>
<tr>
<td>Check Number</td>
</tr>
<tr>
<td>Receipt No.</td>
</tr>
<tr>
<td>Credit card No.</td>
</tr>
<tr>
<td>Bank RDFI No.</td>
</tr>
<tr>
<td>Order ID</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Wire-Transfer ID</td>
</tr>
<tr>
<td>Bank Code</td>
</tr>
<tr>
<td>Bank Account</td>
</tr>
<tr>
<td>Channel</td>
</tr>
<tr>
<td>Suspense Description</td>
</tr>
<tr>
<td>Status Description</td>
</tr>
<tr>
<td>Status Code</td>
</tr>
<tr>
<td>Comment</td>
</tr>
<tr>
<td>Payment Amount</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
You can import data from text files into Payment Tool batch format. For example, if you have electronic files of data formatted in columns, you can import that data into a batch instead of entering it manually.

To import a batch, the information must be in a text file. Payment Tool supports a wide variety of formats, including almost any delimiter character. Data can be in any order: the columns do not need to match the columns in the Payment Tool batch windows.

Data must be in a format that can be imported. Before you begin importing data, make sure you know how your data is formatted:

- When you import data, you need to specify how the data is separated in columns; for example, with spaces or tabs. Open a file containing your data to see how it is formatted.
- Your data can include information that is not formatted in columns; for example, a document heading. This information can be imported as the batch header and batch footer.
- For payment data and refund data, there are two required columns:
  - The amount paid
  - Either the account number or the bill number
- For payment reversal data, the only required column is the payment ID.

A typical input file looks like this:

```
Account Number, Payment Amount, Date, Check Number
0.0.0.1-887, 19.95, 5/11/99, 1243
0.0.0.1-425, 19.95, 5/11/99, 1543
0.0.0.1-776, 19.95, 5/11/99, 1273
0.0.0.1-143, 19.95, 5/11/99, 1254
```

Figure 11–3 shows how an input file, such as the one shown above, appears in the batch import wizard. The data is organized in columns, as defined by the delimiter character (in this case, a comma).
Applying Multiple Payments to the Same Account

Figure 11–3 Batch Import Wizard

Handling Overpayments and Underpayments by Using Payment Tool

If a customer pays too much or too little, you allocate the payment as required by your BRM business policies. For example, these are alternative underpayment policy configurations:

- Automatically apply payments to bill items.
- Suggest payment allocation in Payment Tool.
- Require payment allocation in Payment Tool.

For more information, see “Handling Overpayments and Underpayments”.

Working with Multiple Currency Types in the Payment Tool

When you process payments with Payment Tool, you choose the currency to use for each batch of payments. For information on how BRM uses currencies, see “About System and Account Currencies” in BRM Managing Customers.

Applying Multiple Payments to the Same Account

By default, to prevent duplicate entries, you cannot use Payment Tool to apply more than one payment in a batch to a single account. However, you might need to apply more than one payment if the account uses open item accounting and you need to apply payments to more than one bill.

To apply more than one payment to an account:

1. Open the Payment Tool INI file (C:\Windows\PaymentTool.ini).
2. Change this entry:

   `ALLOWACCOUNTDUP=0`

   To this:

   `ALLOWACCOUNTDUP=1`

   You do not need to exit Payment Tool; the change takes effect the next time you validate a payment.

**Manually Allocating Account-Level Payments to Accounts with Multiple Bill Units**

When you submit an account-level payment to an account with multiple bill units, BRM applies the payment across multiple bill units. You can override the default payment distribution by manually allocating the payment to multiple bill units.

By default, BRM allows you to manually allocate the payment to multiple bill units. To disable the manual payment allocation, set the `NoManualAllocation` flag in the `PaymentTool.ini` configuration file to 1. In this case, you are not allowed to manually allocate the payment, and the default BRM payment distribution applies.

**Configuring Payment Tool to Lock at the Account Level During Batch Processing**

When processing a batch of payments, Payment Tool, by default, locks all accounts associated with the batch and keeps them locked until it finishes processing the batch. This can cause problems if your batches contain a large number of payments, because other processes cannot access the accounts during payment processing.

You can configure Payment Tool to lock only the account that it is currently processing rather than the entire batch.

When processing payments in a batch, Payment Tool initiates the transaction for the entire batch and any errors that are encountered while processing any payment entry in the batch is ignored. However, when you configure Payment Tool to lock only the account it is currently processing, it records the payment item and payment event in the BRM database only when the payment entry is processed successfully. The payment is not recorded if any errors are encountered while processing the payment entry.

---

**Important:** Processing payments at the account level rather than at the batch level can produce problems if the CM or DM fails during payment processing; Payment Tool cannot determine which payments in the batch failed or were successfully processed. Thus, when you resubmit the batch for processing, Payment Tool may process a payment twice for the same account.

To find out which payments failed or were successful after a CM or DM failure, you must either check the logs or check in the database for failed `/event/billing/payment/pay_type` events with the correct batch ID. Then, you must batch only the failed payments and submit it to Payment Tool for processing.

To configure Payment Tool to lock at the account level when processing a batch of payments:
1. Open the CM configuration file (BRM_Home/sys/cm/pin.conf) in a text editor.

2. Edit the payment_batch_lock entry:
   - fm_pymt payment_batch_lock 0

   Use 0 to lock at the account level or 1 to lock at the batch level.

3. Save and close the file.

4. Stop and restart the Connection Manager (CM). For more information, see "Starting and Stopping the BRM System" in BRM System Administrator’s Guide.

Customizing Payment Details Displayed in BRM Client Tools

BRM uses /config/paymenttool objects to create payment, reversal, and refund batch type actions in the Payment Tool database. While you can handle most of your payment decisions using Payment Tool, you can also customize actions in the database. This section describes the characteristics of /config/paymenttool objects that you can customize.

When you customize /config/paymenttool, follow the rules defined in "Rules for Modifying Payment and Reversal Fields".

/config/paymenttool is defined in the init_objects.source file, which BRM reads when it starts. The init_objects.source file provides information that determines:

- What Payment Tool displays, including the columns that are displayed and whether the columns can be used for entering data or are read-only.
- What the Customer Center Payments tab displays, including the payment method and credit card numbers.

Important: You should make a backup copy of the init_objects.source file each time you modify it. When you upgrade BRM, the installation program overwrites init_objects.source, including your customizations. You can use the backup to restore your changes.

About the Default /config/paymenttool

The following is the default config/paymenttool as defined in init_objects.source.

```plaintext
! PaymentTool Payment Config object
! Mandatory fields for creation for each PIN_FLD_PAY_TYPES element specified
!
! PIN_FLD_NAME
! for each PIN_PAYMENT_TOOL_FIELDS
!   PIN_FLD_FIELD_NAME
!   PIN_FLD_COLUMN_NAME
!   PIN_FLD_PURPOSE
!   PIN_FLD_BATCH_TYPE

TYPE = /config/paymenttool
FIELDS =
array PIN_FLD_PAY_TYPES { type = PIN_FLDT_ARRAY, perms = ORW,};
field * PIN_FLD_NAME { type = PIN_FLDT_STR(255), perms = MRW,};
array * PIN_FLD_PAYMENTTOOL_FIELDS{ type = PIN_FLDT_ARRAY, perms = ORW,};
field * PIN_FLD_HIDDEN_NAME { type = PIN_FLDT_STR(255), perms = MRW,};
field * PIN_FLD_COLUMN_NAME { type = PIN_FLDT_STR(255), perms = MRW,};
field * PIN_FLD_PURPOSE { type = PIN_FLDT_INT, perms = MRW,};
```

Managing Externally Initiated Payments  11-17
Rules for Modifying Payment and Reversal Fields

All changes you make in `/config/paymenttool` are reflected in the Payment Tool graphical user interface when you stop and restart BRM. If you are changing or adding values to the payment or reversal fields there are some basic rules you need to follow:

- If you add a bill type in the `/config/paymenttool` storable class, you must add a corresponding entry in the `/config/payment` storable class.
- All user-defined fields for a particular type (data entry or display) for a given batch should be from the same array in the `/event` object.

See the `/config/paymenttool` and `/event` storable class definitions.

- If the charge opcode fields in `/config/payment` are valid, the display fields for the corresponding reversal batch should be from `/event/billing/charge/name`, where name can be defined as a credit card, direct debit, and so on.

**Tip:** The charge object contains a lot of payment information; therefore, it might be useful to display the charge object information when reversing a payment.

- There must be at least one `/config/paymenttool` object for each supported language.

You can set a language as the default by entering `Default` in PIN_FLD_NAME as shown below:

```
 0 PIN_FLD_NAME     STR [0] "PaymentTool payment Types: Default"
```

By default, the language is set to English (United States).

The Customer Center, Payment Tool, and Payment Center client applications read the `/config/paymenttool` object that has PIN_FLD_NAME set to Default.

To customize the client applications to read a specific locale of English, update PIN_FLD_NAME in the `/config/paymenttool` object as shown in the following examples:

- For English (United States):
  ```
  0 PIN_FLD_NAME     STR [0] "PaymentTool payment Types: English(United States)"
  ```

- For English (United Kingdom):
  ```
  0 PIN_FLD_NAME     STR [0] "PaymentTool payment Types: English(United Kingdom)"
  ```

You can also use the PCM_OP_WRITE_FLDS opcode to set the locale.

**Note:** The country name you specify should exactly be the same as the country name in the language parameter for that particular locale.
In the /config/paymenttool definition, an array element must always begin with PIN_FLD_PAYMENTTOOL_FIELDS. Each array corresponds to an array element in the /config/payment object. The sequence of array elements determines the order of columns displayed in Payment Tool.

A payment batch must have only data-entry fields. A reversal batch can have display as well as data-entry fields.

All the fields you use to enter or display data in Payment Tool for a particular batch grid must be defined in the same array in the object definition.

Storable objects for payment reversal are created as subclasses from /event/billing/payment or /event/billing/reversal.

Creating an Object Definition for a New Payment or Reversal Event

To create a new payment or reversal event, start a new PIN_FLD_PAYMENTTOOL_FIELDS array. Use the following example to define your custom fields. Before you define an object, see "Rules for Modifying Payment and Reversal Fields".

PIN_FLD_PAY_TYPES is the first line. The 0 to the left of this line indicates that this is the beginning of a sequence of arrays.

0  PIN_FLD_PAY_TYPE ARRAY [0] allocated 3, used 3

PIN_FLD_NAME indicates that the array has been allocated 5 items and 3 have been used. This line is at the same level as the following PIN_FLD_PAYMENTTOOL_FIELDS items.

1  PIN_FLD_NAME ARRAY [10003] allocated 5, used 3

10003 comes from a corresponding entry in /config/payment/.

PIN_FLD_PAYMENTTOOL_FIELDS begins the definition of an individual, user-defined array. The 1 to the left of the line shows that this is a subset of PIN_FLD_PAYMENTTOOL_FIELDS but of equal status to PIN_FLD_NAME.

1  PIN_FLD_PAYMENTTOOL_FIELDS ARRAY [0] allocated 3, used 3

An example of individual arrays is shown below.

Within the first PIN_FLD_BATCH_TYPE array the numeral 1, not [1], says that this is a reversal batch type while 0 is a payment method. The second PIN_FLD_BATCH_TYPE in this example is also a reversal batch type. Remember that while a payment batch only has data entry fields, a reversal batch can have display and data entry fields.

