Contents

Preface .................................................................................................................................................................. ix
  Audience ......................................................................................................................................................... ix
  Downloading Oracle Communications Documentation .................................................................................... ix
  Documentation Accessibility .......................................................................................................................... ix

1 List of Payment Processing Features
  Payment Processing Features .......................................................................................................................... 1-1

2 Running Payment Collection Utilities
  About the Payment Collection Utilities ........................................................................................................ 2-1
  Running the pin_collect Utility to Collect BRM-Initiated Payments ................................................................. 2-1
    When to Run the pin_collect Utility .................................................................................................................. 2-2
    Setting Start and End Dates for pin_collect ..................................................................................................... 2-2
  Running the pin_deposit Utility to Deposit BRM-Initiated Payments ............................................................... 2-3
    When to Run pin_deposit .................................................................................................................................. 2-3
  Using More Than One Payment Processor ...................................................................................................... 2-3

3 Setting Up Credit Card and Debit Card Payments
  About Credit Card Validation and Authorization ............................................................................................. 3-1
  About Setting Up Payment Processing With Paymentech .................................................................................. 3-2
  Exchanging Connection Information With Paymentech
    Information You Need from Paymentech ........................................................................................................ 3-3
    Information Paymentech Needs from You ......................................................................................................... 3-3
  Configuring Paymentech Processing Performance
    Handling Concurrent Online Paymentech Requests ........................................................................................ 3-5
    Setting the Connection Timeout Length and Retries ....................................................................................... 3-6
  Monitoring the Paymentech Connection ........................................................................................................... 3-6

4 Paymentech Configuration Options
  Adding Soft Descriptor Information to Customer Statements ........................................................................ 4-1
  Changing How BRM Handles Paymentech Authorization Return Codes ......................................................... 4-2
  Changing How BRM Handles Paymentech Address Validation Return Codes ................................................. 4-3
  Specifying the Batch Mode Encryption Key ...................................................................................................... 4-3
  Requiring Additional Protection against Credit Card Fraud ............................................................................ 4-4
Specifying the Maximum Number of Digits Allowed for CVV2 Verification ........................................... 4-4
Enabling Paymentech Direct Debit Processing ........................................................................................ 4-5

5 Supported Paymentech Functionality

About Paymentech Account Verification ................................................................................................. 5-1
  About Action and Response Reason Codes ....................................................................................... 5-1
Supported Transaction Types ................................................................................................................. 5-2
Payment Formats and Batch Processing ................................................................................................. 5-2
Paymentech and International Transactions ......................................................................................... 5-3
How Paymentech Manager Handles Electronic Check Processing ....................................................... 5-3

6 Setting Up Merchant Accounts

Setting Up Merchant Accounts ............................................................................................................. 6-1
Specifying Merchant Accounts for the Payment DM ............................................................................. 6-2
Using More Than One Merchant Name ................................................................................................. 6-3

7 Masking Credit Card Numbers by Using Tokens

Credit Card Tokenization ....................................................................................................................... 7-1
Replacing Credit Card Numbers with Tokens ..................................................................................... 7-2
Purging Old Credit Card Event and Audit Trail Objects ......................................................................... 7-3

8 Testing Paymentech Credit Card Processing

About Testing Paymentech Credit Card Processing ............................................................................ 8-1
Setting Up the Paymentech Simulator .................................................................................................. 8-2
  Defining the Credit Card Functionality to Test .................................................................................. 8-2
  Configuring the Paymentech DM for Testing .................................................................................... 8-3
  Specifying an IP Address for the Paymentech Simulator .................................................................... 8-3
Running the Paymentech Simulators .................................................................................................... 8-4
  Simulating Failed Credit Card Transactions ..................................................................................... 8-4
  Resolving Failed Credit Card Transactions ........................................................................................ 8-4

9 Resolving Failed BRM-Initiated Payment Transactions

About Failed BRM-Initiated Payment Transactions ............................................................................ 9-1
  How BRM Records Transactions ...................................................................................................... 9-2
Resolving Transaction Errors .............................................................................................................. 9-3
Resolving Failed Deposits and Conditional Deposits ......................................................................... 9-5
Resubmitting Transactions .................................................................................................................... 9-6
Checking for Transactions in Paymentech Send Files ......................................................................... 9-7
Checking Paymentech Transmission Log Files .................................................................................... 9-7
Resolving Payments ............................................................................................................................. 9-8
  Resolving Payments for Custom Pay Types .................................................................................... 9-8
Troubleshooting Unresolvable Credit Card Transactions ..................................................................... 9-9

10 Processing Payment Batches in Billing Care

Processing Lockbox Batches .................................................................................................................. 10-1
11 Allocating Payments
   About Payment Allocation ................................................................. 11-1
   About Allocating Payments Manually ............................................... 11-2
      Allocating Multiple Payments for the Same Bill ......................... 11-3
      Allocating Payments Later ......................................................... 11-3
      Allocating Multiple Payments to the Same Account ................... 11-3
      Working with Multiple Currency Types in Billing Care ................ 11-3
   Finding Bills by Due Amount ............................................................ 11-3

12 Processing Atypical Payments
   Processing Overpayments and Underpayments .................................. 12-1
   Processing Late or Missed Payments ................................................ 12-1
   Reversing Payments ......................................................................... 12-1
   Refunding Externally Initiated Payments .......................................... 12-2
   Configuring Unconfirmed Payment Processing ................................... 12-2

13 Configuring Payment Methods
   About Payment Methods ..................................................................... 13-1
   Default Payment Methods .................................................................. 13-1

14 Customizing Payment Applications
   Customizing Payment Details Displayed in BRM Client Tools ........... 14-1
   About the Default /config/paymenttool Object .................................. 14-2
   Rules for Modifying Payment and Reversal Fields ............................ 14-2
   Creating an Object Definition for a New Payment or Reversal Event ... 14-3
   Changing the Order of Columns in Payment Tool ................................ 14-4
      Adding a New Column to Payment Tool .......................................... 14-5
      Adding Direct Debit Details to the Customer Center Payments Tab ... 14-5
      Customizing the Date Format of Batch Files in Payment Tool .......... 14-6
   Customizing the Date Format for Payment Center ............................. 14-7
   Improved Performance of Searches for Payment Events in Payment Center ... 14-8

15 Implementing SEPA Payment Processing
   About SEPA Payments ....................................................................... 15-1
   About Specifying SEPA Payment Information During Account Creation ... 15-2
      About Registering the Mandate for SEPA Direct Debit Payments ....... 15-2
      About the Different Types of Mandates ......................................... 15-2
   Managing Customer’s SEPA Payment Information ............................ 15-3
   Loading Your Creditor Information into the BRM Database ............. 15-3
   Processing SEPA Payments ............................................................... 15-5
      Creating SEPA Direct Debit Payment Requests ............................... 15-5
20 Configuring Payment Channels

Setting Up Payment Channel Information ................................................................. 20-1
Defining Payment Channel Information in BRM ......................................................... 20-1
Mapping Payment Channel IDs for BRM-Initiated Payments ............................... 20-2
Configuring Payment Channel IDs for Externally Initiated Payments ............ 20-2
Assigning Payment Channel IDs to Externally Initiated Payments ..................... 20-3

21 Customizing Payment Collection Dates for BRM-Initiated Payments

About Customizing Payment Collection Dates for BRM-Initiated Payments ........... 21-1
About Configurable Payment Collection Dates and On-Purchase Billing .......... 21-1
About Configurable Payment Collection Dates and Delayed Billing ..................... 21-3

22 Processing Payments in a Multischema System

Processing Payments in a Multischema System ............................................................ 22-1

23 Payment Utilities

load_pin_ach ................................................................................................................. 23-2
pin_balance_transfer ................................................................................................. 23-3
pin_cc_migrate ............................................................................................................. 23-4
pin_clean ....................................................................................................................... 23-6
pin_collect ..................................................................................................................... 23-7
pin_deposit .................................................................................................................... 23-9
pin_recover ................................................................................................................... 23-11
pin_sepa ......................................................................................................................... 23-13

About Payment Status ................................................................................................ 23-1

Default Payment Status Codes ............................................................................... A-1

About Payment Status

Default Payment Status Codes ............................................................................... A-2
Preface

This guide describes how to collect payments in Oracle Communications Billing and Revenue Management (BRM).

Audience
This guide is intended for operations personnel.

Downloading Oracle Communications Documentation
Product documentation is located on Oracle Technology Network:
http://docs.oracle.com

Additional Oracle Communications documentation is available from the Oracle software delivery Web site:
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 that have purchased support 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.
List of Payment Processing Features

This document lists the payment processing features in Oracle Communications Billing and Revenue Management (BRM).
To learn about payments, see BRM Concepts.

Payment Processing Features

This table lists payment processing features.

<table>
<thead>
<tr>
<th>Task</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collect payments from customers who pay by credit card or direct debit.</td>
<td>Running Payment Collection Utilities</td>
</tr>
<tr>
<td>Use more than one payment processor</td>
<td>Using More Than One Payment Processor</td>
</tr>
<tr>
<td>Set up BRM to collect credit card and direct debit payments.</td>
<td>Setting Up Credit Card and Debit Card Payments</td>
</tr>
<tr>
<td></td>
<td>Paymentech Configuration Options</td>
</tr>
<tr>
<td></td>
<td>Testing Paymentech Credit Card Processing</td>
</tr>
<tr>
<td></td>
<td>Supported Paymentech Functionality</td>
</tr>
<tr>
<td></td>
<td>Resolving Failed BRM-Initiated Payment Transactions</td>
</tr>
<tr>
<td>Add information to customer statements; such as charge offer names, or a service phone number.</td>
<td>Adding Soft Descriptor Information to Customer Statements</td>
</tr>
<tr>
<td>Improve fraud prevention by using Visa CVV2 numbers and American Express CID numbers.</td>
<td>Requiring Additional Protection against Credit Card Fraud</td>
</tr>
<tr>
<td>Configure Paymentech to process direct debit payments.</td>
<td>Enabling Paymentech Direct Debit Processing</td>
</tr>
<tr>
<td>Set up merchant accounts for credit-card processing</td>
<td>Setting Up Merchant Accounts</td>
</tr>
<tr>
<td>Payment processors, such as Paymentech, use merchant accounts to identify the companies who send them payments. You need to set up the merchant accounts in BRM.</td>
<td>Masking Credit Card Numbers by Using Tokens</td>
</tr>
<tr>
<td>Mask credit-card numbers in BRM applications and files by replacing the numbers with tokens.</td>
<td>Processing Payment Batches in Billing Care</td>
</tr>
<tr>
<td>Process payments by using Billing Care</td>
<td>Allocating Payments</td>
</tr>
</tbody>
</table>
## Payment Processing Features

<table>
<thead>
<tr>
<th>Task</th>
<th>Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manage:</td>
<td></td>
</tr>
</tbody>
</table>
  - Overpayments and underpayments  
  - Late or missing payments  
  - Payment reversals  
  - Refunds for externally-initiated payments  
  - Unconfirmed payments | Processing Atypical Payments |
| Learn about ways BRM can collect payments (credit card, check, and so on) and create custom payment methods. | Configuring Payment Methods |
| Customize BRM payment applications; for example, customize payment data displayed. | Customizing Payment Applications |
| Implement processing payments by using SEPA. SEPA payments are electronic payment transfers between bank accounts in the euro countries that participate in SEPA. | Implementing SEPA Payment Processing |
| Suspend failed payments, correct them, and recycle them. | Managing Suspended Payments |
| Set up payment fees to charge customers for failed payments. | Configuring Payment Fees |
| Set up payment incentives to reward customers for paying early or consistently. | Configuring Payment Incentives |
| Set up top-up payments. | Configuring Top-Ups |
| Define payment channels, which specify how payments are collected; for example, from the Internet, Interactive Voice Response (IVR) phone service, or Automated Clearing House (ACH). You can customize payment functionality based on payment channels. | Configuring Payment Channels |
| Customize payment collection dates. By default, BRM-initiated payments 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. | Customizing Payment Collection Dates for BRM-Initiated Payments |
| Process payments in a multi-schema system. | Processing Payments in a Multischema System |
This document describes how to use Oracle Communications Billing and Revenue Management utilities to collect and deposit payments.

Topics in this document:

- About the Payment Collection Utilities
- Running the pin_collect Utility to Collect BRM-Initiated Payments
- Running the pin_deposit Utility to Deposit BRM-Initiated Payments
- Using More Than One Payment Processor

See also:

- BRM Concepts
- List of Payment Processing Features
- Payment Utilities

### About the Payment Collection Utilities

To collect payments, use the following utilities:

- Use the `pin_collect` utility to collect the balance due for bills that are paid by credit card or direct debit.
- Use the `pin_deposit` utility to deposit pre-authorized credit card payments into your account.

The `pin_collect` utility performs the credit card authorization and deposits the payment at the same time. The `pin_deposit` utility only deposits payments that have been pre-authorized.

You typically run the `pin_collect` and `pin_deposit` utilities in the same scripts that run other billing utilities. See `BRM Configuring and Running Billing`.

If a payment fails, you can use the `pin_clean` utility and the `pin_recover` utility to resolve the failure. See "Resolving Failed BRM-Initiated Payment Transactions" for information.

### Running the pin_collect Utility to Collect BRM-Initiated Payments

For information about the `pin_collect` utility syntax, see "pin_collect".