The first PIN_FLD_PURPOSE, with a value of 1 indicates that this field is read-only. The second PIN_FLD_PURPOSE value is 0, indicating that data can be entered in this field. In other words, you cannot enter information in "Credit Card No.", but you can enter a value for "Chargeback Date". PIN_FLD_FIELD_NAME is the database field name, not the column name (PIN_FLD_COLUMN_NAME).

The values in brackets, [0], [1], and [2] are index values dealing with the sequence of fields.
Customizing Payment Details Displayed in BRM Client Tools

Changing the Order of Columns in Payment Tool

The sequence of fields, PIN_FLD_PAYMENTTOOL_FIELDS, determines the order of columns in Payment Tool. If you are not satisfied with the default settings and you want to add another column of information or change a column name, you need to customize Payment Tool. To do this, you need to change the /event/billing/payment, /event/billing/reversal, and /config/paymenttool objects.

To change the order of the columns, you need to change the order of PIN_FLD_PAYMENTTOOL_FIELDS arrays in /config/paymenttool because the column order is determined by the order in which they appear in this object.

In the following example, the three configurable, user-defined columns are in the order:

- Credit Card No.
- Chargeback Date
- Reason Code

```
0 PIN_FLD_PAY_TYPES ARRAY [10003] allocated 5, used 5
1 PIN_FLD_NAME STR [0] "Credit Card"
1 PIN_FLD_PAYMENTTOOL_FIELDS ARRAY [0] allocated 3, used 3
2 PIN_FLD_BATCH_TYPE INT [0] 1
2 PIN_FLD_COLUMN_NAME STR [0] "Credit Card No."
2 PIN_FLD_FIELD_NAME STR [0] "PIN_FLD_DEBIT_NUM"
2 PIN_FLD_PURPOSE INT [0] 1
1 PIN_FLD_PAYMENTTOOL_FIELDS ARRAY [1] allocated 3, used 3
2 PIN_FLD_BATCH_TYPE INT [0] 1
2 PIN_FLD_COLUMN_NAME STR [0] "Chargeback Date"
2 PIN_FLD_FIELD_NAME STR [0] "PIN_FLD_EFFECTIVE_T"
2 PIN_FLD_PURPOSE INT [0] 0
1 PIN_FLD_PAYMENTTOOL_FIELDS ARRAY [2] allocated 3, used 3
2 PIN_FLD_BATCH_TYPE INT [0] 1
2 PIN_FLD_COLUMN_NAME STR [0] "Reason Code"
2 PIN_FLD_FIELD_NAME STR [0] "PIN_FLD_REASON_CODE"
2 PIN_FLD_PURPOSE INT [0] 0
```

To change the order of the columns in Payment Tool, you only need to change the order of each array. In the following example, the columns are in the order:

- Credit Card No.
- Reason Code
- Chargeback Date

```
0 PIN_FLD_PAY_TYPES ARRAY [10003] allocated 5, used 5
1 PIN_FLD_NAME STR [0] "Credit Card"
1 PIN_FLD_PAYMENTTOOL_FIELDS ARRAY [0] allocated 3, used 3
2 PIN_FLD_BATCH_TYPE INT [0] 1
2 PIN_FLD_COLUMN_NAME STR [0] "Credit Card No."
2 PIN_FLD_FIELD_NAME STR [0] "PIN_FLD_DEBIT_NUM"
2 PIN_FLD_PURPOSE INT [0] 1
1 PIN_FLD_PAYMENTTOOL_FIELDS ARRAY [1] allocated 3, used 3
2 PIN_FLD_BATCH_TYPE INT [0] 1
2 PIN_FLD_COLUMN_NAME STR [0] "Reason Code"
2 PIN_FLD_FIELD_NAME STR [0] "PIN_FLD_REASON_CODE"
2 PIN_FLD_PURPOSE INT [0] 0
```

```
1 PIN_FLD_PAYMENTTOOL_FIELDS ARRAY [2] allocated 3, used 3
2 PIN_FLD_BATCH_TYPE INT [0] 1
2 PIN_FLD_COLUMN_NAME STR [0] "Reason Code"
2 PIN_FLD_FIELD_NAME STR [0] "PIN_FLD_REASON_CODE"
2 PIN_FLD_PURPOSE INT [0] 0
```
Adding a New Column to Payment Tool

To add a new column to Payment Tool, you add a new column section to an array in `/config/paymenttool`. The new section contains information needed for that column.

The following example shows the information for a **Customer Complaint** column:

```
1 PIN_FLD_PAYMENTTOOL_FIELDS ARRAY [0] allocated 3, used 3
2 PIN_FLD_BATCH_TYPE INT [0] 1
2 PIN_FLD_COLUMN_NAME STR [0] "Customer Complaint"
2 PIN_FLD_FIELD_NAME STR [0] "PIN_FLD_COMPLAINT"
2 PIN_FLD_PURPOSE INT [0] 9
```

Adding Direct Debit Details to the Customer Center Payments Tab

By default, the Customer Center **Payments** tab does not display details from payment vendors on whether direct debit payments were authorized. To have the direct debit details added to the **Payments** tab, add the direct debit fields to the PIN_FLD_PAY_TYPES array of the `/config/paymenttool` class.

1. Use the PCM_OP_WRITE_FLDS opcode to add the direct debit vendor details to the `/config/paymenttool` class. Call the opcode using flag 32. For example:

```
0 PIN_FLD_POID POID [0] 0.0.0.1 /config/paymenttool 10574 0
0 PIN_FLD_OP_CORRELATION_ID STR [0] "1:MCHELLAM-idc:UnknownProgramName:0:AWT-EventQueue-0:5:1226568418:0"
0 PIN_FLD_PAY_TYPES ARRAY [10005] allocated 2, used 2
1 PIN_FLD_NAME STR [0] 'Direct Debit'
1 PIN_FLD_PAYMENTTOOL_FIELDS ARRAY [4] allocated 4, used 4
2 PIN_FLD_BATCH_TYPE INT [0] 0
2 PIN_FLD_COLUMN_NAME STR [0] "Authorization Result"
2 PIN_FLD_FIELD_NAME STR [0] "PIN_FLD_RESULT"
2 PIN_FLD_PURPOSE INT [0] 9
0 PIN_FLD_POID POID [0] 0.0.0.1 /config/paymenttool 10574 0
0 PIN_FLD_OP_CORRELATION_ID STR [0] "1:MCHELLAM-idc:UnknownProgramName:0:AWT-EventQueue-0:5:1226568418:0"
0 PIN_FLD_PAY_TYPES ARRAY [10005] allocated 2, used 2
1 PIN_FLD_NAME STR [0] 'Direct Debit'
1 PIN_FLD_PAYMENTTOOL_FIELDS ARRAY [5] allocated 4, used 4
2 PIN_FLD_BATCH_TYPE INT [0] 0
2 PIN_FLD_COLUMN_NAME STR [0] "Authorization Code"
2 PIN_FLD_FIELD_NAME STR [0] "PIN_FLD_AUTH_CODE"
2 PIN_FLD_PURPOSE INT [0] 9
0 PIN_FLD_POID POID [0] 0.0.0.1 /config/paymenttool 10574 0
0 PIN_FLD_OP_CORRELATION_ID STR [0] "1:MCHELLAM-idc:UnknownProgramName:0:AWT-EventQueue-0:5:1226568418:0"
0 PIN_FLD_PAY_TYPES ARRAY [10005] allocated 2, used 2
1 PIN_FLD_NAME STR [0] 'Direct Debit'
1 PIN_FLD_PAYMENTTOOL_FIELDS ARRAY [6] allocated 4, used 4
2 PIN_FLD_BATCH_TYPE INT [0] 0
2 PIN_FLD_COLUMN_NAME STR [0] "Vendor Results"
2 PIN_FLD_FIELD_NAME STR [0] "PIN_FLD_VENDOR_RESULTS"
2 PIN_FLD_PURPOSE INT [0] 9
0 PIN_FLD_POID POID [0] 0.0.0.1 /config/paymenttool 10574 0
```
Customizing Payment Details Displayed in BRM Client Tools

0 PIN_FLD_OP_CORRELATION_ID STR [0]
"1:MCHELLAM-idc:UnknownProgramName:0:AWT-EventQueue=0:5:1226568418:0"
0 PIN_FLD_PAY_TYPES ARRAY [10005] allocated 2, used 2
1 PIN_FLD_NAME STR [0] "Direct Debit"
1 PIN_FLD_PAYMENTTOOL_FIELDS ARRAY [7] allocated 4, used 4 PIN_FLD_BATCH_TYPE INT [0] 0
2 PIN_FLD_COLUMN_NAME STR [0] "Authorization Date"
2 PIN_FLD_FIELD_NAME STR [0] "PIN_FLD_AUTH_DATE"
2 PIN_FLD_PURPOSE INT [0] 9

2. Stop and restart the CM. See "Starting and Stopping the BRM System" in BRM System Administrator’s Guide.

---

Customizing the Date Format of Batch Files in Payment Tool

By default, when processing both imported and manually created batch files, Payment Tool uses the date format of the system locale of the client on which it is running. For example, if the locale is United States English, Payment Tool uses MM/dd/yyyy format. If the locale is New Zealand English, Payment Tool uses dd/MM/yyyy format.

To configure Payment Tool to use a different date format from the one used by your client system:

1. Open the Payment Tool INI file (C:\Windows\PaymentTool.ini).

2. In the file, find this entry:

   DefaultDateFormat=

3. Set the entry to the appropriate date format.

Possible values:

- MM[sc]dd[sc]yyyy
- dd[sc]MM[sc]yyyy

Where [sc] is the date separator character, such as slash (/) or dot (.), specified in the regional settings of the Windows client.

---

Important: The values are case sensitive.

For example:

DefaultDateFormat=dd/MM/yyyy

If no date format is specified, Payment Tool uses the default format of the system locale.

You do not need to exit Payment Tool after updating the INI file; the change takes effect the next time you create or import a payment batch.
Managing Suspended Payments

This chapter provides an overview of how Oracle Communications Billing and Revenue Management (BRM) handles payment suspense operations performed using Payment Center. For information on how to use Payment Center, see Payment Center Help.

About Payment Center

You use Payment Center to apply suspended payments to customer accounts and to suspend payments that have been posted incorrectly. You can process only one suspended payment at a time, and you need either a valid account number or bill number to associate with the suspended payment.

- **Important:** Payment Center does not support branded databases.

- **Note:** Payment Center is packaged in the PaymentTool.zip file and shares the same installation. To install Payment Center, download and install Payment Tool. To uninstall Payment Center, uninstall Payment Tool.

You can perform the following tasks by using Payment Center:
- Search for accounts, bills, and payments.
- Apply suspended payments to customer accounts.
- Divide one suspended payment into multiple distributed payments, and apply each one to a different customer account.
- Allocate suspended payments to bills and bill items in one or more accounts.
- Suspend payments that have been posted in customer accounts.
- Remove unallocatable payments from the payment suspense account.

For more information, see Payment Center Help.

How BRM Processes Suspended Payments

Payment Center calls the following opcodes to process suspended payments:
- PCM_OP_PYMT_VALIDATE_PAYMENT to perform the validation.
About Searching for Payments

You use Payment Center to search for suspended payments and payments that have been posted in customer accounts. Depending on how the payment analyst specifies the search, BRM looks for the following types of payments:

- **Suspended payments**: This search finds payments that entered BRM as suspended as well as payments that were made to customer accounts and later suspended. Payment analysts typically use this type of search when they want to find suspended payments to distribute to customer accounts.