The `pin_collect` utility collects payments for bills whose payment collection date is on the day the utility is run or on the day before the utility is run. For example, if you run...
pin_collect on 01/01/17, payments are collected from 00:00:00 a.m. on 12/31/16 to 00:00:00 a.m. on 01/02/17.

Use the pin_collect start and end parameters to collect payments on a range of dates.

Use the -rebill option to collect on any outstanding bills. Run this option in the weekly and monthly billing scripts.

You can configure payment collection:

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

- 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.

For information about collecting payments for corrective bills, see BRM Configuring and Running Billing.

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. Run the pin_bill_accts utility before running the pin_collect utility.

- 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.

Setting Start and End Dates for pin_collect

When any of the following conditions are met, the pin_collect utility collects payments for 2 days: the day before the utility is run and the day on which the utility is run:

- The start and end parameters are not set (the default).

- The start and end parameters are set to the same value.

- The start parameter is set to the current date, and the end parameter is not set.

To collect payments only on the day you run pin_collect, set the start parameter with a value of 0. For example:

pin_collect -start 0

You can also specify exact start and end dates, and you can specify a number of days before the current date for the start and end time calculation. The pin_collect utility only processes bills with a collection date within the start and end date range.
Running the pin_deposit Utility to Deposit BRM-Initiated Payments

For information about the pin_deposit utility syntax, see "pin_deposit".

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 Billing Care.

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.

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

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 credit-card authorizations can expire. 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".

---

To adjust performance, 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.

Using More Than One Payment Processor

You can use more than one payment processing Data Manager (DM) simultaneously to collect and validate payments. To use multiple payment processors, you must run the following utilities for each payment processor 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.
### Using More Than One Payment Processor

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

- **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 processors.

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

   ```plaintext
   pin_refund  -active  -pay_type 10003  -vendor fusa
   IF %ERR% EQU 0 IF %ERRORLEVEL% NEQ 0 SET ERR=%ERRORLEVEL%
   ...
   pin_collect  -inactive  -pay_type 10003  -vendor fusa
   IF %ERR% EQU 0 IF %ERRORLEVEL% NEQ 0 SET ERR=%ERRORLEVEL%
   ...
   pin_deposit  -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:

   ```plaintext
   pin_refund  -active  -pay_type 10003  -vendor new_vendor
   IF %ERR% EQU 0 IF %ERRORLEVEL% NEQ 0 SET ERR=%ERRORLEVEL%
   ...
   pin_collect  -inactive  -pay_type 10003  -vendor new_vendor
   IF %ERR% EQU 0 IF %ERRORLEVEL% NEQ 0 SET ERR=%ERRORLEVEL%
   ...
   pin_deposit  -pay_type 10003  -vendor new_vendor
   IF %ERR% EQU 0 IF %ERRORLEVEL% NEQ 0 SET ERR=%ERRORLEVEL%
   ...
   ``

**Important:** There might be several sets of entries for each payment processor. Be sure to add new entries for each set of existing entries.
This document provides instructions for setting up Oracle Communications Billing and Revenue Management (BRM) credit card and direct debit processing.

BRM includes an integration with the Paymentech payment card processor.

---

**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`.

---

Topics in this document:

- About Credit Card Validation and Authorization
- About Setting Up Payment Processing With Paymentech
- Exchanging Connection Information With Paymentech
- Configuring Paymentech Processing Performance
- Monitoring the Paymentech Connection

See also:

- **BRM Concepts**
- **Supported Paymentech Functionality**
- **Paymentech Configuration Options**
- **List of Payment Processing Features**

---

**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.

Credit card validation occurs during account creation, and when a customer changes their credit card number. If you use the Address Verification System (AVS), Paymentech gives you a discount for each credit card transaction charge.

Authorization occurs at the following times:

- At account creation, or when a customer changes their payment method to a credit card payment.
This type of authorization does 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.

- During billing, the pin_collect utility authorizes payments and deposits them.
- If there are charges during account creation, for example, a purchase fee.

---

**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".

---

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

1. BRM sends a validation request with an authorization to charge $1.00.

**Note:** The validation process requires a monetary charge. BRM issues an authorization for $1.00 so that only $1.00 is reserved on the customer’s credit card if the AVS request passes.

The credit card processor returns a validation code and an authorization code. BRM ignores the authorization code and uses the validation code to determine whether the address validation passed. For example, by default an address validation fails if the 5-digit ZIP code is wrong.

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

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

**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 with an authorization to charge for an actual amount, such as 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 whether the authorization passed.

---

**About Setting Up Payment Processing With Paymentech**

To enable BRM-initiated payment processing for Paymentech:

1. Install the Paymentech Manager software. See “Installing Paymentech Manager” in *BRM Installation Guide*.

2. Contact Paymentech to establish a link with Paymentech. See "Exchanging Connection Information With Paymentech".
3. Edit the `BRM_home/sys/dm_fusa/pin.conf` file to configure the Paymentech DM to connect to Paymentech. See *BRM System Administrator’s Guide* for information on configuring DMs.

4. Configure merchant accounts. See "Setting Up Merchant Accounts".

5. Set up the Paymentech HeartBeat application. See "Monitoring the Paymentech Connection".

6. Specify Paymentech configuration options; for example, enabling direct-debit processing and enabling fraud protection. See "Paymentech Configuration Options".

7. Configure performance options. See "Configuring Paymentech Processing Performance".

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

**Exchanging Connection Information With Paymentech**

Before you can connect BRM to Paymentech, you need to exchange connection information.

---

**Note:** Even if you already use Paymentech for credit card processing, you must plan for a setup and testing period for Paymentech direct debit.

---

**Information You Need from Paymentech**

You need the following information from Paymentech:

- The IP address and port for the Paymentech online server (the server used for creating accounts) and batch server (the server used for handling regular payments).
- The presenter ID and password, and the submitter ID and password.
- Merchant account numbers for each currency you support. The same sets of merchant account numbers can be used for both credit card and direct debit. See "Setting Up Merchant Accounts".

**Information Paymentech Needs from You**

Table 3–1 lists the information that Paymentech needs from you.

<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 (<em>dm_fusa</em>).</td>
<td>None. This is required only to use the Paymentech HeartBeat application, which is integrated with the Paymentech Data Manager. For more information, see &quot;Monitoring the Paymentech Connection&quot;.</td>
</tr>
<tr>
<td>Is this for an existing Presenter ID (PID)?</td>
<td>No</td>
</tr>
<tr>
<td>What is the application software that formats the file?</td>
<td>Written by in-house programmers</td>
</tr>
</tbody>
</table>
### 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 communications software that sends the file?</td>
<td>Customized by the software vendor</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 the entry for Paymentech online format in “BRM Software Compatibility” in BRM Installation Guide.</td>
</tr>
<tr>
<td>Will you load balance online authorizations 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 the entry for Paymentech batch format in “BRM Software Compatibility” in BRM Installation Guide.</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 in production? (How many records/transactions?)</td>
<td>None.</td>
</tr>
<tr>
<td></td>
<td>This number should be based on your company’s projected customer account creation 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>
Configuring Paymentech Processing Performance

Configure the following options:

- Handling Concurrent Online Paymentech Requests
- Setting the Connection Timeout Length and Retries

To configure Paymentech options related to account creation, see BRM Managing Customers.

Handling Concurrent Online Paymentech Requests

You can increase billing performance by using the fusamux program. Because 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.

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 must change entries in the Paymentech DM to point to fusamux instead of pointing to 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:

---

<table>
<thead>
<tr>
<th>Paymentech Information</th>
<th>BRM Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>What Paymentech function do you intend to test?</td>
<td>This list reflects a typical pre-paid services company.</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 (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>
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.

### 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/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.

### Monitoring the Paymentech Connection

The Paymentech HeartBeat application is a background process that checks the connectivity between BRM and Paymentech. When the Paymentech DM successfully connects to Paymentech, Paymentech acknowledges the connection by sending a HeartBeat message. The Paymentech DM responds by sending back a HeartBeat message to Paymentech.

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.

---

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.

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

```
Received (20) chars: Heartbeat request [H019999999813123258^M]
Sending Heartbeat response [HI199999998131233300^M]
```
If these entries are missing or are not continuous for the duration of the connection with Paymentech, work with 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:

```
Received (20) chars: Heartbeat response Validation failed in process_it() :
HI1999999981323300'\n```

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 "Testing Paymentech Credit Card Processing".

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)
```
Paymentech Configuration Options

This document provides instructions for setting up Oracle Communications Billing and Revenue Management (BRM) credit card and direct debit processing. BRM includes an integration with the Paymentech payment card processor.

---

**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`.

---

Topics in this document:

- Adding Soft Descriptor Information to Customer Statements
- Changing How BRM Handles Paymentech Authorization Return Codes
- Changing How BRM Handles Paymentech Address Validation Return Codes
- Specifying the Batch Mode Encryption Key
- Requiring Additional Protection against Credit Card Fraud
- Enabling Paymentech Direct Debit Processing

See also:

- BRM Concepts
- Supported Paymentech Functionality
- List of Payment Processing Features

---

**Adding Soft Descriptor Information to Customer Statements**

You can use soft descriptors to add information to customers’ credit-card or checking-account statements. You can add to these statements:

- Your "doing business as" (DBA) name
- Charge offer 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, charge offer name, and phone entries:

- `dm_fusa sd_merchant_dba DBA`
- `dm_fusa sd_merchant_pdt ChargeOfferName`
On the customer's statement, an asterisk is used to separate the DBA name and charge offer name. If the entry is longer than 22 characters (including spaces), it is truncated on the statement. In this 22-character-maximum line, the asterisk delimiter can appear in position 4, 8, or 13.

For example, if the merchant name is psi, the DBA name is BRM, the pdt (charge offer) is InternetSVC, and customer service number is 800-555-1234, use the following entries:

- dm_fusa sd_psi_dba BRM
- dm_fusa sd_psi_pdt InternetSVC
- dm_fusa sd_psi_phone 800-555-1234

To use multiple DBA names, charge offer names, or phone numbers, you must customize the PCM_OP_PYMT_POL_PRE_COLLECT policy opcode. See BRM Opcode Guide.

To add soft descriptor 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.

To create multiple DBA names, charge offer names, and phone number entries, you must customize the PCM_OP_PYMT_POL_PRE_COLLECT policy opcode. See BRM Opcode Guide.

---

**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.
Changing How BRM Handles Paymentech Address Validation Return Codes

Paymentech provides return codes when verifying customer addresses. AVS validates only credit cards with addresses in the United States and Canada. To change how BRM responds to validation return codes, edit the PCM_OP_PYMT_POL_VALIDATE policy opcode source. See BRM Opcode 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 such as the credit card and security code is encrypted.

You specify the encryption method and key in the Paymentech configuration file. The encryption method supported is MD5. For more information, see “About MD5 Encryption” in BRM Developer’s Guide.

Tip: You should change the encryption key regularly. Before changing the encryption key, ensure that 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. Find the following line:
   - crypt
3. Do one of the following:
   - To specify the MD5 encryption key, change the line to the following:
     - crypt md5| libpin_crypt4qm.so "encryption_key"
   - To specify the AES encryption key, change the line to the following:
     - crypt aes| libpin_crypt4qm.so "encryption_key"

where encryption_key is the key you generate.

For example:

For MD5:
- crypt md5| libpin_crypt4qm.so "24CFD43E8CE5273B0B7781140CB71B92"

For AES:
- crypt aes| libpin_crypt4qm.so "24CFD43E8CE5273B0B7781140CB71B92"

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

4. Save and close the file.
5. Stop and restart the Paymentech DM.

**Requiring Additional Protection against Credit Card Fraud**

Paymentech offers additional fraud prevention using Visa CVV2 numbers and American Express CID numbers.

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 account creation, 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 does not 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 account creation. 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
Enabling Paymentech 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:
   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.
Paymentech is a payment card processor. This document describes the Paymentech functionality supported by Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

- About Paymentech Account Verification
- Supported Transaction Types
- Payment Formats and Batch Processing
- Paymentech and International Transactions
- How Paymentech Manager Handles Electronic Check Processing

See also:

- BRM Concepts
- Setting Up Credit Card and Debit Card Payments
- List of Payment Processing Features

**About Paymentech Account Verification**

BRM supports Paymentech’s Account Verification feature. Paymentech recommends the use of Account Verification to differentiate validation requests from authorization requests. This is because Visa imposes a penalty for all authorization requests that are neither deposited nor reversed.

Account Verification supports the following Paymentech credit-card payment methods:

- VI for Visa
- MC for MasterCard

Account Verification supports EC for direct debit cards in the United States and Canada.

**About Action and Response Reason Codes**

BRM sends the following Action Codes to indicate the type of service Paymentech must perform on the transaction:

- To verify a Paymentech-supported direct debit transaction, BRM sends the action code LO and a transaction amount is $0.00. Paymentech validates the transaction against various validation files. If the account verification is successful,
Paymentech responds with a Response Reason Code 101 (Validation passed
Paymentech negative file and data edit check).

- To verify a direct debit transaction against a third-party negative file for United
  States ECP, BRM sends the action code VO and a transaction amount $0.00. If the
  account verification passes, Paymentech responds with a Response Reason Code
  102 (Account verification Passed external negative file).

- To verify the account for VISA or MasterCard, BRM sends the action code VF and
  a transaction amount $0.00. If the account verification is successfully approved,
  Paymentech responds with a Response Reason Code 104 (No Reason to Decline).

For more information, see the Paymentech documentation.

**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 sales 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 sales location where transactions are
  going to be on a periodic basis.

- A transaction type 7 indicates a channel encrypted transaction between a
  cardholder and a seller. 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
Paymentech documentation.

**Payment Formats and Batch Processing**

Paymentech batch processing 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

Consider the following points about batch processing functionality:

- 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. You can obtain more information on obtaining the certification
  from the Paymentech Web site.
For the UK Domestic Maestro (Switch/Solo) card (MOP = SW) with batch processing functionality, Paymentech expects the card issue date and the issue number (if present) in the UK Domestic Maestro extension record.

- BRM does not support creating accounts by using UK Domestic Maestro (Switch/Solo) card type. For existing subscribers, transactions other than the refund (Action Code = RF) are not supported.
- To enhance your existing payment processing with Paymentech’s Account Verification feature, before doing so, ensure that all preauthorized payments are deposited or reversed.

For more information about Paymentech’s 120-byte batch format, see the Paymentech documentation.

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.

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.

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 Paymentech Account Verification”.

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.
- **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, 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.
Setting Up Merchant Accounts

Payment processors, such as Paymentech, use merchant accounts to identify the companies who send them payments. You need to set up the merchant accounts in BRM.

This document describes how to set up merchant accounts for BRM-initiated payments in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

- Setting Up Merchant Accounts
- Specifying Merchant Accounts for the Payment DM
- Using More Than One Merchant Name

See also:

- BRM Concepts
- Setting Up Credit Card and Debit Card Payments
- List of Payment Processing Features

Setting Up Merchant Accounts

To manage BRM-initiated payments, a payment processor, such as Paymentech, creates a merchant account for your company. For example, your account might be assigned the merchant account number 050505. If you accept payments in multiple currencies, a payment processor creates an account number for each currency.

When you send payments to a payment processor, you need to send them the merchant account number so they can determine where to deposit your BRM-initiated payments. To do so, you configure the merchant account number in the configuration file for the Data Manager that sends the payment request to the payment processor:

```
-dm_fusa mid_ispname_840 050505
```

This entry maps the merchant ID (mid_ispname_840) with the merchant account number (050505). The merchant ID is a combination of the merchant name (ispname) with the currency ID (840). You need to load merchant names into the BRM database.

To load merchant names into the BRM database, you edit the pin_ach file and load it by using the load_pin_ach utility. An entry in the pin_ach file includes the merchant name:

```
fusa 0.0.0.1 /payment -l ispname 0
```
If you use multiple currencies, you are given multiple merchant account numbers. You configure multiple entries in the DM `pin.conf` file, for example:

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

You specify merchant names and the payment processors that process your BRM-initiated payment transactions for the entire system. You can specify any number of merchant ID and Merchant account number pairs.

To load merchant names into the BRM database:

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 name used by each payment processor is the first merchant name listed for the payment processor.

   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.
   - `0.0.1.1 /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 payment processor. 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”.

2. Save the `pin_ach` file.

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

   ```
   load_pin_ach pin_ach
   ```

   For more information, see "load_pin_ach".

---

**Specifying Merchant Accounts for the Payment DM**

To enable the Paymentech DM to send merchant account data:

1. Open the Paymentech DM configuration file (`BRM_home/sys/dm_fusa/pin.conf`).
2. Change the merchant account entry. Use this syntax:
Using More Than One Merchant Name

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

If you use multiple merchant names, each merchant name must be entered in the following files:

- The payment processor configuration file (`BRM_home/sys/data/pricing/example/pin_ach`).
- The payment processor Data Manager (DM) configuration file (for example, `BRM_home/sys/dm_fusa/pin.conf`).

For example:

```
- dm_fusa mid_ispname_840 050505
```

3. Save and close the file.
4. Stop and restart the Paymentech DM.

Using More Than One Merchant Name

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

If you use multiple merchant names, each merchant name must be entered in the following files:

- The payment processor configuration file (`BRM_home/sys/data/pricing/example/pin_ach`).
- The payment processor Data Manager (DM) configuration file (for example, `BRM_home/sys/dm_fusa/pin.conf`).
This document describes how to mask credit card numbers by using tokens in Oracle Communications Billing and Revenue Management (BRM).

See also:
- Adding Soft Descriptor Information to Customer Statements
- Supported Paymentech Functionality
- Paymentech Configuration Options
- List of Payment Processing Features

**Credit Card Tokenization**

Credit card tokenization is a secure method of storing credit and debit card data. It replaces credit and debit card numbers with random identifiers called tokens. Tokens are typically the same length as the credit or debit card numbers and include the last four digits of the card numbers.

By default, credit card tokenization is enabled. When the tokenization is enabled, BRM receives tokens from Paymentech and stores only the tokens in the BRM database. The tokens are then used for any BRM-initiated payments instead of the actual card numbers. The actual card numbers and their mapping to the tokens are stored securely in Paymentech. Tokens are valid only between the sales system and the credit card processor. Therefore, the tokens can be transmitted safely without the risk of exposing the credit or debit card data.

Credit card tokenization occurs at the following times:
- During account creation
- When credit cards are used for one-time payments
- When customers change their credit or debit card number
- When customers change to the credit card payment method

If credit card validation fails, tokenization does not occur. In this case, a string value (asterisks (******) followed by the last four digits of the credit card) is stored in the /event/billing/validate/cc object. The string value can be used to authenticate a credit or debit card but cannot be used for any transaction.

If you have already processed credit cards before enabling tokenization, the BRM database stores untokenized credit card numbers. To tokenize existing credit or debit card numbers stored in the BRM database, see "Replacing Credit Card Numbers with Tokens".
Replacing Credit Card Numbers with Tokens

After enabling tokenization and replacing existing credit card and debit card numbers, purge the database of old numbers. See "Purging Old Credit Card Event and Audit Trail Objects".

Replacing Credit Card Numbers with Tokens

**Note:** If you are migrating legacy credit card data into the BRM database, migrate the data before replacing numbers with tokens. See BRM Migrating Accounts to the BRM Database.

If you have already processed credit cards before enabling tokenization, the BRM database stores untokenized credit card numbers. You need to replace those numbers with tokenized numbers.

Before replacing numbers with tokens:

1. If you are migrating from an external database, migrate the data. See BRM Migrating Accounts to the BRM Database.
2. Enable credit card tokenization in BRM. See "Credit Card Tokenization".

To replace credit card numbers with tokens, run the **pin_cc_migrate** utility. See "**pin_cc_migrate**".

- To replace all the credit card numbers with tokens, run the following command:
  
  **pin_cc_migrate** -vendor payment_processor_name

  where **payment_processor_name** is the credit card processor or ACH to use for validating credit and debit cards.

  For example:
  
  pin_cc_migrate -vendor fusa

- To replace only a specified number of credit card numbers in the **/payinfo/cc** objects, run the following command:

  **pin_cc_migrate** -vendor payment_processor_name -num number

  where **number** is the number of **/payinfo/cc** objects to be selected for tokenization.

  For example:
  
  pin_cc_migrate -vendor fusa -num 10

- To replace the credit card numbers for only a specified account, run the following command:

  **pin_cc_migrate** -vendor payment_processor_name -account account_POID

  where **account_POID** is the Portal object ID (POID) of the account to select for tokenization.

  For example:
  
  pin_cc_migrate -vendor fusa -account 3421343

- To specify the time range for selecting credit card numbers for tokenization, run the following command:

  **pin_cc_migrate** -vendor payment_processor_name -start_date mm/dd/yy -end_date
For example:
```
pin_cc_migrate -vendor fusa -start_date 01/01/11 -end_date 10/30/11
```

The start and end dates specify the time range for selecting `/payinfo/cc` objects for tokenization.

See "pin_cc_migrate" for information on the `pin_cc_migrate` utility.

---

## Purging Old Credit Card Event and Audit Trail Objects

When you run the `pin_cc_migrate` utility, only the credit and debit card numbers stored in `/payinfo/cc` objects are replaced with tokens. The credit and debit card numbers stored in the following objects are not replaced:

- Event objects created for credit card validation and credit card charges (such as `/event/billing/charge/cc` objects)
- Audit trail objects created for tracking credit card payments (such as `/event/audit/customer/payinfo/cc` objects)

Oracle recommends that you purge these event and audit trial objects immediately after you run the `pin_cc_migrate` utility. You can purge the old event and audit trail objects by using the BRM utilities or purging scripts. See the following for more information:

- To purge the event objects, see "About Purging Event Objects" in *BRM System Administrator’s Guide*.
- To purge the audit trail objects, see "Archiving Audit Data" in *BRM Developer’s Guide*.

---

**Important:** If you purge `/event/billing/charge/cc` objects, you cannot refund payments to the same credit card accounts that were used for one-time payments made before running `pin_cc_migrate`. 
This document describes how to test Paymentech credit card processing in Oracle Communications Billing and Revenue Management (BRM).

See also:
- Resolving Failed BRM-Initiated Payment Transactions
- Setting Up Credit Card and Debit Card Payments
- Supported Paymentech Functionality
- Running Payment Collection Utilities
- List of Payment Processing Features

About Testing Paymentech Credit Card Processing

Paymentech provides connections for testing credit card and direct debit processing, but you must schedule testing times with Paymentech. In addition, you can use the BRM Paymentech simulators to test credit card and direct debit processing without connecting to Paymentech.

**Caution:** To test credit card and debit card processing with BRM Paymentech simulators, you must use the account numbers from the test environment only.

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 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 a 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 \textit{BRM\_home/bin}. It includes two utilities, \texttt{answer\_s} and \texttt{answer\_b}.

\begin{itemize}
  \item The \texttt{answer\_s} utility simulates online transactions.
  \item The \texttt{answer\_b} utility simulates batch transactions.
\end{itemize}

You can create test accounts that use a credit card payment method. You must use one of the following pairs of credit card numbers and expiration dates listed in Table 8–1 for your test accounts:

\begin{table}[!h]
\centering
\begin{tabular}{|c|c|}
\hline
Credit Card Number & Expiration Date \\
\hline
4444 1111 2222 3333 & 0999  \\
4101 0000 0000 0000 & any expiration date \\
\hline
\end{tabular}
\caption{Example Credit Card Expiration Data}
\end{table}

\section*{Setting Up the Paymentech Simulator}

Setting up the Paymentech simulator involves the following tasks:

\begin{itemize}
  \item \textbf{Defining the Credit Card Functionality to Test}
  \item \textbf{Configuring the Paymentech DM for Testing}
  \item \textbf{Specifying an IP Address for the Paymentech Simulator}
\end{itemize}

\section*{Defining the Credit Card Functionality to Test}

You can define which area of functionality to test with \texttt{answer\_s} and \texttt{answer\_b} by editing the Paymentech simulator configuration file (\texttt{BRM\_home/apps/fusa\_server/pin.conf}). This file includes configuration instructions.
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.

**Configuring the Paymentech DM for Testing**

To configure the Paymentech DM 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.
6. Stop and restart the Paymentech DM.

**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 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

---

**Note:** The entries can be changed interactively because the answer_a and answer_s servers read them from the configuration file at each connection.
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.
6. Advance the time one month and run `pin_bill_day`.
7. Verify that the amount due is not collected.
8. 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 an account that uses a credit card.
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`.

where IP_address is the IP address of the system running the simulator.

For example:

```bash
- answer answer_name 102.13.112.122
```
c. Stop the answer_b utility while billing runs.

3. 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".
Resolving Failed BRM-Initiated Payment Transactions

This document describes how to resolve failed credit card and direct debit transactions in Oracle Communications Billing and Revenue Management (BRM).

For information about the utilities used for resolving BRM-initiated payment transactions, see "pin_clean" and "pin_recover".

Topics in this document:

- About Failed BRM-Initiated Payment Transactions
- Resolving Transaction Errors
- Resolving Failed Deposits and Conditional Deposits
- Resubmitting Transactions
- Checking for Transactions in Paymentech Send Files
- Checking Paymentech Transmission Log Files
- Resolving Payments
- Troubleshooting Unresolvable Credit Card Transactions

See also:

- Testing Paymentech Credit Card Processing
- Setting Up Credit Card and Debit Card Payments
- Supported Paymentech Functionality

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 does not 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 received the transaction successfully, checkpoint records are left in the database. To resolve failed transactions, you must resolve all checkpoint records. Transactions usually have a checkpoint record for the following reasons:
About Failed BRM-Initiated Payment Transactions

- A transaction batch was received by the credit card processor, but it wasn’t
  processed completely. To resolve this error, you must 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 must 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 do not 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 can delete failed verifications without restoring them, but you usually
need to restore failed authorizations.

**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
Resolving Transaction Errors

- FAIL_INVALID_CONTENT = 16
- FAIL_TECHNICAL_PROBLEM = 17
- DEPOSIT_PENDING = 777
- AUTH_PENDING = 888
- 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.

---

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

---

Resolving Transaction Errors

You should check for transaction errors every day. The best way to do this is to create a script that:

1. Runs the "pin_clean" utility and reports transaction failures.

   ```bash
   pin_clean -summary
   ```

2. Writes the output to a file. The `pin_clean` utility writes output to stdout.

   ```bash
   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.
   ```

Running the `pin_clean` utility automatically only reports errors. To resolve errors, you need to run the `pin_clean` and `pin_recover` utilities manually.