- **All payments**: This search finds suspended payments and active payments made to customer accounts. Payment analysts can use this type of search if they are also looking for payments that have already been allocated to customer accounts but that need to be placed in suspense instead. You can narrow this type of search to include only payments that have a specific payment type such as cash or check.

After it has the list of payments, Payment Center retrieves information on the state of any suspended payments returned by the search. It does so by performing a second search, this time for payments that have never been reversed. This search finds states for all payments that were generated from a suspended payment, such as distributed payments, but it does not retrieve payment states for payments that are no longer active.

The types of payments returned in these searches include:

- Partially distributed suspended payments.

- Fully distributed suspended payments (only if the payment analyst specifies Include Fully Allocated Payment).

- Suspended payments that were removed from the suspense account as unallocatable.

- Payments that were reversed using Payment Tool.

About Payment Center Validation

Before a distribution list from Payment Center is submitted to BRM, it is validated. You can choose how to validate payments that you submit through Payment Center. Payment Center can either validate payments:

- **Automatically**: When you enter information, the data is validated as soon as you move your focus to the next field. If errors are detected, you must correct them before continuing.

If a payment is an overpayment or underpayment, Payment Center notifies you and allows you to change the allocated amount. During allocation, you will also be...
notified if you underallocate a payment. You are prevented from overallocating payments.

Field-level validation is the default for Payment Center. If you do not want to use field-level validation, you can turn it off and validate the information manually after entering any amount of data.

- **Manually**: After you enter any amount of data, you click **Validate**. Payment Center then uses the **Status** column in the Allocate Payment window to display the validation result of each allocation. It also indicates whether the allocation was an overpayment or an underpayment so you can correct the payment amount.

Although allocation is on a per-account basis, validation occurs for the entire transaction; if one of the distributed payments fails, the entire distribution list will fail. Therefore, when performing a distributed payment allocation, it is better to use the default (field-level validation) so you do not encounter problems after entering a large amount of payment data.

Payment Center populates certain lists depending on the validation level you choose. If field-level validation is enabled during account-level allocation, the open bills in an account are automatically listed in the user interface as you enter information. If field-level validation is disabled, you must manually enter each bill number for a particular account. The open bills for an account are not populated in the **Bill Number** column.

If one or more payments fail validation, the suspended payment is returned to its pre-allocation state. In effect, the entire result of the payment distribution list is undone. You must correct the problems and the list must pass validation before you can submit the distributed payments. For information about payment validation, see the discussion about managing payments in the BRM documentation. For information on how BRM validates suspended payments, see the discussion about payment process implementation and customization in the BRM documentation.

If you determine that one or more distributed payments from the same list were submitted incorrectly, you can resuspend them at any time. And, if you do not know how to fully allocate a suspended payment to a distribution list, you can leave an amount in suspense. You can continue allocating the payment at any time.

### About Allocating Suspended Payments

You allocate suspended payments from the Payment Allocation window.

You can allocate a payment based on the account number only, the bill number only, or either the account number or bill number, depending on the information available in the payment. The allocation method you choose configures the columns in the Payment Allocation window. This makes entering payment information easier, the column for the chosen method is enabled and the column for the method that is not chosen is disabled. For example:

- If you choose **Account Number**, the bill number field is disabled so that you can tab directly from the **Account Number** column to the **Allocate** column across each row.
- If you choose **Bill Number**, the account number field is disabled, so that you can tab directly from the **Bill Number** column to the **Allocate** column across each row.
- If you choose **Account Number or Bill Number**, both columns are enabled. If you enter information in only one of the columns, you must tab through the other column before you can enter the amount in the **Allocate** column.
About Allocating Suspended Payments

About Allocating Suspended Payments

If you apply a payment to specific bills or items in an account, BRM retrieves all open bills and all open bill items. The results are displayed in the allocation window so you can assign an amount to each account, bill, or item. If you perform an item-level allocation after having entered a bill number for the payment, the item list will only contain open items for that bill. If you perform an item-level allocation with the account number entered only, the item list will contain all open items for that account.

About Deferred Payment Allocation

By default, when a payment is submitted to BRM, the BRM business policies either allocate the payment automatically or post it to the account without being allocated to specific bills or bill items. Unallocated payments remain as a credit on the account and can be allocated later by using Customer Center.

For an account with multiple bill units, when an account-level payment is submitted with deferred allocation flag set, the payment is distributed to multiple bill units but remains unallocated.

Payment Center provides both options when you submit payments by using the account number only. If you choose to have BRM allocate the payment automatically and the allocated amount is less than the amount due, you will have the option of manually allocating the payment to specific bills and items.

You can restrict Payment Center from providing the defer allocation option by setting the Allocation Window preferences in Payment Center. For instructions on how to allocate payments and set Payment Center preferences, see Payment Center Help. See "About Payments" for more information on how BRM allocates payments.

About Overallocations and Underallocations

Overallocation occurs when the sum of distributed payment amounts is greater than the amount of the suspended payment. Underallocation occurs when the sum of distributed payment amounts is less than the amount of the suspended payment.

Overallocation is not valid in BRM, but you can configure BRM to handle underallocations. For example, if an underallocation occurs, BRM can cancel the entire allocation operation or leave the remaining amount in suspense.

In Payment Center, if you overallocate a suspended payment to a bill or item in an account, you are required to change the allocation amount to be less than or equal to the amount due on the bill or item, respectively. If you underallocate a suspended payment, you are warned that the payment has been underallocated and are prompted to fix the allocation. For example, if a suspended payment is $100, and you divide it into three $25 distributed payments, the result is a $25 underallocation.

If this occurs, you can use Payment Center to correct the payment allocation any of the following ways:

- Cancel the entire payment allocation and leave the suspended payment in its previous state.
- Modify the total amount that was allocated in each distributed payment.
- Add an account to the payment distribution list, and allocate the remaining amount to that account.
About Allocating Suspended Payments

- Leave the remaining amount in the suspended payment.

For more information on allocating payments, see Payment Center Help.

About Allocating Suspended Payments to Multiple Bill Units

If an account has multiple bill units, you can allocate a suspended payment to specific bill units of that account.

After a suspended payment is validated for an account having multiple bill units, it can have any of the following statuses:

- Validated (with MBI Distribution): UnderPayment: If the payment-allocated amount is less than the actual bill amount.
- Validated (with MBI Distribution): OverPayment: If the payment-allocated amount is more than the actual bill amount.
- Validated (with MBI Distribution): Exact Match: If the payment-allocated amount is the same as the actual bill amount.

You can allocate a payment to multiple bill units only if:

- The suspended payment is allocated by specifying the account number.
- The validation status is Validated (with MBI Distribution): UnderPayment.

Payment Center displays the default payment distribution to multiple bill units. The default payment distribution is defined in your business policy. You can manually allocate the payment that overrides the default distribution. After manual allocation, you need to revalidate the payment.

- For a parent account having multiple bill units, you can allocate the payment to different bill units of the parent account and its child accounts.
- For a non-paying child account having multiple bill units, you can allocate the payment to different bill units of the child account.
- For accounts with multiple bill units, you can allocate the suspended payment that has been partially allocated to multiple bill units of that account.

For more information on how to allocate account-level payments to multiple bill units, see Payment Center Help.

Working with Overpayments and Underpayments

As you enter data in Payment Center to allocate a suspended payment to one or more accounts, you are warned if a row contains an underpayment or an overpayment. If there is an overpayment, you can:

- Apply the overpayment to the account, leaving the excess amount unallocated in the account.
- Change the distributed payment amount to equal the account balance.
- Cancel the entire distributed payment allocation, leaving the suspended payment in its original state.

If there is an underpayment, you can:

- Change the distributed payment amount to equal the account balance.
- Accept the underpayment, and continue with the allocation.
Cancel the entire distributed payment allocation, leaving the suspended payment in its original state.

If an underpayment occurs during item-level allocation, you can either accept the item allocation list suggested by BRM or reject the list and manually select which items receive the payment.

When you submit the payment batch, BRM processes the payments and identifies the items to which the payment is applied. If you allowed the underpayment or overpayment in Payment Center, BRM allocates overpayments and underpayments of funds according to the payment analyst’s instructions.

If the payment amount equals the amount due, BRM closes each bill and bill item. Payment amounts that do not equal the amount due for the bill are underpayments or overpayments, and your business policies should be set up to handle both these conditions. For example, a $35 payment is received for an unpaid $30 bill. When the payment analyst enters the payment and indicates the bill number, the bill is closed automatically and the $5 overpayment is either allocated to another open bill or is recorded at the account level, according to your business policies.

---

**Note:** Depending on your business policies, overpayments may require manual allocation.

---

For information on how to customize BRM to handle overpayments and underpayments, see the discussion about payment processing implementation and customization in the BRM documentation.

### Configuring Payment Center for Custom Payment Methods

If you create custom payment methods for your BRM system, you must customize Payment Center to handle them. This overview procedure describes how to create custom classes and fields and enable Payment Center to handle them.

---

**Note:** Before customizing your payment functionality, you should be familiar with the Java PCM and the BRM Storable Class Editor, which you use to create custom classes and fields. For background information on creating custom classes and fields, see “Creating Client Applications by Using Java PCM” and “Creating Custom Fields and Storable Classes” in BRM Developer’s Guide. For information on the Storable Class Editor, see Storable Class Editor Help.

---

1. Complete the following tasks by using Storable Class Editor:

   a. Create your storable classes and fields in the Java PCM package.

   b. Create source files for your custom fields.

   **Important:** Storable Class Editor creates a C header file called cust_flds.h, a Java properties file called InfranetPropertiesAdditions.properties, and a Java source file for each custom field.

2. In the directory where Storable Class Editor created the Java source files, compile the source files:
javac -d . *.java

3. Package the new class files into a JAR file. For example:
   jar cvf customfields.jar *.class

4. Copy the contents of the InfranetPropertiesAdditions.properties file and paste it into the Payment Center Infranet.properties file. By default, this file is located in the following directory.
   C:\Program Files\Portal Software\PaymentCtr\PaymentCenter

5. Append the location of the JAR file to the PAYCTRCP environment variable path. For example:
   ;;C:\Program Files\Portal Software\PaymentCtr\customfields.jar;

---

Customizing the Date Format for Payment Center

You can customize the format of the date displayed in the Payment Search dialog box, the Undo Allocation dialog box, and the Payment Results screen in Payment Center.

To customize the date format for Payment Center:

1. Open the PaymentCenter_Home/paymentcenter.properties file in a text editor, where PaymentCenter_Home is the directory in which you installed Payment Center.

   \[\text{Note: } \text{If the paymentcenter.properties file does not exist, you must create it manually.}\]

2. Add the following entry:
   DefaultDateFormat=format

   where format is one of the following:
   - dd/MM/yyyy
   - dd/MMM/yyyy
   - dd.MMMM.yyyy
     where MMMM is the spelled-out name of the month (for example, September.)
   - yyyy/dd/MM
   - MMM/dd/yyyy
   - MM/dd/yyyy

   The default is MM/dd/yyyy.

   For example, if you set DefaultDateFormat=dd/MM/yyyy, Payment Center displays June 30, 2012 as 30/06/2012.

3. Save and close the file.
Improved Performance of Searches for Payment Events in Payment Center

By default, Payment Center retrieves five payment events for each step of a search. You can improve Payment Center’s performance of payment event searches by configuring the `paymentsearch.stepsize` entry in the `paymentcenter.properties` configuration file.