---

Important: Do not delete deposit or conditional deposit records until you know whether the corresponding charge was successful. The length of time for charges to occur depends on the payment processor. Generally, you should only delete records generated more than 7 days previously. Otherwise, you might charge customers twice if you delete records created within the duplicate detection period. Check with your payment processor.

---

1. Run the `pin_clean` utility with the `summary` option:

   ```bash
   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
```

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. Follow the instructions to review or delete transactions, for example:

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

You can delete all verifications, because they are not associated with any charge. For authorizations and refunds, you might need to repeat the transaction. Read the event details to find out if this is a transaction you need to repeat. For example:

```
0 PIN_FLD_SYS_DESCR       STR [0] "Authorization"
0 PIN_FLD_ACCOUNT_OBJ    POID [0] 0.0.0.1 /account 28456 0
0 PIN_FLD_AMOUNT          NUM [0] 100.000000
0 PIN_FLD_CREATED_T    TSTAMP [0] (827435459) Thu Mar 21 11:10:59 2017
```

Make a note of the amount and the account number so that you can repeat the transaction later.

Table 9–1 describes how to resolve each type of transaction.

**Important:** To resolve failed deposits ("pin_deposit") and conditional deposits ("pin_collect") batches, run the "pin_recover" utility with the `rfr` option before you run the `pin_clean` utility. See "Resolving Failed Deposits and Conditional Deposits".

**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. Use the `pin_recover` utility to resubmit the batch. See "Resubmitting Transactions".
Resolving Failed Deposits and Conditional Deposits

Resolving Failed Deposits and Conditional Deposits

To resolve failed deposits ("pin_deposit") and conditional deposits ("pin_collect") batches, run the "pin_recover" utility with the \texttt{rfr} option before you run the \texttt{pin_clean} utility.

\textbf{Important:} You cannot use the \texttt{rfr} option if the transaction was an online transaction such as a charge or refund made by using Billing Care.

1. Request an RFR file from the Paymentech customer support service. If there is no RFR file, you cannot complete this procedure. See "Resubmitting Transactions".

\textbf{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 \texttt{pin_recover} utility with the \texttt{rfr} option:

\texttt{pin_recover -rfr}

The \texttt{pin_recover} utility is in \texttt{BRM_home/bin}.

3. After the \texttt{pin_recover} utility has finished, run it again by using the \texttt{rfr} option. Paymentech sometimes posts multiple RFR files, and you must process all of them before continuing.

\begin{table}[h]
\centering
\caption{Types of Failed Credit Card Transactions}
\begin{tabular}{|l|p{12cm}|p{12cm}|}
\hline
Record Type & Error & Action \\
\hline
verify & The connection was lost during an online transaction such as account creation. & Delete the transaction record from the BRM database. You do not need to resubmit it. \\
\hline
authorize & The connection was lost during an online transaction such as account creation, or a one-time charge to a single account. & Delete the transaction record from the BRM database. If necessary, repeat the transaction; for example, use Billing Care to charge the account again. Because the transaction was for an authorization, not for a payment, the customer cannot be charged twice. \\
\hline
conditional deposit & The connection was lost while running the "pin_collect" utility. & See "Resolving Failed Deposits and Conditional Deposits". \\
\hline
deposit & The connection was lost while running the "pin_deposit" utility. & See "Resolving Failed Deposits and Conditional Deposits". \\
\hline
refund & The connection was lost when a refund was made. & Delete the transaction record from the BRM database. If necessary, repeat the transaction; for example, use Billing Care to charge the account again. Because the transaction was for an authorization, not for a payment, the customer cannot be charged twice. \\
\hline
\end{tabular}
\end{table}
Resubmitting Transactions

4. Run the pin_clean utility in summary mode again. If you still find transaction errors, see “Resubmitting Transactions”.

Resubmitting Transactions

Caution: If you use a payment processor other than Paymentech, ensure 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.

If running the pin_recover utility with the rfr option does not 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.

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 must 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, delete them.

---

### 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 Data Manager (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 Connection Manager (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".

---

### Checking Paymentech Transmission Log Files

The pin_collect utility creates transmission log files to record the billing transactions sent to and received from Paymentech. The files for information sent have the prefix fusas (Paymentech), and the files for information received have the prefix fusar (Paymentech).

The Paymentech transmission log files are stored in the system temporary directory. If that directory is not defined or does not exist, BRM looks for a different folder, in the following order:

- The Directory defined by the temp_dir entry in the Paymentech DM configuration file (BRM_home/sys/dm_fusa/pin.conf)
Resolving Payments

You must delete or archive billing transmission logs periodically to prevent the file system from overflowing. If data security is an issue, delete or archive the files to a secure location immediately after you run billing. Good business practice suggests archiving the files for at least 30 days before discarding them.

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. Because the charge has 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_BILLING_PAYMENT_CC_T value = 0.

Resolving Payments for Custom Pay Types

To resolve payments for custom pay types, you must perform additional configuration steps before you run the pin_recover utility with the recover_payment option for the first time.

To resolve payments for custom pay types:

1. Customize the PCM_OP_PYMT_POL_CHARGE policy opcode to perform the following when it processes your custom pay type:
   a. In the policy opcode’s output flist, set the PIN_FLD_BATCH_INFO.PIN_FLD_RESULT field to PIN_CHARGE_RES_OFFSET.
   b. Update your custom /event/billing/charge/* subclass by setting its PIN_FLD_Charge.PIN_FLD_RESULT field to 1000 (PIN_CHARGE_RES_OFFSET).

2. Go to the BRM_Home/apps/pin_billd directory.

3. Open the pin.conf file in a text editor.

4. Add the following line for each custom pay type:
   - pin_recover config_payment paymentPOID
where paymentPOID is the POID of your /config/payment object.

For example:
- pin_recover config_payment 0.0.0.1 /config/payment 200

**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 does not 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. Be 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.

---

**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.

---

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_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.
You use Billing Care to process externally-initiated payments, such as payments made by check.

This document describes how to use Oracle Communications Billing and Revenue Management (BRM) to handle batches of externally initiated payments.

Topics in this document:
- Processing Lockbox Batches
- Importing Batch Payment Files into Billing Care
- Configuring Payment Tool to Lock at the Account Level during Batch Processing
- Managing Nonvalidated Batch Entries

See also:
- BRM Concepts
- List of Payment Processing Features

**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 Billing Care.

Most banks that perform lockbox processing can format a text file according to your specifications. You can specify:
- Which data to include
- 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 Billing Care to import data directly from the file. See "Importing Batch Payment Files into Billing Care".
Note:
- You might need to create an application to retrieve the file.
- If the bank creates the file with the EBCDIC character set, you must create an application to convert it to ASCII.
- Payment Tool does not support the EDI 823 format.

Importing Batch Payment Files into Billing Care

You can import data from text files into Billing Care batch payment 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.

Batch payment files are text files containing payment information, such as account number, payment amount, and payment date, in delimiter-separated rows.

Before you begin importing data, ensure that you know how your data is formatted:
- When you import data, you must 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
```

Configuring Payment Tool to Lock at the Account Level during Batch Processing

When processing a batch of payments, Payment Tool 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, the payment item and payment event is recorded 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 Data Manager (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 CM.

**Managing Nonvalidated Batch Entries**

While processing a payment, refund, or reversal batch, you might have some entries that cannot be validated easily. You can create a batch that includes only those entries with validation errors. This enables 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 nonvalidated entries, first validate the batch, then choose **Tools - Create Exception Batch**. A new batch is created that includes only invalid batches. The invalid batches are removed from the validated batch.
Allocating Payments

This document describes how to allocate payments in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

- About Payment Allocation
- About Allocating Payments Manually
- Finding Bills by Due Amount

See also:

- BRM Concepts
- List of Payment Processing Features

About Payment Allocation

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.

Payments are allocated according to how they were collected:

- BRM-initiated payments for credit card or direct debit accounts are automatically allocated during the collection process.
- Externally-initiated payments, such as by check, are manually allocated by a payment clerk.

You can configure the allocation level for payments. The allocation level determines where the payment is applied:

- **Account**

  When a payment is made at account level (without specifying bill or item), the payment can be allocated to accounts with a single bill unit (`billinfo` object) 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 Billing Care 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. See BRM Opcode Guide.

When allocating payments manually, you can override the default distribution.

---

**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".

---

**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.

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 do so, customize the PCM_OP_PYMT_POL_VALIDATE_PAYMENT policy opcode. See BRM Opcode Guide.

---

**Note:** You cannot allocate a payment to a nonpaying 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.

---

**About Allocating Payments Manually**

When processing payments manually in Billing Care, you can allocate payments before or after validating them. When you validate a payment, if you see a message in the status bar that says something similar to “Payment allocation required,” you must allocate the payment.

Manual 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 recommends that you allocate the payment, but allocation is not required.

You can manually allocate payments as follows:

- You can allocate an account-level payment when the payment is applied to an account with multiple bill units (/billinfo objects).
- If your batch supports bill-level allocation, you can allocate a payment to a specific bill when there are multiple unpaid bills for an account.
- You can specify which items on the bill to apply the payment to.
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 Later

You can create a batch for only those payments that need allocations. See “Managing Nonvalidated Batch Entries”.

As an alternative, you can apply a payment to an account and allocate it later by using Payment Tool. When you apply a payment at the account level, the account balance is reduced, but items and bills are not closed.

Allocating Multiple Payments to the Same Account

By default, to prevent duplicate entries, you cannot use Payment Tool to allocate more than one payment in a batch to a single account. However, you might need to allocate more than one payment if the account uses open item accounting and you need to allocate payments to more than one bill.

To allocate more than one payment to an account:

1. Open the Payment Tool INI file (C:\Windows\PaymentTool.ini).
2. Change this entry:
   
   \texttt{ALLOWACCOUNTDUP=0}

   To this:

   \texttt{ALLOWACCOUNTDUP=1}

   You do not need to exit Payment Tool; the change takes effect the next time you validate a payment.

Working with Multiple Currency Types in Billing Care

When you allocate payments with Billing Care, you choose the currency to use for each batch of payments. You can allocate a payment to an account in any currency. The amount will be converted to the account’s primary currency and then posted to the account.

Finding Bills by Due Amount

If BRM cannot find a bill to allocate a payment to, BRM searches for a bill whose total due amount matches a specified payment amount. The search is restricted to bills that
belong to the account with which the payment is associated. By default, this search is
disabled.

To enable this feature, run the `pin_bus_params` utility to change the
`SearchBillAmount` business parameter. For information about this utility, see `BRM
Developer’s Guide`.

To enable this search:

1. Go to `BRM_home/sys/data/config`.
2. Create an XML file from the `/config/business_params` object:
   
   ```
   pin_bus_params -r BusParamsAR bus_params_AR.xml
   ```

3. In the file, change `disabled` to `enabled`:
   
   ```
   <SearchBillAmount>enabled</SearchBillAmount>
   ```

4. Save the file as `bus_params_AR.xml`.
5. Load the XML file into the BRM database:
   
   ```
   pin_bus_params bus_params_AR.xml
   ```

6. Stop and restart the CM.
7. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG
   parameter. For more information, see `BRM System Administrator’s Guide`. 
This document describes how to handle payments in Oracle Communications Billing and Revenue Management (BRM) that are not tailored to your normal payment processing.

Topics in this document:

■ Processing Overpayments and Underpayments
■ Processing Late or Missed Payments
■ Reversing Payments
■ Refunding Externally Initiated Payments
■ Configuring Unconfirmed Payment Processing

See also:

■ BRM Concepts
■ List of Payment Processing Features

Processing 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 "Allocating Payments".

Processing 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.

See BRM Opcode Guide.

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 enables BRM to treat the payment as if it never happened.
You use Payment Tool to process payment reversals. See BRM Opcode Guide to create a custom application for reversing payments.

Payment Tool reverses payment directly. Payments are reversed indirectly during suspended payment processing. See "Managing Suspended Payments".

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 technical details on payment reversals and recycling, see BRM Opcode Guide.

---

**Refunding Externally Initiated Payments**

To refund externally initiated payments, 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.

- You cannot refund suspended payments. For information on suspended payment processing, see "Managing Suspended Payments".
- You cannot reverse a refund. If you refund a customer account by mistake, adjust the account for the refunded amount. To do so, you need to customize A/R opcodes. See BRM Opcode Guide.

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.

---

**Configuring Unconfirmed Payment Processing**

BRM requires acknowledgment from a bank or payment processor before posting BRM-initiated payments. In some cases, the response from the bank or payment processor does not occur immediately with the request for funds. In that case, you can allow BRM to post unconfirmed payments.
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. See BRM Opcode Guide.
This document describes how to configure payment methods for Oracle Communications Billing and Revenue Management (BRM) payments.

See also:
- BRM Concepts
- List of Payment Processing Features

To create a custom payment method, see BRM Opcode Guide.

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 (/billinfo object) in an account, but you can use only one payment method per bill unit.

---

You can configure custom payment methods. To do so, you need to do the following:

- Update the /config/payment object.
- Modify the PCM_OP_CUST_POL_PREP_PAYINFO policy opcode to validate the custom payment method. For example, add code for your custom payment method everywhere the opcode checks the various payment methods.

Default Payment Methods

By default, BRM supports the following payment methods:

- **Cash, check, and postal order payment methods.** You allocate cash, check, and postal order payments by using Payment Tool. Customers of this type usually use the Invoice payment method.

- **Credit card payment method.** Credit card payments are BRM-initiated. Because some credit card payments are made automatically, accounts that pay bills by these methods should always use the balance forward accounting 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.
When a credit card payment is made, BRM returns a confirmation number that the customer can use to identify the payment. See the document about payments in *BRM Opcode Guide*.

- **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. Direct debit payments are BRM-initiated.

In the US and Canada, 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.

In Europe, BRM supports Single Euro Payments Area (SEPA) Direct Debit (SDD) and SEPA Credit Transfer (SCT) as BRM-initiated payments. For more information about SEPA payment processing, see "Implementing SEPA Payment Processing".

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

- **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.

- **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. When you run billing, no collection process is performed on prepaid balances because they are paid in advance of billing.

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.)

- **Nonpaying payment method.** Nonpaying bill units are child bill units; their bill is paid by the account that owns their paying parent bill unit. If an account has two bill units (and thus two bills), one paying and one nonpaying, the account pays one bill and the account that owns the nonpaying bill unit’s paying parent pays the other. See “About Account Groups” in *BRM Concepts*.

- **Undefined payment method.** Accounts with the Undefined payment method never receive a payment request. You typically use undefined accounts for free trial offers. Creating an undefined account enables 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.

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.

Because an account with a payment method of Undefined never pays a bill, you need to set the credit limit to **Unlimited**.

Revenue generated from undefined accounts can be recorded as general ledger (G/L) data.
- Voucher payment method. When a customer buys a voucher, either a CSR or the customer enters the voucher ID & PIN and BRM validates the voucher and transfers its prepaid balances to the specified account balance.

Voucher payments cannot be handled by the BRM-initiated payment process or by Payment Tool. To provide voucher payments for your customers, you must have Voucher Manager and Voucher Administration Center installed. For more information, see “About Managing Voucher Inventory” in *BRM Telco Integration*.

- **Wire transfer payment method.** Wire transfers include any transfer of money from a customer’s bank account to your company or to your company’s payment processor through an 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.

Payment methods are stored in the `config/payment` object and defined in the `BRM_home/include/ pin_pynt.h` file. Each payment method is associated with an element ID.

**Important:** To avoid conflicts with payment IDs reserved by BRM, assign custom payment methods an element ID greater than 10099.

*Table 13–1* list the standard payment methods and element IDs.

<table>
<thead>
<tr>
<th>Payment method</th>
<th>Element ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN_PAY_TYPE_UNDEFINED</td>
<td>0</td>
</tr>
<tr>
<td>Used during account creation.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_PREPAID</td>
<td>10000</td>
</tr>
<tr>
<td>Used to keep negative balances.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_INVOICE</td>
<td>10001</td>
</tr>
<tr>
<td>Used for monthly invoices.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_DEBIT</td>
<td>10002</td>
</tr>
<tr>
<td>Used for checking account debit.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_CC</td>
<td>10003</td>
</tr>
<tr>
<td>Used for credit cards.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_DD</td>
<td>10005</td>
</tr>
<tr>
<td>Used for US/Canadian direct debits.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_SMARTC</td>
<td>10006</td>
</tr>
<tr>
<td>Used for smartcards.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_SUBORD</td>
<td>10007</td>
</tr>
<tr>
<td>Used to roll up balances to the parent account.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_BETA</td>
<td>10008</td>
</tr>
<tr>
<td>For use by beta testers only. Billing utilities ignore this.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_INTERNAL</td>
<td>10009</td>
</tr>
<tr>
<td>Used for internal employees. Used the same way as guest accounts.</td>
<td></td>
</tr>
</tbody>
</table>
### Table 13–1  (Cont.) Payment Methods and Element IDs

<table>
<thead>
<tr>
<th>Payment method</th>
<th>Element ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN_PAY_TYPE_GUEST</td>
<td>10010</td>
</tr>
<tr>
<td>Used for guest accounts. It is not charged, but credit limits apply.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_CASH</td>
<td>10011</td>
</tr>
<tr>
<td>Used for cash.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_CHECK</td>
<td>10012</td>
</tr>
<tr>
<td>Used for personal bank checks.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_WTRANSFER</td>
<td>10013</td>
</tr>
<tr>
<td>Used for wire transfers.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_PAYORDER</td>
<td>10014</td>
</tr>
<tr>
<td>Used for inter-bank payment orders.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_POSTALORDER</td>
<td>10015</td>
</tr>
<tr>
<td>Used for postal payment orders.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_VOUCHER</td>
<td>10016</td>
</tr>
<tr>
<td>Used for payment vouchers.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_FAILED</td>
<td>10017</td>
</tr>
<tr>
<td>Used for unconfirmed payments that failed.</td>
<td></td>
</tr>
<tr>
<td>PIN_PAY_TYPE_SEPA</td>
<td>10018</td>
</tr>
<tr>
<td>Used for SEPA payments.</td>
<td></td>
</tr>
</tbody>
</table>
This document describes how to customize payment details displayed in Oracle Communications Billing and Revenue Management (BRM) client applications.

Topics in this document:

- Customizing Payment Details Displayed in BRM Client Tools
- About the Default /config/paymenttool Object
- Rules for Modifying Payment and Reversal Fields
- Creating an Object Definition for a New Payment or Reversal Event
- Changing the Order of Columns in Payment Tool
- Customizing the Date Format for Payment Center
- Improved Performance of Searches for Payment Events in Payment Center

See also:

- BRM Concepts
- List of Payment Processing Features

**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.
About the Default /config/paymenttool Object

The following is the default config/paymenttool object as defined in init_objects.source.