To configure the step search size:

1. Open the `Payment_Center_Home/paymentcenter.properties` file in a text editor, where `Payment_Center_Home` is the directory in which Payment Center is installed.

2. Set the `paymentsearch.stepsize` entry to a value based on the number of events in your system and your client/server memory configuration. For example:

   `paymentsearch.stepsize=100`

3. Save the file.
This chapter describes how to resolve failed credit card and direct debit transactions in Oracle Communications Billing and Revenue Management (BRM) for Paymentech.

For information about the utilities used for resolving BRM-initiated payment transactions, see "pin_clean" and "pin_recover".

About Failed BRM-Initiated Payment Transactions

Failed credit card or direct debit payment transactions occur when BRM connects to a credit card processor, sends a transaction, but doesn’t get confirmation from the credit card processor that the transaction was completed. This is usually caused by a connection loss.

BRM identifies failed transactions by keeping a record of each transaction in the BRM database. If BRM does not get confirmation that the clearing house (Paymentech) received the transaction successfully, checkpoint records are left in the database. To resolve failed transactions, you need to resolve all checkpoint records. Transactions usually have a checkpoint record for the following reasons:

- A transaction batch was received by the credit card processor, but it wasn’t processed completely. To resolve this error, you need to resubmit the transaction batch.
- A transaction was processed by the credit card processor, but the connection was lost before BRM received the results. To resolve this error, you need to update the checkpoints in the BRM database.

**Note:** If the payment processor is offline or too busy to handle your transactions, BRM records a no-answer (1) record. You don’t need to resolve no-answer records.

BRM includes two utilities that you use to resolve failed BRM-initiated payment transactions, `pin_recover` and `pin_clean`. To resolve a failed BRM-initiated payment transaction, you first run the `pin_clean` utility to see if there are any errors. If there are, the method you use for resolving the error depends on the type of transaction. For example, you follow a different procedure for resolving a failed verification than you do for resolving a failed authorization. See Table 13–1, "Types of Failed Credit Card Transactions".
How BRM Records Transactions

Before BRM connects to the credit card processor, a table row with the value 999 is inserted in the database. This value remains in the row until BRM processes the result from the Paymentech credit card processor. BRM first sets the result field to the number 1000, plus the Paymentech result code. When BRM begins processing the payment, it resets the result value to the Paymentech result code. If the transactions are completed successfully: regardless of whether the credit card charge was successful: the result column will not have any values over 999.

The following Paymentech result codes are used by BRM:

- PASS = 0
- FAIL_NO_ANS = 1
- FAIL_ADDR_AVS = 2
- FAIL_ADDR_LOC = 3
- FAIL_ADDR_ZIP = 4
- FAIL_CARD_BAD = 5
- SRVC_UNAVAIL = 6
- FAIL_DECL_SOFT = 7
- FAIL_DECL_HARD = 8
- FAIL_NO_MIN = 9
- INVALID_CMD = 10
- FAIL_SELECT_ITEMS = 11
- CVV_BAD = 12
- NO_CREDIT_BALANCE = 13
- FAIL_LOGICAL_PROBLEM = 14
- FAIL_FORMAT_ERROR = 15
- FAIL_INVALID_CONTENT = 16
- FAIL_TECHNICAL_PROBLEM = 17
- AUTH_PENDING = 888
- DEPOSIT_PENDING = 777
- CHECKPOINT = 999
- OFFSET = 1000

Failed credit card transactions (checkpoint value of 999) are not collected by pin_collect or PCM_OP_BILL_COLLECT to avoid double charges. Such results indicate a communication problem between Paymentech and BRM.

Result values of 1000+, indicate that an exception occurred within BRM. This means that the 999 checkpoint has been cleared; however payment events were not created in BRM. See "Checkpoints are Cleared but Payment Events Are Not Created" to fix these transaction errors.
Checking for Transaction Errors

You should check for transaction errors every day. The best way to do this is to create a script that runs the `pin_clean` utility and reports transaction failures.

**Tip:** The `pin_clean` utility writes output to `stdout`, so you can write a script to run the `pin_clean` utility daily and write the output to a file.

The following procedure shows how to run the `pin_clean` utility manually.

1. Run the `pin_clean` utility with the `summary` option:
   ```
   pin_clean -summary
   ```
   The `pin_clean` utility is in `BRM_Home/bin`.

   **Tip:** If there are a lot of checkpoint records, use the `-search_count_limit` option to limit the number of records found.

   ```
   pin_clean -summary -search_count_limit n
   ```

2. Review the results. The following example contains six failures: 1 verification failure, 3 authorization failures, and 2 refund failures.

   CheckPoint Log Summary:
   ```
   PIN_CHARGE_CMD_VERIFY 1
   PIN_CHARGE_CMD_AUTH_ONLY 3
   PIN_CHARGE_CMD_CONDITION 0
   PIN_CHARGE_CMD_DEPOSIT 0
   PIN_CHARGE_CMD_REFUND 2
   ```

3. Determine how to resolve the transaction, as described in Table 13–1.

<table>
<thead>
<tr>
<th>Record type</th>
<th>Error</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>verify</td>
<td>The connection was lost during an online transaction such as registration.</td>
<td>Delete the transaction record from the BRM database. You don’t need to resubmit it. See &quot;Deleting Failed Verifications&quot;.</td>
</tr>
<tr>
<td>authorize</td>
<td>The connection was lost during an online transaction such as registration, or a one-time charge to a single account.</td>
<td>Delete the transaction record from the BRM database. If necessary, repeat the transaction, for example, use Customer Center to charge the account again. Since the transaction was for an authorization, but not for a payment, the customer can’t be charged twice. See &quot;Resolving Authorizations&quot;.</td>
</tr>
<tr>
<td>conditional deposit</td>
<td>The connection was lost while running the <code>pin_collect</code> utility.</td>
<td>See &quot;Resolving Transactions by Using a Request for Response (RFR) File&quot;.</td>
</tr>
</tbody>
</table>
Deleting Failed Verifications

Since no charge was authorized or made during a verification, you can delete all failed verification transactions. There is no need to resubmit a verification.

1. Run the `pin_clean` utility without the `summary` option to display unresolved transaction records:
   ```bash
   pin_clean
   ```
   The `pin_clean` utility is in `BRM_Home/bin`.
   A summary of transaction errors appears, followed by a choice of commands.

2. To display the transactions for verifications, press 1.
   When you display a type of transaction, a list of batches appears, followed by a list of commands.

3. To delete all verification transaction records, press 2.

   **Note:** You should delete records with a value greater than 999 when you want to recharge an account by using `pin_collect`. (The `pin_clean` utility only processes payments with checkpoint records = 999.) This deletes the `/event/billing/charge` object and the appropriate rows in the `EVENT_T`, `EVENT_BILLING_CHARGE_T`, and `EVENT_BILLING_CHARGE_CC_T` tables.

4. To quit, press 3.

Resolving Authorizations

To resolve failed authorizations, you can delete all transactions, but you need to find out which transactions need to be resubmitted. For example, if a CSR issues a debit in Customer Center, and that transaction failed, you need to reissue the debit after deleting the checkpoint record.

1. Run the `pin_clean` utility without the `summary` option to display unresolved transaction records:
   ```bash
   pin_clean
   ```
   The `pin_clean` utility is in `BRM_Home/bin`.
   A summary of transaction errors appears, followed by a choice of commands.

2. To display the transactions for authorizations, press 2.
   A list of transactions appears, followed by a list of commands.

3. To display a single transaction record, press 1.
4. Enter the number of the transaction.
   The event details appear.

5. Read the event details to find out if this is an event you want to repeat, for example:
   - The event description. The following example is for credit card transactions.
     
     0 PIN_FLD_SYS_DESCR STR [0] "Authorization"
   
   - The account number:
     
     0 PIN_FLD_ACCOUNT_OBJ POID [0] 0.0.0.1 /account 28456 0
   
   - The amount:
     
     0 PIN_FLD_AMOUNT NUM [0] 100.000000
   
   - The date and time:
     
     0 PIN_FLD_CREATED_T TSTAMP [0] (827435459) Thu Mar 21 11:10:59 1996

Make a note of the amount and the account number so that you can repeat the transaction later.

6. To delete the checkpoint, press 1.

7. Continue displaying and deleting checkpoints until all authorization checkpoints are deleted.

8. To return to the summary screen, press 2.


10. Repeat the transactions that you recorded.

---

**Resolving Refunds**

To resolve failed refunds, you can delete all transactions, but you need to find out which transactions need to be resubmitted.

If there is a checkpoint, there is no original refund. You can resubmit the refund.

1. Run the `pin_clean` utility without the `summary` option to display unresolved transaction records:

   ```
   pin_clean
   ```

   The `pin_clean` utility is in `BRM_Home/bin`.

   A summary of transaction errors appears, followed by a choice of commands.

2. To display the transactions for refunds, press 5.

3. To display a single transaction record, press 1.

4. Enter the number of the transaction.

   The event details appear.

5. Read the event details to find out if this is an event you want to repeat, for example:
   - The event description:
Resolving Transactions by Using a Request for Response (RFR) File

Make a note of the amount, date, and the account number so that you can repeat the action, if necessary.

6. To delete the checkpoint, press 1.
7. Continue displaying and deleting checkpoints until all refund checkpoints are deleted.
8. To return to the summary screen, press 2.
10. Find out if the refund was made. Use the Event Browser to search for the refund.
11. If the refund was made, you’re finished. If the refund was not made, issue the refund again.

Resolving Transactions by Using a Request for Response (RFR) File

To resolve failed pin_collect and pin_deposit batches, you should always first use the pin_recover utility with the rfr option.

---

Important: You cannot use the rfr option if the transaction was an online transaction such as a charge or refund made by using Customer Center. See "Resolving Authorizations" and "Resolving Refunds".

---

1. Request an RFR file from the Paymentech customer support service. If there is no RFR file, you can’t complete this procedure. See "Resubmitting Transactions".

---

Note: When you request an RFR file, Paymentech does not send you the file. Instead, it posts it so that the pin_recover utility can access it at Paymentech.

---

2. Run the pin_recover utility with the rfr option:

```
pin_recover -rfr
```

The pin_recover utility is in BRM_Home/bin.

3. After the pin_recover utility has finished, run it again by using the rfr option. Paymentech sometimes posts multiple RFR files, and you need to process all of them before continuing.
4. Run the pin_clean utility in summary mode again. If you still find transaction errors, refer to "Resubmitting Transactions".

Resubmitting Transactions

**Note:** Regardless of the number of times you run the pin_recover utility with the rfr option, accounts are charged only once for each transaction.

---

If running the pin_recover utility with the rfr option doesn’t resolve all transactions, run the pin_recover utility with the resubmit option to resubmit the same batch and process the transactions that didn’t go through.

**Caution:** If you use a payment processor other than Paymentech, make sure that it uses duplicate transaction detection. If not, using the pin_recover utility with the resubmit option can cause customers to be billed twice. The duplicate transaction detection offered by Paymentech eliminates this problem.

---

**Important:** Resubmit your transactions promptly, or the authorizations might need to be redone. VISA authorizations, for example, last only seven days.

---

1. To run the pin_recover utility with the resubmit option, you need to find the batch ID for the batch you are resubmitting. To do so, run the pin_clean utility in summary mode again:

   `pin_clean -summary`

The pin_clean utility is in BRM_Home/bin.

A summary of transaction errors appears, followed by a choice of commands, for example:

```
CheckPoint Log Summary:
PIN_CHARGE_CMD_VERIFY 1
PIN_CHARGE_CMD_AUTH_ONLY 3
PIN_CHARGE_CMD_CONDITION 1
PIN_CHARGE_CMD_DEPOSIT 1
PIN_CHARGE_CMD_REFUND 2
```

Choose function by number:
1) View PIN_CHARGE_CMD_VERIFY
2) View PIN_CHARGE_CMD_AUTH_ONLY
3) View PIN_CHARGE_CMD_CONDITION
4) View PIN_CHARGE_CMD_DEPOSIT
5) View PIN_CHARGE_CMD_REFUND
6) Delete All
7) Done

2. Do one of the following:
   - Press 3 to display transactions made by running the pin_collect utility.
Press 4 to display transactions made by running the `pin_deposit` utility.

A list of batches appears.

3. Make a note of the batch ID that you want to resubmit, for example T,2f.

---

Note: When resubmitting deposits, each transaction has two transaction IDs, one for the original authorization, and one for the deposit batch sent by the `pin_deposit` utility. Use the batch ID that was used by the `pin_deposit` utility.

---

4. Press 3 to quit the `pin_clean` utility.

5. Run the `pin_recover` utility with the `resubmit` option to resubmit the unprocessed transactions. The `pin_recover` utility is in `BRM_Home/bin`.

   ```
   pin_recover -resubmit batch_ID
   ```

   For example:

   ```
   pin_recover -resubmit T,2f
   ```

6. Run the `pin_clean` utility in summary mode again. If you still find transaction errors, refer to "Deleting Transactions".

---

Checking for Transactions in Paymentech Send Files

If there are still checkpoint records after using the `pin_recover` utility with the `rfr` and `resubmit` options, you can search the Paymentech send files to find out if the transaction was sent to Paymentech, located by default in `/fusa_temp`. (You define the location of the send files in the `temp_dir` Paymentech DM configuration file entry.)

There will probably be multiple files. Find the file that matches the date of the transaction. Open the file in a text editor and search for the batch ID that was reported by the `pin_clean` utility. If the batch ID is not present in any file, the connection was broken between the CM and the DM, and the transaction was never sent.

If the transaction is a deposit, you should search the database to find outstanding deposits. To do so, use the `testnap` utility to search for authorization records with no matching deposit record. See "Testing Your Applications and Custom Modules" in `BRM Developer’s Guide`.

If the transaction is a payment, see "Resolving Payments".

---

Resolving Payments

In rare cases, a credit card charge is made and the checkpoint record is cleared, but the `/event/billing/payment` object is not recorded in the BRM database. This might happen because of a network or hardware failure.

To find charge events in your database that have no matching payment events, use the `testnap` utility. See "Testing Your Applications and Custom Modules" in `BRM Developer’s Guide`.

If you are missing payment events, use the `pin_recover` utility with the `recover_payment` option. Since the charge has already been made, this option has no effect on the customer’s credit card.

   ```
   pin_recover -recover_payment
   ```
This creates the payment item (if necessary) and payment event objects.

---

**Note:** To create the objects, rows are inserted into the EVENT_T and EVENT_BILLING PAYMENT_T database tables. If the payment item does not exist for the bill, a row is also inserted into the ITEM_T database table. If possible, the money is allocated to open items; however, you may need to manually allocate the payment.

---

When a payment recovery was successful, the Event Browser displays the message, “Payment - Thank you” and the EVENT_BILLING_PAYMENT_CC_T value = 0.

### Deleting Transactions

**Important:** Deleting transactions is typically only used for failed verification and authorization transactions. Always use the **pin_recover** utility first to resolve transactions submitted by the **pin_collect** and **pin_deposit** utilities. Only delete and resubmit **pin_collect** and **pin_deposit** transactions as a last resort.

1. Run the **pin_clean** utility without the **summary** option to display any remaining unresolved transaction records:

   ```
   pin_clean
   ```

   The **pin_clean** utility is in **BRM_Home/bin**.

   A summary of transaction errors appears, followed by a choice of commands.

2. Press the key that corresponds to the command you want to carry out. For example, press 3 to display transactions made by running the **pin_collect** utility.

   A list of batches appears, followed by a list of commands.

   **CheckPoint Log Summary:**

   ```
   PIN_CHARGE_CMD_VERIFY 1
   PIN_CHARGE_CMD_AUTH_ONLY 3
   PIN_CHARGE_CMD_CONDITION 1
   PIN_CHARGE_CMD_DEPOSIT 1
   PIN_CHARGE_CMD_REFUND 2
   ```

   **Choose function by number:**

   1) View PIN_CHARGE_CMD_VERIFY
   2) View PIN_CHARGE_CMD_AUTH_ONLY
   3) View PIN_CHARGE_CMD_CONDITION
   4) View PIN_CHARGE_CMD_DEPOSIT
   5) View PIN_CHARGE_CMD_REFUND
   6) Delete All
   7) Done
Troubleshooting Unresolvable Credit Card Transactions

**Important:** Do not delete condition or deposit records until you know that the corresponding charge was successful. Do not delete records that are less than seven days old.

If you delete checkpoints for failed condition or deposit records that are created within the duplicate detection period, customers might be charged twice, because BRM resubmits these records.

The length of time for charges to be processed depends on the payment processor. Check with your payment processor.

3. Make a note of the batch IDs that you need to delete. You’ll need them when you resubmit the batches.

4. Press the key that corresponds to the command you want to carry out, for example, press 2 to delete all transaction records.
   
   When there is more than one record per transaction type, you can display each record and delete it, or you can delete all records of that type without displaying them.

5. When you have deleted all of the transactions that need to be resubmitted, quit the utility.

6. Use the `pin_recover` utility to resubmit the batch.
   ```
   pin_recover -resubmit T,2a
   ```
   
   The `pin_recover` utility is in `BRM_Home/bin`.

**Troubleshooting Unresolvable Credit Card Transactions**

This section lists problems you might encounter while trying to resolve failed credit card transactions and provides information on how to fix them.

**Unable to remove checkpoints when using an RFR file**

If checkpoints still exist after running the `pin_recover` utility, resubmit the batch. See "Resubmitting Transactions" for more information.

---

**Note:** Paymentech has duplicate transaction detection, which prevents a customer from being charged twice.

---

If resubmitting the batch doesn’t clear the checkpoints, do the following:

1. Delete the transactions.

2. Run the `pin_recover` utility with the `resubmit` option.

3. Run the `pin_clean` utility with the `summary` option to select and delete batches. Make sure to note the batch ID.

4. Run the `pin_ecover` utility with the `-resubmit` option and provide the batch ID.

**Checkpoints are Cleared but Payment Events Are Not Created**

Look in the database for checkpoints with a value of 1000. If they exist, run the `pin_recover` utility with the `recover_payment` option.
This creates the payment item (if necessary) and payment event objects.

Note: The pin_clean utility does not show charges that have checkpoint values greater than 999.

Important: You should only use this option when a credit card number is reported as charged in both BRM and Paymentech, but it has not been recorded as paid in BRM. This is an uncommon scenario that can occur when the network connection is dropped after Paymentech responds and before BRM allocates the payment.

When a payment recovery was successful, the Event Browser displays the message, “Payment - Thank you” and the EVENT_BILLING_PAYMENT_CC_T value = 0.

Paymentech does not have an RFR file and never received the payment batch

If you requested an RFR file from Paymentech and one does not exist, run the pin_recover utility with the -resubmit option and provide the batch ID. See "Resubmitting Transactions" for more information.

If Paymentech confirms they received the batch but checkpoints still exist, request an RFR file and run the pin_recover utility with the rfr option.
This chapter describes the Oracle Communications Billing and Revenue Management (BRM) Resource Reservation Manager and explains how to use its components to reserve resources.

---

**Important:** Resource Reservation Manager is an optional component that requires a separate license.

---

Before using Resource Reservation Manager, you should be familiar with the following:

- BRM concepts and architecture. See "Introducing BRM" and "BRM System Architecture" in *BRM Concepts*.

### About Resource Reservation Manager

Resource Reservation Manager allows you to set aside a portion of a customer’s resources for a prepaid session. It allows you to reserve resources for single-RUM and multi-RUM sessions. This prevents customers from applying those resources to other services while the session is in progress.

It also allows customers to use multiple prepaid services concurrently and charge the usage against a single account balance. For example, it allows customers to make a wireless phone call while downloading an MP3 file.

You can use Resource Reservation Manager to implement the following functionality:

- Create resource reservations. See "About Creating Reservations" for more information.
- Extend the amount that is reserved for an existing reservation. See "About Extending a Resource Reservation Amount" for more information.
- Extend the validity period for an existing reservation. See "About Extending a Resource Reservation Expiration Time" for more information.
- Release reservations. See "About Releasing a Partially Used Reservation" for more information.

Resource Reservation Manager is a framework, consisting of opcodes and storable classes, that allows you to quickly support reservations for prepaid services.
About Creating Reservations

Resource Reservation Manager creates reservation objects in one of the following:

- The BRM database.

The object specifies how much resource has been reserved, when the reservation expires, and the service type to which the reservation request applies.

Resource Reservation Manager performs these tasks when creating a reservation:
1. Determines whether to store the reservation in IMDB Cache or in the BRM database. See "About Storing Reservations in IMDB Cache" and "About Storing Reservations in the BRM Database".
2. Calculates the amount of resources that are required to use the service.
3. Calculates the account’s available resources.
4. Applies rules to determine whether the account contains sufficient resources.
5. If there are sufficient resources, creates the reservation object.

See "Creating Reservations" for more information.

About Storing Reservations in IMDB Cache

In IMDB Cache, BRM stores reservations in these transient objects:

- `/reservation/active`. This object stores information about a single reservation. BRM creates one such object for each balance group that is impacted by a prepaid session. For example, if a session impacts balance groups A and B, BRM creates one object for balance group A and one object for balance group B.
- `/reservation_list`. This object tracks reservations for a single balance group. It stores a list of current `/reservation/active` objects as well as the total resources currently reserved in `/reservation/active` objects, separated by resource type. For example, it stores that a balance group is currently reserving $10 and 30 minutes of free usage. BRM creates one object for each balance group.

When a prepaid session ends, BRM deletes or releases the `/reservation/active` object, decrements the RESERVED_AMOUNT field in the `/reservation_list` object, and returns unused resources back to the customer’s prepaid account balance.

About Storing Reservations in the BRM Database

In the BRM database, reservations are stored in these objects:

- `/reservation`. This object stores information about a reservation for single-RUM and multi-RUM quantities. BRM creates one such object for each balance group that is impacted by the prepaid session.
- `/balance_group`. The RESERVED_AMOUNT field in this object stores the total resources the balance group currently has reserved in `/reservation` objects.

When a session ends, BRM deletes or releases the `/reservation` object, decrements the RESERVED_AMOUNT field in the `/balance_group` object, and returns any unused resources back to the account.
Setting the Type of Resource Reserved

You can reserve both currency and non-currency resources. If multiple resource types such as dollars and free minutes are specified, the order in which resources are used must be specified in the price list. To use one resource type such as free minutes first, give that resource a higher priority. You do this by creating multiple rates in your price plan and assigning a priority to the rate. For more information, see "Common Price List Solutions" in BRM Setting Up Pricing and Rating.

Setting an Expiration Time for the Reservation Request

You can limit how long a reservation is valid by setting an expiration time. The length of time you reserve a resource can have important effects on your customers’ usage and your business. For example, if the expiration time is too long, resources remain unavailable to other services. If the expiration time is too short, the reservation might expire before the customer has a chance to use the resources.