```
! 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_FIELD_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,};
field * * PIN_FLD_BATCH_TYPE{
    type = PIN_FLDT_INTINT,
    perms = MRW,
};
```

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, follow these rules:

- 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.

---

**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.
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 and 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.

- 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 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.

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.

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 must customize Payment Tool. To do this, you must change the /event/billing/payment, /event/billing/reversal, and /config/paymenttool objects.

To change the order of the columns, you must 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
```
Changing the Order of Columns in Payment Tool

To change the order of the columns in Payment Tool, you must change the order of each array. In the following example, the columns are in the order:

- **Credit Card No.**
- **Reason Code**
- **Chargeback Date**

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 NT32 [0] 1
```

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` storable class.

1. Use the PCM_OP_WRITE_FLDS opcode to add the direct debit vendor details to the `/config/paymenttool` storable class. Call the opcode using flag 32. For example:
Changing the Order of Columns in Payment Tool

1. 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 \texttt{MM/dd/yyyy} format. If the locale is New Zealand English, Payment Tool uses \texttt{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. Stop and restart the CM.

\textbf{Note:} All changes you make in /config/paymenttool are reflected in the Customer Center UI when you restart BRM.
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.

---

**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 Payment Center is installed.

   **Note**: 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:

```plaintext
paymentsearch.stepsize=100
```

3. Save the file.
Implementing SEPA Payment Processing

This document describes the Oracle Communications Billing and Revenue Management (BRM) Single Euro Payments Area (SEPA) payment processing.

See also:
- **BRM Concepts**
- **List of Payment Processing Features**

### About SEPA Payments

SEPA payments are electronic payment transfers between bank accounts in the euro countries that participate in SEPA.

SEPA defines a common set of standards and rules for any organization or individual making or receiving payments in euro. With SEPA, all bank accounts are uniquely identified by the International Bank Account Number (IBAN), and the banks related to the accounts are uniquely identified by the Business Identifier Code (BIC). These standards improve the ability of consumers to transfer money, for example, from the home bank account to an account in another country that participates in SEPA.

**Note:** BRM supports the SEPA specifications in the SEPA Rulebook Version 7.0.

SEPA defines two payment schemes: SEPA Direct Debit and SEPA Credit Transfer. Both SEPA Direct Debit and SEPA Credit Transfer are supported as BRM-initiated payments.

**SEPA Direct Debit** is a payment transfer that is initiated by the service provider for automated payments from the customer’s bank account. This type of payment is commonly used for recurring payments such as automated payments for a monthly subscription charge (can also be used for one-time payments) and requires a pre-authorization (mandate) from the customer.

**SEPA Credit Transfer** is a payment transfer that is initiated by the service provider to transfer money from the service provider’s bank account to the customer’s bank account. SEPA Credit Transfer is used to give refunds to customers. The service provider must provide the customer’s IBAN and the customer’s bank’s BIC to initiate the credit transfer.
About Specifying SEPA Payment Information During Account Creation

When you create an account (or when an existing customer purchases a new service) and the customer wants to pay by SEPA Direct Debit, specify SEPA as the payment method.

In addition to the customer’s name and address information, your customer must provide a mandate, a pre-authorization form that is signed by your customer, to debit the customer’s bank account automatically through SEPA Direct Debit.

SEPA Direct Debit and SEPA Credit Transfer payments are allowed in euro only. When you create an account, the account’s primary currency must be euro.

About Registering the Mandate for SEPA Direct Debit Payments

To pay for services by SEPA Direct Debit, your customer must first fill out and sign a mandate (provided by you) to authorize automatic payments from the customer’s bank account.

SEPA requires the service provider to send this mandate information with each collection of the SEPA Direct Debit payment. You are also required to retain the mandate (throughout the period when the customer is using SEPA Direct Debit and according to the national legal requirements and its Terms and Conditions) along with any amendments or information concerning its cancellation or lapse with the service provider’s bank.

The mandate must include the following information:

■ Your customer’s name and address
■ Your customer’s IBAN
■ Your customer’s bank’s BIC
■ Your business name and address
■ Your creditor identification number
■ Type of mandate (recurrent or one-off)
■ Your customer’s signature

Your customer service representative (CSR) receives the signed mandate and enters the data into the BRM system by using Billing Care.

A mandate is identified by the unique mandate reference (UMR) number. If a unique mandate reference number is not provided, BRM automatically generates one for the mandate.

In BRM, a mandate is associated with a bill unit and is valid for collection of the payment for this bill unit. If your customer has multiple services associated with different bill units and wants to pay for the different services by SEPA Direct Debit, your customer must provide separate mandates for the collection of payments for each service. If the same mandate is associated with multiple services, it is assumed that your customer has authorized collection of payment for all the services using a single mandate.

For information on the requirements for retaining the paper mandate and any amendments to it, refer to the SEPA Direct Debit Rulebook.

About the Different Types of Mandates

Mandates are of two types: recurrent and one-off.
A **recurrent mandate** is used to collect multiple bill payments for a bill unit; for example, to collect a recurring monthly service fee. If a recurrent mandate is not used within a 36-month period, it is considered expired; BRM automatically sets the mandate status to Expired.

A **one-off mandate** is used to collect only one bill payment for a bill unit. For example, your customer pays bills regularly by check or credit card but wants to pay a bill by SEPA Direct Debit. After collection of the one bill payment, the mandate cannot be used to collect other bill payments; BRM automatically sets the mandate status to Expired.

You cannot re-activate a mandate that is expired. A new mandate is required to process any SEPA payment requests.

**Managing Customer’s SEPA Payment Information**

Use Billing Care to change or delete your customer’s SEPA payment information.

You can do the following:

- **Change the customer’s payment method**
  
  If your customer wants to change from SEPA to a different payment method, you need to register new payment information and associate the customer’s services with the new payment information. Your customer’s existing SEPA payment method in the BRM database is not changed.

  If your customer wants to change from a different payment method to SEPA, you need to first register the SEPA payment information. For instance, if your customer is currently paying by credit card and wants to pay by SEPA Direct Debit instead, register new payment information that includes the SEPA-related information such as the IBAN, BIC, and the mandate information. Your customer’s existing payment information in the BRM database is not changed.

- **Delete the payment method**
  
  When you delete a SEPA payment method, BRM also cancels the mandate that is associated with the payment method and the mandate cannot be used with any future payment requests; a new mandate is required.

  You cannot delete the SEPA payment method if it is associated with a bill unit.

  If the SEPA payment method is associated with a payment request that is pending, BRM cancels the mandate only for future payment requests.

- **Change the mandate information**
  
  To update the creditor information in a mandate, you update the creditor configuration object. See **BRM Opcode Guide** for more information.

  BRM stores the new mandate information and also keeps a record of the information that is amended and sends both the new and amended information to the bank with the next SEPA payment collection.

**Loading Your Creditor Information into the BRM Database**

---

**Note:** To update creditor information, see **BRM Opcode Guide**.

---
Loading Your Creditor Information into the BRM Database

Your creditor information includes your business name and address and the creditor identification number. You load the creditor information into the creditor configuration objects (/config/creditor) in the BRM database (see "Loading Your Creditor Information into the BRM Database").

During account creation, Billing Care retrieves your creditor configuration information from the BRM database.

Creditor information is stored in the /config/creditor object in the BRM database.

To set up and load creditor information:

1. Open the BRM_home/sys/data/config/config_creditor.xml file in a text editor, where BRM_home is the directory in which the BRM server software is installed.

2. In the CREDITOR_INFO child element, provide the values listed in Table 15–1.

3. Save and close the file.

4. Run the following command, which loads the contents of the file into the /config/creditor object:

   \[ BRM_home/apps/load_config/load_config -v config_creditor.xml \]

   The load_config utility validates the contents using the config_creditor.xsd file before loading the data.

   See "load_config" in BRM Developer’s Guide for more information about the utility’s syntax and parameters.

5. Read the object by using the robj command with the testnap utility or by using Object Browser to verify that the creditor configurations are loaded.

   See "Using testnap" in BRM Developer’s Guide for general instructions on using the testnap utility.

   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).

---

Table 15–1  Elements in the CREDITOR_INFO Child Element

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDRESS</td>
<td>Your business street address</td>
</tr>
<tr>
<td>BIC</td>
<td>Your Business Identifier Code</td>
</tr>
<tr>
<td>CITY</td>
<td>The city where your business is located</td>
</tr>
<tr>
<td>COUNTRY</td>
<td>The country where your business is located</td>
</tr>
<tr>
<td>CREDITOR_ID</td>
<td>Your creditor identification number</td>
</tr>
<tr>
<td>CURRENCY</td>
<td>Your currency</td>
</tr>
<tr>
<td>IBAN</td>
<td>Your International Bank Account Number</td>
</tr>
<tr>
<td>NAME</td>
<td>Your business name</td>
</tr>
<tr>
<td>REF_PARTY</td>
<td>The name of your reference party</td>
</tr>
<tr>
<td>REF_PARTY_ID_CODE</td>
<td>The identification code of your reference party</td>
</tr>
<tr>
<td>ZIP</td>
<td>The postal code where your business is located</td>
</tr>
</tbody>
</table>
You can use `load_config` utility to add new creditor configuration data; it does not overwrite any existing data in the configuration objects. However, to update or delete a creditor configuration object, you need to use opcodes (see BRM Opcode Guide).

**Processing SEPA Payments**

Processing of SEPA payments includes these tasks:

- Creating the payment requests in BRM. See "Creating SEPA Direct Debit Payment Requests" and "Creating SEPA Credit Transfer Payment Requests".

- Generating the SEPA request files. See "Generating SEPA Request XML Files".

- Collecting the payments. See "Sending the SEPA Request XML Files to Your Bank to Collect Payment".

- Handling the failed payments. See "Processing SEPA Response XML Files to Handle Failed Payment Transactions".

**Creating SEPA Direct Debit Payment Requests**

You run the `pin_collect` utility to create the SEPA Direct Debit payment requests in the BRM database.

The SEPA Direct Debit payment requests record the customer's payment details, such as the amount due and mandate information, and the payment transaction ID.

The `pin_collect` utility retrieves the pending bills for accounts that use the SEPA payment method and calculates the amount due. For each bill unit, it records the payment details in the payment request (`/sepa/dd`) and sets the payment request status to Pending.

The `pin_collect` utility does not create a payment request if the mandate for the bill unit is expired. To collect the payment, your customer has to provide a valid mandate or use another payment method.

SEPA Direct Debit payments are applied to the accounts at the time payment requests are created (before payment requests are sent to the bank). If your bank is unable to collect the payment from your customer’s bank, you reverse the payment recorded in BRM using the `pin_sepa` utility (see "Processing SEPA Response XML Files to Handle Failed Payment Transactions").

For more information about the `pin_collect` utility, see "pin_collect".

**Creating SEPA Credit Transfer Payment Requests**

You run the `pin_mass_refund` and the `pin_refund` utilities to create SEPA Credit Transfer refund requests in the BRM database.

The SEPA Credit Transfer payment requests record the customer's payment details, such as the refund amount and the payment transaction ID.

The `pin_mass_refund` utility aggregates the credit balance for each bill unit for each account and generates refund items for the aggregated credit amount.

The `pin_refund` utility retrieves the refund items for the accounts that use the SEPA payment method. For each bill unit, it records the payment details in the refund request (`/sepa/ct`) and sets the refund request status to Pending. You run the `pin_refund` utility after running the `pin_mass_refund` utility.
SEPA Credit Transfer refunds are applied to the accounts at the time refund requests are created (before refund requests are sent to the bank). If your bank is unable to process the refund, you reverse the refund recorded in BRM using the `pin_sepa` utility (see "Processing SEPA Response XML Files to Handle Failed Payment Transactions").

For more information about the `pin_mass_refund` and `pin_refund` utilities, see "About Refunds" in BRM Managing Accounts Receivable.

### Generating SEPA Request XML Files

You run the `pin_sepa` utility to generate the SEPA request XML files (see "`pin_sepa`" for more information about the utility syntax).

Before running `pin_sepa`, configure the utility to provide the information it requires for generating the SEPA request XML files (see "Configuring the `pin_sepa` Utility for Generating and Processing SEPA XML Files").

The `pin_sepa` utility extracts payment details from the SEPA Direct Debit and SEPA Credit Transfer payment requests (created by the `pin_collect` and `pin_refund` utilities), which are in Pending status, from the BRM database into SEPA request XML files. All the payment transactions belonging to the same creditor are grouped in one file. The number of payment transactions in a file is configurable by using the `infranet.threadpool.fetchsize` entry in the `Infranet.properties` file for `pin_sepa`.

You must manually send the SEPA request XML files to your bank for collection of the payments (see "Sending the SEPA Request XML Files to Your Bank to Collect Payment").

After the SEPA request XML files are generated, BRM considers the payment as successful and changes the status of the payment requests to Requested. The payment requests remain in Requested status unless the payment is reversed for any reason.

---

**Important:**

- The SEPA request XML files cannot be regenerated. You must ensure the files are protected from accidental loss or corruption.
- The SEPA request XML files contain sensitive customer data. You must ensure the files are protected from unauthorized access.

For more information on security, see BRM Security Guide.

---

By default, the `pin_sepa` utility is not included in the `pin_bill_day` billing script. You can either add it to the daily billing script or run it separately; however, Oracle recommends to run `pin_sepa` daily for SEPA payment collection. You can run the `pin_sepa` utility manually or as a `cron` job that runs at specified times.

### Sending the SEPA Request XML Files to Your Bank to Collect Payment

The SEPA request XML files are stored in the directory that you specify in the `Infranet.properties` file until they are delivered to your bank for collection of payment. You must manually send the files to your bank or payment processing center: BRM does not send the files.

After sending the files, ensure the files were successfully delivered to your bank. Potential revenue loss can occur if the SEPA request XML files that are generated in BRM are not received by your bank for processing.
**Processing SEPA Response XML Files to Handle Failed Payment Transactions**

Your bank sends back the SEPA response XML files with the payment transactions that are rejected. Your bank may reject a SEPA payment or refund request for reasons such as the following:

- The payment or refund request contains an invalid IBAN or BIC.
- The payment request contains an invalid or incorrect mandate.
- The customer’s bank account has insufficient funds to process the payment.

SEPA Direct Debit payments and SEPA Credit Transfer refunds are applied to the accounts in BRM at the time payment requests are created. Therefore, any payment transactions that are rejected by the bank needs to be reversed in BRM.