To specify an expiration time, set the PIN_FLD_EXPIRATION_T field in the input flist to PCM_OP_RESERVE_CREATE.

If the field isn't passed in, the expiration is set to a default time of 24 hours. However, you can change the default expiration time by customizing the PCM_OP_RESERVE_POL_PREP_CREATE policy opcode.

Setting the Expiration Time for Prepaid Services

If your customers prepay for services, it is possible to lose revenue when all of the following are true:

- You implement a custom application to return unused reserved resources to the account. See "About Releasing an Unused Reservation" for more information.
- A customer starts a session before the reservation expires but ends the session after the reservation expiration time.
- The same customer starts a session for a second service before the first service session ends but after the resource for the first session is returned to the account balance.

The customer’s account balance is not debited for a session until the session ends. Therefore, if a resource reservation expires while an initial session is in progress, the entire reserved amount is returned to the account balance. If a customer starts another service session, all of the account balance can be reserved for the second session, even though the initial session might still be using those resources.

Figure 14–1 shows how a revenue loss occurs when a customer ends an initial session after the reservation time has expired:
In the above example, revenue is also lost if the customer does not use service2 and the reserved resource for service2 is returned to the account balance: Resource Reservation Manager debits the account $4.00 for service1, because that’s all that is available, resulting in a loss of $4.00. When the account is credited for unused service2 resources, the lost $4.00 is not recovered, even though there is now enough resource to cover that amount.

To help prevent revenue loss, you can extend the resource reservation expiration time. See "About Extending a Resource Reservation Expiration Time" for more information.

### About Extending a Resource Reservation Expiration Time

You might want to extend the reservation time if a customer has not used all of the reserved amount before the reservation expiration time.
To extend the reservation expiration time, you use the PCM_OP_RESERVE_RENEW opcode. This opcode’s input flist includes the POID of the reservation object to extend and the length of time in seconds to add to the current reservation expiration time. For example, if the reservation object expires in 30 minutes, and you extend the reservation by 15 minutes, the total reservation time will be for 45 minutes.

**About Releasing a Partially Used Reservation**

When a reservation is released, any unused resources are credited back to the account balance by the PCM_OP_RESERVE_RELEASE opcode. This occurs at the end of a session after final rating is completed. If a session is terminated unexpectedly, this opcode also releases and returns the reservation amount to the account balance.

If a reservation expires while a session is still in progress, the reserved resource for that session is also returned to the account. This might result in lost revenue if your customers have prepaid services. See "Setting the Expiration Time for Prepaid Services" for more information.

**About Releasing an Unused Reservation**

If resources are reserved for a session but the session doesn’t start before the resource reservation expires, the resource is locked and is not returned to the account balance. Therefore, if you want resources from unused reservations returned to the account balance, you must write a script to do this.

For example, create a script that searches for /reservation and /reservation/active objects with an expiration time (PIN_FLD_EXPIRATION_T field) that is less than the current time and that has a reservation status (in the PIN_FLD_RESERVATION_STATUS field) of PIN_RESERVATION_RESERVED (0). Then return the resources from those objects back to the appropriate accounts by using the PCM_OP_RESERVE_RELEASE opcode.

You should run this script as often as possible to ensure that resources are always available. The script can be stored and run from any directory. If you use PCM_OP_RESERVE_RELEASE to return resources, your script needs to include the location of the Connection Manager (CM).

**Tip:** Using the cron command is a good way to run your script. For details, see your UNIX documentation.

**About Reserving and Releasing Disputed Amounts**

You can reserve resources whenever an account disputes a bill item. This prevents your customers from misusing resources during the course of the dispute. Once a dispute is settled, you can release the resources back to the account. For more information, see "Customizing Item-Level Disputes" and "Customizing Item-level Settlements" in BRM Managing Accounts Receivable.

You use the following policy opcodes to manage reservations for disputes:

- To reserve a disputed amount, use the PCM_OP_RESERVE_POL_POST_DISPUTE policy opcode. See "Customizing Resource Reservation for Disputes" in BRM Managing Accounts Receivable.

- To release a reservation as part of the settlement process, use the PCM_OP_RESERVE_POL_POST_SETTLEMENT policy opcode. See "Customizing Resource Reservation for Settlements" in BRM Managing Accounts Receivable.
Sending Reservation Requests to the Resource Reservation Manager Opcodes

To manage reservations for prepaid services, your system must be designed to collect the information needed for creating or updating the relevant objects and pass the appropriate fields in the input flist to the resource reservation manager opcodes.

There are two types of reservation requests that can be passed on to the resource reservation manager opcodes: amount-based and quantity-based. Resource reservation manager reserves resources for amount-based requests without going through rating and discounting. For example, if a customer requests 15 minutes of usage time for a session and the customer’s account balance is greater than 15 minutes, then 15 minutes are reserved for the session. Otherwise, the reservation request is denied.

Resource reservation manager reserves resources for quantity-based requests based on the results of rating and discounting. For example, if a customer requests 45 minutes for a download session, BRM rates this request to determine the cost for the session. If the cost is $0.10 per minute, the requested session will cost $4.50. If the customer’s account balance allows it, then $4.50 is reserved for the requested session. Otherwise, the request is denied (if the minimum quantity requested is 45 minutes) or a portion of the requested quantity is reserved depending on the customer’s account balance.

You can use the resource reservation manager API to perform the following:

- Create a reservation. See "Creating Reservations" for more information.
- Associate a reservation with a session. See "Associating a Session with a Reservation" for more information.
- Extend the reservation amount. See "Extending the Reservation Amount" for more information.
- Find a reservation. See "Finding a Reservation" for more information.
- Return unused resources back to the customer’s account balance. See "Releasing Reservations" for more information.
- Extend the expiration time for a reservation. See "Extending the Expiration Time for a Reservation" for more information.

Creating Reservations

Use the PCM_OP_RESERVE_CREATE opcode to create a /reservation or /reservation/active object.

---

**Note:** The /reservation object is not created for zero balance impact events or free events.

---

To create a reservation, this opcode performs the following actions:

1. Determines whether to create a /reservation or /reservation/active object by reading the balance_coordinator entry in the CM pin.conf file:
   - If balance_coordinator is set to 0, the opcode creates a /reservation/active object. This configuration is used for IMDB Cache-enabled prepaid systems.
   - If balance_coordinator is set to 1, the opcode creates a /reservation object. This setting is used for prepaid systems that use the BRM database for AAA and for non-prepaid systems.
2. Verifies whether an amount-based or a quantity-based field is passed in the input flist. If both are passed, the opcode generates an error.

3. Calls the PCM_OP_BAL_GET_BALANCES opcode to retrieve the `/balance_group` object POID.

4. For prepaid requests, calls the PCM_OP_BAL_GET_PREPAID_BALANCES opcode to retrieve the balance group’s `/reservation_list` object and current reservation balance.

5. Calls the PCM_OP_RESERVE_POL_PREP_CREATE policy opcode to perform any custom validations. See “Customizing Resource Reservation Rules” for more information.

6. Determines whether it is an amount-based or a quantity-based request by checking the PIN_FLD_BALANCES array in the input flist:
   - If the array is present, it’s an amount-based request. The opcode determines whether the account’s available resources are greater than the requested amount and, if they are, proceeds to the next step. Otherwise, the reservation request is denied.
   - If the array is not present, it’s a quantity-based request. The opcode calls the PCM_OP_ACT_USAGE opcode in CALC_ONLY mode to calculate whether the account has resources to cover the requested quantity (single-RUM) or quantities (multi-RUM) based on the rating parameters that are passed in the PIN_FLD_EVENT substruct.

   For information on how BRM applies rating and discounting to reserve the requested resources, see "How BRM Rates and Records Usage Events" in BRM Setting Up Pricing and Rating.

   PCM_OP_ACT_USAGE returns the reservation amount and the list of balance groups impacted by the event. If the account’s available resources are greater than the reservation request, it proceeds to the next step. Otherwise, the reservation request is denied.

7. Creates a reservation object for each balance group impacted by the event.

8. Updates the customer’s reservation balance:
   - For `/reservation/active` objects, updates each balance group’s `/reservation_list` object.
   - For `/reservation` objects, updates the reserved amount in each balance group’s PIN_FLD_RESERVED_AMOUNT field.

9. Returns the following, depending on the success of the action:
   - If it is successful, returns the amount reserved and sets PIN_FLD_RESERVATION_ACTION to PIN_FLD_RESERVATION_SUCCESS. It also returns reservation information for any sponsoring accounts in the PIN_FLD_RESERVATION_LIST array.
   - If it fails, returns PIN_FLD_RESERVATION_ACTION set to PIN_FLD_RESERVATION_FAIL.

### Associating a Session with a Reservation

Use the PCM_OP_RESERVE_ASSOCIATE opcode to associate a session with a reservation.
This opcode is called at the start of a session. When a session starts, a session POID is created. This opcode copies the session POID to the reservation object, thereby associating the session with the reservation object. This opcode takes as input the POID of the reservation object and the POID of the session object.

This opcode returns the POID of the reservation object.

**Extending the Reservation Amount**

Use the PCM_OP_RESERVE_EXTEND opcode to extend an existing amount for a resource reservation. See “About Extending a Resource Reservation Amount” for more information.

To extend a reservation amount, this opcode performs the following actions:

1. Verifies whether an amount-based or a quantity-based field is passed in the input flist. If both are passed, the opcode generates an error.

2. Calls PCM_OP_BAL_GET_BALANCES to retrieve the *balance_group* object POID.

3. Determines whether to extend */reservation* or */reservation/active* objects by reading the *balance_coordinator* entry in the CM *pin.conf* file:
   - If *balance_coordinator* is set to 0, the opcode extends */reservation/active* objects. Use this configuration for IMDB Cache-enabled prepaid systems.
   - If *balance_coordinator* is set to 1, the opcode extends */reservation* objects. Use this configuration for non-prepaid systems and for prepaid systems that use the BRM database for AAA.

4. For prepaid requests, calls PCM_OP_BAL_GET_PREPAID_BALANCES to retrieve the balance group’s */reservation_list* object and current reservation balance.

5. Calls the PCM_OP_RESERVE_POL_PREP_EXTEND policy opcode to perform any custom validations.

6. Determines whether this is a quantity-based or an amount-based request by checking the PIN_FLD_BALANCES array in the input flist.
   - If the array is present, it’s an amount-based request. The opcode determines whether the requested amount is an incremental amount or an aggregated amount by reading the PIN_FLD_RESERVATION_MODE input flist field.
     - If the field is set to PIN_RESERVE_INCREMENTAL_AMOUNT, the opcode adds the amount passed in the input flist to the existing reservation amount.
     - If the field is set to PIN_RESERVE_AGGREGATED_AMOUNT, the opcode uses the amount passed in the input flist.
   - If the array is not present, it’s a quantity-based request. The opcode calls PCM_OP_ACT_USAGE in CALC_ONLY mode to calculate whether the account has resources to extend the requested quantity (single-RUM) or quantities (multi-RUM) based on the rating parameters that are passed in the PIN_FLD_EVENT substruct.

For information on how BRM applies rating and discounting to extend the requested resources, see “How BRM Rates and Records Usage Events” in BRM Setting Up Pricing and Rating.

PCM_OP_ACT_USAGE returns the extension amount and the list of balance groups that are impacted.
7. Determines whether PIN_FLD_FLAGS is set to ignore previous reservations; if yes, ignores previous reservations. For example, if the current reservation is $20, and the extension amount is $10, if you ignore the previous reservation, the extended reservation amount is set to $10. If you do not ignore the previous reservation, the reservation amount is set to $30.

8. Calculates the account’s available resources.

9. Determines whether the account’s available resources are greater than the reservation request and, if they are, proceeds to the next step.

10. Updates the reservation object for each balance group impacted by the event.

11. Updates the account’s reservation balance:
   - For /reservation objects, updates the PIN_FLD_RESERVED_AMOUNT field in each /balance_group object.
   - For /reservation/active objects, updates the PIN_FLD_BALANCES field in each /reservation_list object.

12. Returns the reservation amount and sets PIN_FLD_RESERVATION_ACTION to either PIN_RESERVATION_SUCCESS to indicate that the extension succeeded or to PIN_RESERVATION_FAILURE to indicate that the extension failed.

**Finding a Reservation**

Use the PCM_OP_RESERVE_FIND_OBJ opcode to find one or more /reservation or /reservation/active objects.

This opcode takes as input:
- A routing POID from PIN_FLD_POID indicating the database to search.
- The reservation number from PIN_FLD_RESERVATION_NO.
- One or more of the following:
  - The account object POID
  - The session POID
  - The service object POID
  - The balance group POID
- An optional reservation status field, PIN_FLD_RESERVATION_STATUS. If this field is included, PCM_OP_RESERVE_FIND_OBJ will search reservation objects with the status specified. If this field is not included, PCM_OP_RESERVE_FIND_OBJ searches reservation objects with a status of PIN_RESERVATION_RESERVED.

When PCM_OP_RESERVE_FIND_OBJ finds an object successfully, it returns the POID of the matching /reservation or /reservation/active object.

When the PCM_OPFLG_READ_RESULT flag is included in the call to PCM_OP_RESERVE_FIND_OBJ, PCM_OP_RESERVE_FIND_OBJ also returns the following details about the object:
- Quantity (single-RUM) or quantities (multi-RUM) reserved
- POID of the account, service, balance group, and session objects
- Expiration time
- Status
- Reservation number
All amounts that have been reserved for this reservation object
If not successful, check the flist spec to determine the error.

Releasing Reservations

Use the PCM_OP_RESERVE_RELEASE opcode to release one or more reservations. This opcode returns unused resources to the account so they can be used later.

Note: If a reservation object is not specified, this opcode searches for and releases all expired reservation objects.

See "About Releasing a Partially Used Reservation" or "About Releasing an Unused Reservation" for more information.

PCM_OP_RESERVERELEASE is called when a session is terminated by a stop accounting event. This opcode takes as input the POIDs of the /account and /reservation_list objects. Additional input may include an expiration time to search for the expired reservation object, the reservation object array to be released, and the reservation object POID.

To release /reservation or /reservation/active objects, this opcode performs the following actions:

1. Retrieves the /reservation and /reservation/active objects by using the specified /account and /reservation_list POIDs. If the /reservation_list POID is not passed in, this opcode finds all unexpired reservation objects that are associated with the /account object.

2. Calls the PCM_OP_RESERVE_POL_PRE_RELEASE policy opcode to perform custom validation.

3. Updates the reservation balance:
   - For /reservation/active objects, updates the /reservation_list object by removing the POIDs of the /reservation/active objects and by decrementing the total reserved amount in its PIN_FLD_BALANCES field.
   - For /reservation objects, updates the PIN_FLD_RESERVED_AMOUNT fields in their associated /balance_group objects.

4. Determines whether to delete or release the reservation objects by checking the PIN_FLD_DELETED_FLAG field in the input flist:
   - If the flag is set to True, deletes the reservation objects.
   - If the flag is set to False or if it is missing, releases the reservation objects.

5. Returns the POIDs of the /account object and the released /reservation and /reservation/active objects.

Extending the Expiration Time for a Reservation

Use the PCM_OP_RESERVE_RENEW opcode to extend the expiration time for an existing resource reservation. See "About Extending a Resource Reservation Expiration Time" for more information.

This opcode takes as input:

- The /reservation or /reservation/active object from PIN_FLD_POID.
The amount of time, in seconds, to add to the expiration time from PIN_FLD_EXPIRATION_T.

To extend the expiration time for a reservation, this opcode performs the following actions:

1. Checks the reservation status.
   - If the reservation is expired or no longer active, this opcode returns an error.
   - If the reservation is active, this opcode proceeds to the next step.

2. Extends the reservation expiration time by the amount specified in PIN_FLD_EXPIRATION_T.

3. If it is successful, returns the POID of the /reservation or /reservation/active object.

PCM_OP_RESERVE_RENEW fails when the reservation has already expired or is no longer active.

Customizing Resource Reservation

Use the following policy opcodes to customize the resource reservation rules:

- To specify the rules to qualify for a resource reservation, use the PCM_OP_RESERVE_POL_PREP_CREATE policy opcode. See "Customizing Resource Reservation Rules" for more information.

- To specify the rules to qualify for a resource reservation extension, use the PCM_OP_RESERVE_POL_PREP_EXTEND policy opcode. See "Customizing the Rules for Extending a Reservation" for more information.

- To specify the rules for releasing a resource reservation, use the PCM_OP_RESERVE_POL_PRE_RELEASE policy opcode. See "Customizing the Rules for Releasing a Reservation" for more information.

Customizing Resource Reservation Rules

Use the PCM_OP_RESERVE_POL_PREP_CREATE policy opcode to specify the rules that permit a resource reservation. By default, this policy opcode generates a unique reservation ID if one isn’t passed in the input flist. The reservation ID uses this format: hostname#threadid#systemtime (systemtime is in milliseconds).

However, you can customize this policy opcode to include custom resource reservation rules. For example, you can:

- Reserve whatever resource is available, even if the available resource is less than the requested amount.

- Check for duplicate reservation requests.

This policy opcode is called by the PCM_OP_RESERVE_CREATE opcode before a /reservation or /reservation/active object is created.

Customizing the Rules for Extending a Reservation

Use the PCM_OP_RESERVE_POL_PREP_EXTEND policy opcode to specify the rules for extending a resource reservation amount. By default, this policy opcode does nothing, but you can customize it to include custom extension rules.

This policy opcode is called by PCM_OP_RESERVE.Extend before the reservation amount is extended.
See "Extending the Reservation Amount" for more information.

**Customizing the Rules for Releasing a Reservation**

Use the PCM_OP_RESERVE_POL_PRE_RELEASE policy opcode to perform custom actions before releasing a /reservation or /reservation/active object. By default, this policy opcode does nothing, but you can customize it to release unused resources back to the account or to perform other custom actions.

This policy opcode is called by the PCM_OP_RESERVE_RELEASE opcode before it releases a reservation object.

**Installing Resource Reservation Manager**

Before installing Resource Reservation Manager, you should be familiar with BRM concepts and architecture. See "Introducing BRM" and "BRM System Architecture" in BRM Concepts.

**System Requirements**

Resource Reservation Manager is available for the HP-UX IA64, Linux, Solaris, and AIX operating systems. For information on disk space requirements for the HP-UX IA64, Solaris, AIX, and Linux operating systems, see Disk space requirements.

**Software Requirements**

This section describes the software that must be installed before you install Resource Reservation Manager.

Before installing Resource Reservation Manager you must install:

- Third-Party software, which includes the PERL libraries and JRE required for installing BRM components. See "Installing the Third-Party Software" in BRM Upgrade Guide.

- BRM. For information, see "Putting Together Your BRM System" in BRM Installation Guide.

- Oracle 9i or Oracle 10g or Oracle 11g.

**Installing Resource Reservation Manager**

**Note:** If you have already installed the product, features that are already installed cannot be reinstalled without uninstalling them first. To reinstall a feature, uninstall it and then install it again.

To install Resource Reservation Manager:

1. Download the software to a temporary directory (temp_dir).
Important:

- If you download to a Windows workstation, use FTP to copy the .bin file to a temporary directory on your UNIX server.
- You must increase the heap size used by the Java Virtual Machine (JVM) before running the installation program to avoid “Out of Memory” error messages in the log file. For information, see “Increasing Heap Size to Avoid “Out of Memory” Error Messages” in BRM Installation Guide.

2. Go to the directory where you installed the Third-Party package and source the `source.me` file.

```
Bash shell:
source source.me.sh

C shell:
source source.me.csh
```

Caution: You must source the `source.me` file to proceed with installation, otherwise “suitable JVM not found” and other error messages appear.

3. Go to the temp_dir directory and enter this command:

```
7.4_ResourceResMgr_platform_32_opt.bin
```

Note: You can use the -console parameter to run the installation in command-line mode. To enable a graphical user interface (GUI) installation, install a GUI application such as X Windows and set the DISPLAY environment variable before you install the software.

4. Follow the instructions displayed during installation. The default installation directory for Resource Reservation Manager is `opt/portal/7.4`.

Note: The installation program does not prompt you for the installation directory if BRM or Resource Reservation Manager is already installed on the machine and automatically installs the package at the BRM_Home location.

5. Go to the directory where you installed the Resource Reservation Manager package and source the `source.me` file:

```
Bash shell:
source source.me.sh

C shell:
source source.me.csh
```

6. Go to the BRM_Home/setup directory and run the `pin_setup` script.
Uninstalling Resource Reservation Manager

To uninstall Resource Reservation Manager, run the BRM_Home/uninstaller/ResourceResMgr/uninstaller.bin.

Note: The pin_setup script starts all required BRM processes.

Your Resource Reservation Manager installation is now complete.
This chapter provides reference information for Oracle Communications Billing and Revenue Management (BRM) payment utilities.
load_pin_ach

Use this utility to load the merchant information for all credit card processor and automated clearing house (ACH) vendors into the /config/ach object in the Oracle Communications Billing and Revenue Management (BRM) database. You define the payment processor and merchant information in the `BRM_Home/sys/data/pricing/example/pin_ach` file.

For information about configuring merchants and payment processors, see "Setting Up Merchants and Payment Processors".

---

**Note:** You cannot load separate /config/ach objects for each brand. All brands use the same object.

---

**Caution:** When you run the `load_pin_ach` utility, it replaces the existing payment processor and merchant information. If you are updating a set of processors and merchants, you cannot load new information only. You load complete sets of information each time you run the `load_pin_ach` utility.

---

**Important:** To connect to the BRM database, the `load_pin_ach` utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in BRM System Administrator’s Guide.

---

**Location**

`BRM_Home/bin`

**Syntax**

```
load_pin_ach [-d] [-v] [-t] pin_ach_file
```

**Parameters**

- `-d`
  Creates a log file for debugging purposes. Use this parameter for debugging when the utility appears to have run with no errors, but the ACH merchant data has not been loaded into the database.

- `-v`
  Displays information about successful or failed processing as the utility runs.

- `-t`
  Runs a test to check the input to the utility. It verifies that the input file exists and validates the information contained in the input file.

  **pin_ach_file**
  The name and location of the file that defines payment processor merchants. The default `pin_ach` file is in `BRM_Home/sys/data/pricing/example`.
If you copy the pin_ach file to the same directory from which you run the load_pin_ach utility, you don’t have to specify either the path or the file name.

If you run the command in a different directory from where the pin_ach file is located, you must include the entire path for the file.

Results

The load_pin_ach utility notifies you when it successfully creates the /config/ach storable object.

If the load_pin_ach utility doesn’t notify you that it was successful, look in the utility log file (default.pinlog) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.
pin_balance_transfer

Use this utility to perform all automatic sponsored top-ups that are scheduled to occur within the time range specified in the utility’s command line parameters.