The SEPA response XML file indicates a status at the group level, payment-information level, and transaction level.

If the group-level status is Reject, all the payment transactions in the response file are rejected.

If the payment-information-level status is Reject, all the payment transactions in the payment information are rejected.

If the transaction-level status is Reject, only the payment for this transaction is rejected.

You run `pin_sepa` utility to process the rejected payments in the SEPA response file (see "pin_sepa" for more information about the utility syntax). The utility automatically initiates the payment reversal in BRM. Using the payment transaction ID, BRM locates the corresponding SEPA payment request in the database and changes the status of the payment request to Reject.

**Reversing an Erroneous Payment Collection**

An erroneous or duplicate payment occurs when your customer is billed twice for the same charge. The payment is recorded in BRM, and the payment transaction is successfully completed by the bank.

Unlike a payment reversal that occurs when a payment is rejected by the bank, duplicate payment reversals are not initiated by BRM.

After the payment reversal requests are created, you run the `pin_sepa` utility to generate the SEPA reversal request XML files (see "pin_sepa" for more information about the utility syntax). The `pin_sepa` utility extracts the payment details from the payment reversal requests, which are in Pending status, from the BRM database into SEPA reversal request XML files.

After the SEPA reversal request XML files are generated, BRM considers the payment reversal as successful and changes the status of the payment reversal requests to Requested.

You must manually send the SEPA reversal request files to your bank to reverse the charges from the customer’s bank account.

**Using SEPA XML Messages to Exchange Customer’s Payment Information**

For SEPA compliance, banks are required to use SEPA ISO20022 XML messages to exchange customer’s payment information.

BRM supports the following ISO20022 XML messages:
For SEPA Credit Transfer:

- Customer Credit Transfer Initiation (pain.001.001.03): This message transports the customer-to-bank credit transfer information sent by the customer (originator) to the customer’s bank.
- Customer Payment Status Report (pain.002.001.03): This message transports the credit transfer reject instruction between the bank and its remitting customer.

For SEPA Direct Debit:

- Customer Direct Debit Initiation (pain.008.001.02): This message transports the direct debit collection instruction from the creditor to the creditor’s bank.
- Customer to Bank Payment Reversal (pain.007.001.02): This message transports the customer-to-bank reversal instruction for a collection sent by the creditor to the creditor’s bank.
- Bank to Customer Payment Status Report (pain.002.001.03): This message transports the direct debit reject instruction between the bank and its remitting customer.

You and your bank must use this version of ISO20022 XML message to ensure the messages sent and received are interpreted correctly.

The SEPA request and response XML files must comply with the XML schema definitions (XSD) that are provided in BRM.

Before processing a SEPA response file, BRM validates the contents using the XSD. BRM cannot process a response file that uses a different XSD.

**Configuring the pin_sepa Utility for Generating and Processing SEPA XML Files**

You use the `pin_sepa` utility to generate the SEPA request XML files and to process SEPA response XML files.

Before running the `pin_sepa` utility, you must edit the utility’s `Infranet.properties` file to include the information that it requires to generate and process SEPA request and response XML files.

To configure the `Infranet.properties` file:

1. Open the `BRM_home/apps/pin_sepa/Infranet.properties` file in a text editor.
2. Provide the values listed in Table 15–2.

The `Infranet.properties` file for the `pin_sepa` utility includes standard configuration entries. See “Using Configuration Files to Connect and Configure Components” in *BRM System Administrator’s Guide* for more information.

### Table 15–2  pin_sepa Infranet.properties Configuration Entries

<table>
<thead>
<tr>
<th>Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>infranet.connection</td>
<td>Specifies the connection information to connect to the BRM database.</td>
</tr>
<tr>
<td>infranet.login.type</td>
<td>Specifies if a login name and password is required to connect to the BRM database. The default is 1.</td>
</tr>
</tbody>
</table>
### Table 15–2 (Cont.) pin_sepa Infranet.properties Configuration Entries

<table>
<thead>
<tr>
<th>Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>infranet.log.level</td>
<td>Specifies the error reporting level. The default is 1.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>infranet.log.file</td>
<td>Specifies the file name used to log errors. The default is pin_sepa.pinlog</td>
</tr>
<tr>
<td>infranet.threadpool.size</td>
<td>Specifies the number of threads. The default is 3.</td>
</tr>
<tr>
<td>infranet.threadpool.maxsize</td>
<td>Specifies the maximum number of threads. The default is 5.</td>
</tr>
<tr>
<td>infranet.threadpool.fetchsize</td>
<td>Specifies the number of records fetched from the BRM database and assigned to a thread at one point of time. This entry also controls the maximum number of payment transactions that can be in a SEPA request XML file. The default is 100.</td>
</tr>
<tr>
<td>infranet.sepa_dd_req_dir.path</td>
<td>Specifies the directory path to the SEPA Direct Debit request XML files. The default directory is BRM_home/apps/pin_sepa/sepa_dd. If you change the default directory path, you must create the new directory where you want to store the files before running pin_sepa.</td>
</tr>
<tr>
<td>infranet.sepa_ct_req_dir.path</td>
<td>Specifies the directory path to the SEPA Credit Transfer request XML files. The default directory is BRM_home/apps/pin_sepa/sepa_ct. If you change the default directory path, you must create the new directory where you want to store the files before running pin_sepa.</td>
</tr>
<tr>
<td>infranet.sepa_rev_req_dir.path</td>
<td>Specifies the directory path to the SEPA Direct Debit reversal request XML files. The default directory is BRM_home/apps/pin_sepa/sepa_rev. If you change the default directory path, you must create the new directory where you want to store the files before running pin_sepa.</td>
</tr>
<tr>
<td>infranet.sepa_resp_dir.path</td>
<td>Specifies the directory path to the SEPA Direct Debit, Credit Transfer, and Direct Debit reversal response XML files. The default directory is BRM_home/apps/pin_sepa/sepa_resp/input. If you change the default directory path, you must create the new directory where you want to store the files before running pin_sepa. The utility reads all the files in the directory for processing. Hence, it recommended to store only response XML files in this directory.</td>
</tr>
<tr>
<td>infranet.sepa.sddrequest.ReqdColltnDt.pattern</td>
<td>Specifies the date pattern for the SEPA Direct Debit request.</td>
</tr>
<tr>
<td>infranet.sepa.sddrequest.ReqdColltnDt.value</td>
<td>Specifies the date on which to collect the money from the customer.</td>
</tr>
<tr>
<td>infranet.sepa.sddrequest.InitgPtNm</td>
<td>Specifies the name of the party initiating the SEPA Direct Debit request.</td>
</tr>
</tbody>
</table>
Table 15–2  (Cont.) pin_sepa Infranet.properties Configuration Entries

<table>
<thead>
<tr>
<th>Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>infranet.sepa.sddrequest.InitgPt.OrgId</td>
<td>Specifies the ID of the party initiating the SEPA Direct Debit request.</td>
</tr>
<tr>
<td>infranet.sepa.sddrequest.PmtInf.PmtMtd</td>
<td>Specifies the SEPA Direct Debit payment method. This entry must be set to DD.</td>
</tr>
<tr>
<td>infranet.sepa.sddrequest.InstrPrty</td>
<td>Specifies the instruction priority for the SEPA Direct Debit request. The default is NORM.</td>
</tr>
<tr>
<td>infranet.sepa.sddrequest.ChrgBr</td>
<td>Specifies the party who will pay for the charges. The default is SLEV.</td>
</tr>
<tr>
<td></td>
<td>According to the SEPA Rulebook, the only value allowed for this entry is SLEV.</td>
</tr>
<tr>
<td>infranet.sepa.sddrequest.PmtTpInf.LclInstrm</td>
<td>Specifies the Local instrument code for SEPA Direct Debit request. The default is CORE.</td>
</tr>
<tr>
<td></td>
<td>■ CORE: Core Scheme</td>
</tr>
<tr>
<td></td>
<td>■ B2B: Business to Business Scheme</td>
</tr>
<tr>
<td>infranet.sepa.sddrequest.PmtTpInf.SvcLvl</td>
<td>Specifies the service level for the SEPA Direct Debit request. The default is SEPA.</td>
</tr>
<tr>
<td>infranet.sepa.sctrequest.PmtInf.PmtMtd</td>
<td>Specifies the SEPA Credit Transfer payment method. This entry must be set to TRF.</td>
</tr>
<tr>
<td>infranet.sepa.sctrequest.ReqdExctnDt.pattern</td>
<td>Specifies the date pattern for the SEPA Credit Transfer request.</td>
</tr>
<tr>
<td>infranet.sepa.sctrequest.ReqdExctnDt.value</td>
<td>Specifies the date on which to credit the money to customer account.</td>
</tr>
<tr>
<td>infranet.sepa.sctrequest.InstrPrtty</td>
<td>Specifies the instruction priority for SEPA Credit Transfer request. The default is NORM.</td>
</tr>
<tr>
<td>infranet.sepa.sctrequest.ChrgBr</td>
<td>Specifies the party who will pay for the charges. The default is SLEV.</td>
</tr>
<tr>
<td></td>
<td>According to the SEPA Rulebook, the only value allowed for this entry is SLEV.</td>
</tr>
<tr>
<td>infranet.sepa.sctrequest.InitgPty.Nm</td>
<td>Specifies the name of the party initiating the SEPA Credit Transfer request.</td>
</tr>
<tr>
<td>infranet.sepa.sctrequest.InitgPty.OrgId</td>
<td>Specifies the ID of the party initiating the SEPA Credit Transfer request.</td>
</tr>
<tr>
<td>infranet.sepa.sctrequest.PmtTpInf.LclInstrm</td>
<td>Specifies the Local instrument code for SEPA Credit Transfer request. The default is CORE.</td>
</tr>
<tr>
<td></td>
<td>■ CORE: Core Scheme</td>
</tr>
<tr>
<td></td>
<td>■ B2B: Business to Business Scheme</td>
</tr>
<tr>
<td>infranet.sepa.sctrequest.PmtTpInf.SvcLvl</td>
<td>Specifies the service level for the SEPA Credit Transfer request. The default is SEPA.</td>
</tr>
<tr>
<td>infranet.sepa.sddreversal.InitgPty.Nm</td>
<td>Specifies the name of the party initiating the SEPA Direct Debit Reversal request.</td>
</tr>
<tr>
<td>infranet.sepa.sddreversal.InitgPty.OrgId</td>
<td>Specifies the ID of the party initiating the SEPA Direct Debit Reversal request.</td>
</tr>
</tbody>
</table>

3. Save and close the file.
This document describes how to implement payment fees in Oracle Communications Billing and Revenue Management (BRM).

See also:
- BRM Concepts
- Configuring Payment Incentives
- List of Payment Processing Features

For information on customizing payment fees, see BRM Opcode Guide.

About Payment Fees

Payment fees are one-time fees that can be charged for failed payments. For example, if a 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.

Payment fees can be applied to payments that are processed by using BRM-initiated payment processing, such as credit-card payments, or externally-initiated payment processing, such as checks.

BRM applies the balance impact of the payment fee event to the default balance group of the bill unit (/billinfo object). If a customer receives a payment fee in error, you can perform a balance adjustment to remove the fee.

When payments are processed, they are given a status; for example, successful or failed. When a payment is given a failed payment status, BRM creates the following events:

- A failed payment fee event (/event/billing/fee/failed_payment).
- A payment event for the failed payment (/event/billing/payment/failed).

You can use data from both of those events to charge customers for payment fees, and to customize how to implement payment fees.

To charge customers for payment fees, you create a charge offer that uses a charge based on the failed payment fee event (/event/billing/fee/failed_payment). For example, the failed payment fee event includes the amount of the original payment that failed. You can create a charge offer that creates a payment fee if the payment amount was over a specified amount.

To customize payment fees, you use the data in the payment event (/event/billing/payment/failed). For example, this event includes a reason ID that records the reason that the payment failed. You can use this reason, in conjunction with the PCM_OP_PYMT_POL_APPLY_FEES policy opcode, to customize how
Creating a Payment Fee Charge Offer

To create a charge offer to charge for payment fees:

1. In the PDC service-event map, create a Payment Failure event in the Account events:
   
   ![Service-Event Map]

   - **Billing Time Discount**
   - **Cycle Fold**
   - **Payment Failure**
     - **Amount**
     - **Purchase Fee**
     - **Admin Client**
     - **Wireless Data**

2. Create one or more RUMs based on fields in the failed payment fee event. The relevant fields are:
   - **Amount of original payment.** For example, you could create a payment fee for failed payments over $50.00.
   - **Customer segment.** For example, you could create customer segments such as *early bill payer* and *delinquent bill payer*. You could create different fees for each segment, based on their payment patterns. See *BRM Managing Customers* for information about customer segments.
   - **Payment method.** For example, you could create different payment fees for credit-card payments and cash payments.
   - **Channel ID.** See "Configuring Payment Channels."

   To charge a payment fee based on multiple attributes, use a charge selector. For example, you could create a payment fee based on failed credit-card payments over $50.00.
3. Create a charge offer based on the Payment Failure event. The charge has these attributes:
   - Use **Subscription** for the charge offer type.
   - Use **One Time** for the charge category.
   - Use **Payment Failure** for the charge type.
   - Use a debit currency balance impact in the charge to charge a fee when the Payment Failure event occurs.

If an account owns a payment fee charge offer, you cannot exempt the account from receiving payment fees. However, you can create a discount that grants the same amount as the fee.
This document describes how to implement payment incentives in Oracle Communications Billing and Revenue Management (BRM).

See also:

- BRM Concepts
- Configuring Payment Fees
- List of Payment Processing Features

For information on customizing payment incentives, see BRM Opcode Guide.

### About Payment Incentives

A payment incentive is a reward for customers who pay their bills early and in full. For example, you can award 20 minutes or provide a 5% reduction in the monthly bill amount.

You create payment incentives by creating charge offers that grant the incentives. The payment incentives are based on the fields defined in the /event/billing/incentive event. For example, you can create an incentive based on the amount of the current or previous bill, the payment method, customer segments, and payment channels.

A single payment incentive can impact multiple balances; for example, both minutes and the amount due for a cycle forward fee. Customers might be eligible for multiple payment incentives depending on which charge offers they purchase and whether any payment incentive system charge offers are valid for their account.

You can customize payment incentives by editing the PCM_OP_PYMT_POL_GRANT_INCENTIVE policy opcode and extending the /event/billing/incentive storable class. See BRM Opcode Guide.

In addition to creating payment incentive charge offers, you must enable payment incentives in BRM. See “Enabling BRM for Payment Incentives.”

### How BRM Creates Payment Incentives

BRM determines if an account is eligible for a payment incentive by comparing the time that the customer paid their previous bill, and the time when the payment was due for the previous bill. If the payment was paid before it was due, the account is eligible for a payment incentive. The incentive is granted the next time that billing is run. Figure 17–1 shows how a payment incentive is granted.
How Payment Reversals Affect Payment Incentives

If the account qualifies for a payment incentive, BRM adds a trigger to the bill unit (/billinfo object). This is known as provisioning the payment incentive. During the next billing run, BRM checks the bill unit for this trigger. If the trigger is present, indicating that the payment incentive is provisioned, BRM uses the customer’s purchased charge offer to calculate the payment incentive. The payment incentive is applied to the default balance group of the bill unit associated with the bill.

Note: The current bill total indicates the current bill amount. This does not include the debits and credits from the previous bill.

By default, 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.

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

■ On-purchase 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.

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 must 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 must perform the adjustment manually as a balance adjustment. You can create a custom application to find accounts that need a payment event reversal. For information, see the chapter about payments in *BRM Opcode Guide*.

**Enabling BRM for Payment Incentives**

To enable payment incentives, run the `pin_bus_params` utility to change the `PaymentIncentive` business parameter. For information about this utility, see *BRM Developer’s Guide*.

To enable payment incentives:

1. Go to `BRM_home/sys/data/config`.
2. Create an XML file from the `/config/business_params` object:
   ```
   pin_bus_params -r BusParamsAR bus_params_AR.xml
   ```
3. In the file, change `disabled` to `enabled`:
   ```
   <PaymentIncentive>enabled</PaymentIncentive>
   ```
4. Save the file as `bus_params_AR.xml`.
5. Load the XML file into the BRM database:
   ```
   pin_bus_params bus_params_AR.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see *BRM System Administrator’s Guide*.