For information about automatic sponsored top-ups, see the following topics:

- About Sponsored Top-ups
- Performing Automatic Sponsored Top-ups

**Important:** To connect to the Oracle Communications Billing and Revenue Management (BRM) database, this utility needs a configuration file in the directory from which you run the utility. For information about creating configuration files for BRM utilities, see "Creating Configuration Files for BRM Utilities" in *BRM System Administrator’s Guide*.

---

**Location**

`BRM_Home/bin`

**Syntax**

```
pin_balance_transfer [-verbose] [-test] [-start mm/dd/yy] [-end mm/dd/yy]
```

**Parameters**

**-verbose**

Displays information about successful or failed processing as the utility runs.

**Note:** This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter `-verbose` at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace `filename.log` with the name of the log file:

```
pin_balance_transfer other_parameter -verbose > filename.log
```

**-test**

Tests the utility, but does not affect accounts. Use this parameter to see which accounts will receive automatic sponsored top-ups without actually transferring funds from group owner accounts to member accounts.

**-start mm/dd/yy or yyyy**

Start date. For information on using this parameter, see "Specifying Start and End Times" in *BRM Configuring and Running Billing*.

**-end mm/dd/yy or yyyy**

End date. For information on using this parameter, see "Specifying Start and End Times" in *BRM Configuring and Running Billing*.
Results

This utility notifies you only if it encounters errors. Look in the `default.pinlog` file for errors. This file is either in the directory from which the utility was started or in a directory specified in the utility configuration file.
Use this utility to find all unresolved credit card and direct debit payments recorded in the Oracle Communications Billing and Revenue Management (BRM) database. For more information about unresolved payment transactions, see "Resolving Failed BRM-Initiated Payment Transactions".

**Note:** To connect to the Oracle Communications Billing and Revenue Management (BRM) database, the pin_clean utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in BRM System Administrator’s Guide.

**Important:** For multidatabase systems, you must run the utility separately against each database in your system. See "Running Non-MTA Utilities on Multidatabase Systems" in BRM System Administrator’s Guide.

### Location

`BRM_Home/bin`

### Syntax

```
pin_clean [-summary] [-search_count_limit n] [-auth_pending]
            [-verbose] [-help]
```

### Parameters

**-summary**
Displays the total number of each type of unresolved credit card transactions. In this example, there are three verification errors, three authorization errors, and two refund errors. For more information about these errors, see Table 13–1, "Types of Failed Credit Card Transactions".

CheckPoint Log Summary:

```
    PIN_CHARGE_CMD_VERIFY       3
    PIN_CHARGE_CMD_AUTH_ONLY    3
    PIN_CHARGE_CMD_CONDITION    0
    PIN_CHARGE_CMD_DEPOSIT      0
    PIN_CHARGE_CMD_REFUND       2
```

Without the **summary** option, the log summary is displayed and a menu if there are checkpoints to resolve.

**-search_count_limit n**
Specifies the number of records to return.

**-auth_pending**
Specifies the number of records with the auth pending status. This is only applicable to nontransactional payments.
-verbose
Displays information about successful or failed processing as the utility runs.

Note: This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter -verbose at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace filename.log with the name of the log file:

```
pin_clean other_parameter -verbose > filename.log
```

-help
Displays syntax and parameters for this utility.

Results
If the pin_clean utility doesn’t notify you that it was successful, look in the utility log file (default.pinlog) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.
Use this utility to collect the balance due for accounts that use credit card and direct debit payment methods.

When you use multiple payment processors, you run this utility for each one. See "Using More Than One Payment Processor" for more information.

By default, this utility collects payments for bills whose payment collection date is on the day the utility is run, as well as on the day before the utility is run. For example, if you run `pin_collect` on 01/01/06, payments are collected from 00:00:00 a.m. on 12/31/05 to 00:00:00 a.m. on 01/02/06.

To collect BRM-initiated payments for bills whose payment collection date is on a day other than the days listed above, use this utility's `start` and `end` parameters. See "Specifying Start and End Times" in BRM Configuring and Running Billing.

---

**Note:** The payment collection date of a bill (/bill object) is stored in the /billinfo object with which the bill is associated.

---

For more information on collecting BRM-initiated payments, see these topics:

- About Collecting BRM-Initiated Payments
- Configuring Payment Collection Dates for Automatic Payments

---

**Note:** To connect to the BRM database, the `pin_clean` utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in BRM System Administrator’s Guide.

---

**Location**

`BRM_Home/bin`

**Syntax**

```
pin_collect  -pay_type  payment_method  
              -vendor  payment_processor_name  
              [-active  |  -close  |  -inactive]  
              [-start  [mm/dd/yy  |  number_of_days]]  
              [-end  [mm/dd/yy  |  number_of_days]]  
              [-report]  
              [-rebill]  
              [-test]  
              [-verbose]  
              [-help]  
```

**Parameters**

- **-pay_type payment_method**
  Specifies the payment method. There are two possible values:

  - 10003 for credit card
  - 10005 for direct debit
Payment methods are defined in BRM_Home/include/pin_pymt.h.

-vendor payment_processor_name
Specifies the credit card processor or automated clearing house (ACH) to use for validating credit cards, debit cards, and direct debit transactions. This parameter is used to get the payment processor information from the /config/ach storable object.

For information on configuring payment processor information, see "Setting Up Merchants and Payment Processors" for more information.

-active l- close l -inactive
Specifies the status of the accounts to collect payments from.

-test
Runs a test to find out how many accounts meet the criteria without performing the collection. The test has no effect on the accounts. This is most useful when run with the -verbose and -report options.

-verbose
Displays information about successful or failed processing as the utility runs.

Note: This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter -verbose at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace filename.log with the name of the log file:

    pin_collect other_parameter –verbose > filename.log

-report
Displays more information than the verbose parameter. Requires the verbose option. Returns a list of the accounts being charged and the amount of each charge. This is an important verification tool when used with the -test and -verbose options: you can check the list of accounts and charges before charging them. See "Running Billing Utilities Manually" in BRM Configuring and Running Billing.

-rebill

-start [mm/dd/yyyy or yyyy l number_of_days]
Start date. Collects payments on the day utility is run and the day before the utility is run. For information on using this parameter, see "Specifying Start and End Times" in BRM Configuring and Running Billing.

-end [mm/dd/yyyy or yyyy l number_of_days]
End date. For information on using this parameter, see "Specifying Start and End Times" in BRM Configuring and Running Billing.

-help
Displays syntax and parameters for this utility.
Results

If the `pin_collect` utility doesn’t notify you that it was successful, look in the utility log file (`default.pinlog`) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.
**pin_deposit**

Use this utility to deposit all pre-authorized credit card and direct debit transactions made within the past 30 days (from yesterday).

When you use multiple payment processors, you run this utility for each one. See "Setting Up Merchants and Payment Processors" for more information.

---

**Note:** To connect to the BRM database, the **pin_clean** utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator’s Guide*.

---

See "About Collecting BRM-Initiated Payments" for more information on collecting BRM-initiated payments.

---

**Important:** For multidatabase systems, you must run the utility separately against each database in your system. See "Running Non-MTA Utilities on Multidatabase Systems" in *BRM System Administrator’s Guide*.

---

**Location**

`BRM_Home/bin`

**Syntax**

```
pin_deposit  -pay_type  payment_method
             -vendor  payment_processor_name
             [-start  mm/dd/yy | number_of_days ]
             [-end    mm/dd/yy | number_of_days ]
             [-test]
             [-verbose]
             [-help]
```

**Parameters**

**-pay_type  payment_method**

Specifies the payment method. There are two possible values:

- `10003` for credit card
- `10005` for direct debit

Payment methods are defined in `BRM_Home/include/pin_pymt.h`.

**-vendor  payment_processor_name**

Specifies the credit card processor or automated clearing house (ACH) to use for validating credit cards, debit cards, and direct debit transactions. This parameter is used to get the payment processor information from the `config/ach` storable object.

See "Setting Up Merchants and Payment Processors" for more information on configuring payment processor information.
-start [mm/dd/yy or yyyy | number_of_days]
-end [mm/dd/yy or yyyy | number_of_days]
Start and end date. For information on using start and end parameters, see "Specifying Start and End Times" in BRM Configuring and Running Billing.

-test
Runs a test to find out how many accounts meet the criteria without performing the deposit. The test has no effect on the accounts.

-verbose
Displays information about successful or failed processing as the utility runs.

Note: This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter -verbose at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace filename.log with the name of the log file:

pin_deposit other_parameter -verbose > filename.log

-help
Displays syntax and parameters for this utility.

Results
If the pin_deposit utility doesn't notify you that it was successful, look in the utility log file (default.pinlog) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.
pin_recover

Use this utility to resolve failed credit card and direct debit transactions. See "Resolving Failed BRM-Initiated Payment Transactions" for more information about resolving failed credit card and direct debit transactions.

---

**Note:** To connect to the BRM database, the pin_clean utility needs a configuration file in the directory from which you run the utility. See "Creating Configuration Files for BRM Utilities" in *BRM System Administrator’s Guide.*

---

**Important:** For multidatabase systems, you must run the utility separately against each database in your system. See "Running Non-MTA Utilities on Multidatabase Systems" in *BRM System Administrator’s Guide.*

---

**Location**

`BRM_Home/bin`

**Syntax**

```
pin_recover  -pay_type  payment_method  
-payer  payment_processor_name  

[-rfr | -resubmit batch_ID | -recover_payment]  
[-verbose] [-test] [-help]
```

**Parameters**

- **-pay_type payment_method**
  Specifies the payment method. There are two possible values:
  - 10003 for credit card
  - 10005 for direct debit

  Payment methods are defined in `BRM_Home/include/pin_pymt.h`.

- **-vendor payment_processor_name**
  Specifies the credit card processor or automated clearing house (ACH) to use for validating credit cards, debit cards, and direct debit transactions. This parameter is used to get the payment processor information from the `/config/ach` storable object.

  See "Setting Up Merchants and Payment Processors" for more information on configuring payment processor information.

- **-rfr**
  Uses the Paymentech request for response (RFR) file to retrieve and reprocess an incomplete batch. See "Resolving Transactions by Using a Request for Response (RFR) File" for more information.
**-resubmit batch_ID**
Resends the original batch with the same batch ID to avoid double authorization. To find the batch ID, run the **pin_clean** utility. See “Resubmitting Transactions” for more information.

---

**Note:** If you use a transaction processing service or credit card processing service other than Paymentech, make sure that it uses duplicate transaction detection. If not, using **-resubmit** can cause customers to be billed twice.

---

**-recover_payment**
Creates payment events for payments that have been successfully charged, but not recorded. See “Resolving Payments” for more information.

**-verbose**
Displays information about successful or failed processing as the utility runs.

---

**Note:** This parameter is always used in conjunction with other parameters and commands. It is not position dependent. For example, you can enter **-verbose** at the beginning or end of a command to initiate the verbose parameter. To redirect the output to a log file, use the following syntax with the verbose parameter. Replace **filename.log** with the name of the log file:

```
pin_recover other_parameter -verbose > filename.log
```

**-test**
Runs a test to find out how many accounts meet the criteria without performing the recovery. The test has no effect on the accounts.

**-help**
Displays syntax and parameters for this utility.

**Results**

If the **pin_recover** utility doesn’t notify you that it was successful, look in the utility log file (**default.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.