**Creating a Payment Incentive Charge Offer**

To create a charge offer to grant a payment incentive:

1. In the PDC service-event map, create a Payment Incentive event in the Account events:
2. Create one or more RUMs based on fields in the payment incentive event. The relevant fields are:
   - **Current bill amount or previous bill amount** For example, you can create a balance impact that credits a percentage of the total bill.
   - **Customer segment.** For example, you could create customer segments such as *early bill payer* and *delinquent bill payer*. You could create different payment incentives for each segment, based on their payment patterns. See *BRM Managing Customers* for information about customer segments.
   - **Payment method.** For example, you could create different payment incentives for credit-card payments and cash payments.
   - **Channel ID.** See "Configuring Payment Channels."

To grant a payment fee based on multiple attributes, use a charge selector.

3. Create a charge offer based on the Payment Incentive event. The charge has these attributes:
   - Use **Subscription** for the charge offer type.
   - Use **One Time** for the charge category.
   - Use **Payment Incentive** for the charge type.
   - Create a balance impact to grant the incentive when the Payment Incentive event occurs.
This document describes how to implement the Oracle Communications Billing and Revenue Management (BRM) top-up features.

See also:

- **BRM Concepts**
- **List of Payment Processing Features**

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".

**Important**: Most top-up implementation tasks require a custom application. See **BRM Opcode Guide**.

### About Standard Top-Ups

A *standard top-up* is a top-up that a customer makes to his or her own account.

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.

  Manual top-ups can occur at any time and can be performed on any account. They can be used to add assets to credit balances or to debit balances.

- **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.

  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.

  Customers can use the following payment methods for standard top-ups:

  - **Credit card** or **direct debit** (manual and automatic standard top-ups)
  - **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.

A voucher can be used to top up one or more balances in a specified balance group (you cannot allocate a voucher’s balances to multiple balance groups). The balances can include one currency balance and an unlimited number of noncurrency balances. Top-ups for currency balances are added to the existing currency sub-balance, which maintains its original validity period. Top-ups for noncurrency balances are added to sub-balances according to their validity period.

Sponsored top-ups cannot be made between the following accounts:

- Accounts with different primary currencies
- Accounts in different database schemas in a BRM multischema system

To implement standard top-ups, see *BRM Opcode Guide*.

### About Taxes Applied During Voucher Top-Ups

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.

### Reversing Voucher Top-Ups

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 assets cannot be reapplied to another account or balance group.

If a voucher has only noncurrency balances, 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.

If a voucher has currency and noncurrency balances, 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.

### About Vouchers Having Noncurrency Balances with a Positive Impact

By default, when you apply a voucher with noncurrency balances to an account, a negative balance impact is applied to the account balance. For example, if a voucher grants 30 minutes, a balance impact of -30 is applied to the customer’s account balance. As the customer uses the minutes, the account balance approaches 0. For example, if the customer uses 20 of the 30 minutes, the account balance becomes -10. In this case, the noncurrency balance has a credit limit of 0 by default, or it can be changed to a negative value.

To have vouchers with noncurrency balances apply a *positive balance impact* to account balances, you must set the balance’s credit limit to a positive nonzero value. For example, you must set the minutes balance to +2.
To set the credit limit for noncurrency balances to a positive value, perform one of the following:

- Specify the credit limit in your package.
- Specify the credit limit in an account.

**About Sponsored Top-Ups**

A *sponsored top-up* is a top-up that is performed by transferring assets 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. Assets 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.

  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".

- **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".

  To implement standard top-ups, see *BRM Opcode Guide*.

**About Sponsored Top-Up Groups**

To top up other accounts, an account must own a sponsored top-up group. 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.

An account can be a member of only one sponsored top-up group at a time.

---

**Caution:** An account should be either a sponsored top-up group owner or member. It should not be both. If an account both owns sponsored top-up groups and belongs to one or more sponsored top-up groups, its accounts receivable (A/R) data may become inaccurate.

---

All accounts that belong to a sponsored top-up group have one of the member statuses shown in Table 18–1:
About Sponsored Top-Ups

<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. 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.

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 noncurrency debits that can accumulate in the owner’s paying balance group.

- **Credit limit of group balance**

  Each balance supported by a sponsored top-up group has a group top-up cap. The cap specifies the maximum amount of the balance 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.

  **Note:** Member accounts do not have individual sponsored top-up credit limits.

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 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 * * * BRM_home/bin/pin_balance_transfer &
```

Tracking Sponsored Top-Up Adjustments

To differentiate sponsored top-up adjustments (/event/billing/adjustment/account objects) from other types of account adjustments, the reasons.locale file includes the following reason codes and domain IDs:

- **Sponsored top-up debit reason code 4 and domain ID 1**

  ```
  DOMAIN = "Reason Codes-Debit Reasons" ;
  ```
About Sponsored Top-Ups

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 are in the **pin_pymt.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
  ```

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/sys/msg/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
```

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**.

**Canceling a Single Member’s Sponsored Top-Ups**

To stop sponsored top-ups temporarily, inactivate the top-up group members.
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.

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.

---

**Topping Up Accounts in Customer Center**

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.

**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:

```plaintext
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*.

### 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 do not 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*.

### 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. 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.

---

### Voucher Top-up Errors

*Table 18–2* 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><code>PIN_FLD_EXTENDED_INFO</code></td>
</tr>
<tr>
<td>Invalid voucher ID/PIN combination</td>
<td>ERR_NOT_FOUND</td>
<td><code>PIN_FLD_POID</code></td>
</tr>
<tr>
<td>Voucher has already been used or has expired</td>
<td>ERR_BAD_VALUE</td>
<td><code>PIN_FLD_STATE_ID</code></td>
</tr>
<tr>
<td>Error Message</td>
<td>Error Type</td>
<td>Field Name</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Voucher has expired</td>
<td>ERR_BAD_VALUE</td>
<td>PIN_FLD_EXPIRATION_T</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>
Managing Suspended Payments

This document describes how to handle suspended payments in Oracle Communications Billing and Revenue Management (BRM).

**Important:** Only externally initiated payments can be suspended.

See also:
- *BRM Concepts*
- *List of Payment Processing Features*

**About Suspending Payments**

A suspended payment is a payment that needs to be corrected. Payment Suspense Manager saves payments to a special account called a payment suspense account. You can then allocate them manually, or save them to an exception batch.

Payment suspense handles the following payment scenarios:

- Payments that fail the BRM validation process.
- Payments that were posted incorrectly to customer accounts.
- 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 (*billinfo* objects).

Managing suspended payments 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.
- 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.
- Allocate suspended payments to one or more accounts.
- Partially allocate a suspended payment so that an amount remains in the payment suspense account. 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. Payments can remain suspended indefinitely. You can move payments back and forth between customer accounts and the payment suspense account any number of times.

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.

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.

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 must 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.

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.

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.

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.

---

Figure 19–1 shows the steps involved in payment suspension and the basic operations they perform:
About Payment Suspense 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 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 19–2 shows how the payment suspense process works when you use Payment Tool to load payments:

**Figure 19–2 Payment Suspense Process Using Payment Tool**

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 and 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 19–3 shows how the payment suspense process works if you use a payment gateway to process payments.

**Figure 19–3  Payment Suspense Process Using Payment Gateway**

In this case, the payment gateway directs BRM to perform the validation and suspense tasks you would otherwise perform by using Payment Tool. After BRM determines payment status, it submits the payments to the BRM database and moves any suspended payments to the payment suspense account. Then you use Payment Center to review the contents of the payment suspense account, investigate the suspended payments, correct any problems, and submit the corrected payments to BRM.

When you suspend payments that were successfully submitted to customer accounts, you use Payment Center to undo the allocation of the payments in the customer accounts and save them to the payment suspense account. You can then investigate the suspended payments, correct any problems, and resubmit them to the correct accounts.

For detailed information on Payment Tool, see Payment Tool Help. For information on Payment Center, see Payment Center Help.

**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.

---

**Note:** When removing an unallocatable payment from suspense, only the active suspended payment is reversed. You cannot reverse any distributed payments or payments that have been reversed due to recycling. After you remove a payment as unallocatable, you cannot return it to the BRM system.

---

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 must 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.

---

**Payment Suspension Guidelines and Restrictions**

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.
- 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.

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 Billing Care.
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.

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**

To set up BRM for payment suspense, you complete three tasks:

- **Enabling Payment Suspense in BRM**
- **Creating a Payment Suspense Account**
- **Configuring Suspense Reason Codes and Action Owner Codes**

**Enabling Payment Suspense in BRM**

To enable Payment Suspense Manager, run the `pin_bus_params` utility to change the `PaymentSuspense` business parameter. For information about this utility, see *BRM Developer’s Guide*.

To enable Payment Suspense Manager:

1. Go to `BRM_home/sys/data/config`.
2. Create an XML file from the `/config/business_params` object:
   ```bash
   pin_bus_params -r BusParamsAR bus_params_AR.xml
   ```
3. In the file, change `disabled` to `enabled`:
   ```xml
   <PaymentSuspense>enabled</PaymentSuspense>
   ```
4. Save the file as `bus_params_AR.xml`.
5. Load the XML file into the BRM database:
   ```bash
   pin_bus_params bus_params_AR.xml
   ```
6. Stop and restart the CM.
7. (Multischema systems only) Run the `pin_multidb` script with the `-R CONFIG` parameter. For more information, see *BRM System Administrator’s Guide*.
Creating a 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, BRM supports only one payment suspense account. You create payment suspense accounts by using Customer Center and base them on the default customer service representative (CSR) plan. For more information about supporting multiple payment suspense accounts, see the documentation about PCM_OP_PYMNTPOL_SUSPEND_PAYMENT in BRM Opcode Guide.

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 package for your BRM system.
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.

Configuring 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.

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.
- **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:
Configuring BRM for Payment Suspense

- 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.

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.

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.
The following example shows a `reasons.locale` file segment defining a payment suspense reason code domain. Some reason codes are BRM defaults, and some are defined by a user (ID >= 100,000).

```plaintext
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

STR
  ID = 100,101;
  VERSION = 14;
  STRING = "Payment is too large."
  HELPSTR = "The amount of a cash payment is over 10,000."
END

DOMAIN = "Reason Codes - Payment Suspense Action Owner reason";

STR
  ID = 102,001;
  VERSION = 15;
  STRING = "Alaya Baker"
  HELPSTR = "Payments Processing department"
END

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;
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;
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
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.

To define reason codes and action owner codes for payment suspense, you edit the reasons.en_US sample file in the BRM_home/sysmsgs/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" in BRM Developer’s Guide.

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.
Configuring Payment Channels

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.

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.

This document 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.

See also:
- BRM Concepts
- List of Payment Processing Features

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 must 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 must 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.

---

**Important:** By default, verification of accurate payment channel ID mapping is not performed in BRM.

---

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_
Setting Up Payment Channel Information

/home/sys/msgs/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.

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). 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 merchant accounts and automated clearing houses, see "Setting Up Merchant Accounts".

### 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.

**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.
This document explains how to configure Oracle Communications Billing and Revenue Management (BRM) payment collection dates for BRM-initiated customer payments.

See also:
- BRM Concepts
- List of Payment Processing Features

About Customizing Payment Collection Dates for BRM-Initiated Payments

By default, BRM-initiated payments 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 BRM Opcode Guide.

2. At the end of each billing cycle, BRM calculates the payment collection date after determining the bill’s due date.

3. BRM collects the payment.

   BRM-initiated payments are collected by the pin_collect utility.

About Configurable Payment Collection Dates and On-Purchase 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-purchase billing feature to bill the customer immediately. When you use these features, multiple bills associated with a single bill unit 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 21–1)
- Due date = September 10 (August 10 + 31 days)
- Payment collection date = September 5 (September 10 - 5 days)

**Figure 21–1 Regular Billing Cycle Dates**

The Bill 1 payment collection date (September 5) is stored in the bill unit associated with Bill 1.

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

- Bill finalized = “Bill Now” (see Figure 21–2)
- Due date = September 18 (August 18 + 31 days)
- Payment collection date = September 13 (September 18 - 5 days)

**Figure 21–2 Bill Now Billing Cycle Dates**

However, the Bill Now payment collection date (September 13) is not stored in the bill unit. Instead, the earlier payment collection date (September 5) is applied to both bills, as shown in Figure 21–3:
About Configurable Payment Collection Dates and Delayed Billing

The BRM delayed billing feature enables billing for all the bill units 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 more information about delayed billing, see “Setting up Delayed Billing” in BRM Configuring and Running Billing.

Note: If the Bill Now payment collection date were stored in the bill unit 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-purchase billing, see “About Bill Now” and “About On-Purchase 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 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.
Processing Payments in a Multischema System

This document describes how to collect payments in an Oracle Communications Billing and Revenue Management (BRM) multi-schema system.

See also:
- BRM Concepts
- List of Payment Processing Features

Processing Payments in a Multischema System

You can use Payment Center and Payment Tool in a multiscema environment.

The multischema support for payment processing enables you to:
- Create and submit a single payment batch containing payments made to customer accounts residing in multiple schemas.
- Create and submit a single reverse or refund payment batch containing payments made to customer accounts residing in multiple schemas.
- Move the suspended payments to the payment suspense account set up for the connected schema.
- Recycle the suspended payments to one or more customer accounts residing in multiple schemas by using Payment Center. For example, if you are connected to schema 1, you can use Payment Center to recycle payments from the payment suspense account set up for schema 1 to customer accounts residing in different schemas.

To add multischema support for payment processing:

1. Set up a multischema system. See "Installing a Multischema System" in BRM Installation Guide.

2. Create separate payment suspense accounts for each schema in your system. See "Creating a Payment Suspense Account".
This document provides reference information for Oracle Communications Billing and Revenue Management (BRM) payment utilities.

Topics in this document:

- load_pin_ach
- pin_balance_transfer
- pin_cc_migrate
- pin_clean
- pin_collect
- pin_deposit
- pin_recover
- pin_sepa
load_pin_ach

Use this utility to load the merchant account information for all credit card processor and automated clearing house (ACH) vendors into the BRM database. See “Setting Up Merchant Accounts”.

Location

BRM_home/bin

Syntax

load_pin_ach [-d][-v][-t] pin_ach_file

Parameters

-d
Creates a log file for debugging purposes.

-v
Displays information as the utility runs.

-t
Runs a test to check the input to the utility.

pin_ach_file
The name and location of the file that defines merchant accounts. The default pin_ach file is in BRM_home/sys/data/pricing/example.

Results

If the load_pin_ach utility does not notify you that it was successful, look in the utility log file (default.pinlog) to find any errors.
**pin_balance_transfer**

Use this utility to perform automatic sponsored top-ups.

For information about automatic sponsored top-ups, see "About Sponsored Top-Ups".

**Location**

`BRM_home/bin`

**Syntax**

```
pin_balance_transfer [-verbose [file_name.log]] [-test] [-start mm/dd/yy] [-end mm/dd/yy]
```

**Parameters**

- **-verbose**
  Displays information as the utility runs.

- **-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.

- **-end mm/dd/yy or yyyy**
  End date.

**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.
pin_cc_migrate

Use this utility to replace credit or debit card numbers stored in /payinfo/cc objects with tokens. See "Replacing Credit Card Numbers with Tokens".

Important: Ensure that the outstanding payments for credit card accounts are closed before running this utility.

Location

BRM_home/bin

Syntax

pin_cc_migrate -vendor payment_processor_name
[-num number]
[-account account_POID]
[-start_date mm/dd/yy]
[-end_date mm/dd/yy]
[-verbose [file_name.log]]
[-report]
[-help]

Parameters

-vendor payment_processor_name
Specifies the credit card processor or ACH to use for validating credit and debit cards.
See Setting Up Merchant Accounts.

-num number
Specifies the number of /payinfo/cc objects to select for tokenization.

-account account_POID
Specifies the account POID. Use this parameter to replace credit or debit card numbers with tokens for a single account.

-start_date mm/dd/yy
Specifies the start date. The start and end dates specify the time range for selecting objects for tokenization. If you do not specify a value for the -start_date parameter, the start date is set to the current date. If a start date is specified, the entire day is included.

-end_date mm/dd/yy
Specifies the end date. If you do not specify a value for the -end_date parameter, the end date is set to the current date. If an end date is specified, the entire day is included, ending exclusively on the first second of the next day (00:00:00 a.m.).

-verbose file_name.log
Displays information about successful or failed processing as the utility runs.

-report
Displays more information than the verbose parameter. Requires the verbose option. Returns a list of the /payinfo/cc objects for which tokenization has been completed.

Important: Ensure that the outstanding payments for credit card accounts are closed before running this utility.
-help
Displays syntax and parameters for this utility.

Results
If the pin_cc_migrate utility does not notify you that it was successful, look in the utility log file (default.pinlog) to find any errors.
pin_clean

Use this utility to find unresolved credit card and direct debit payments. See "Resolving Failed BRM-Initiated Payment Transactions".

---

**Important:** For multischema systems, you must run the utility separately against each database schema in your system.

---

**Location**

`BRM_home/bin`

**Syntax**

```
```

**Parameters**

- **-summary**
  Displays the total number of each type of unresolved credit card transactions. 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.

- **-verbose**
  Displays information about successful or failed processing as the utility runs.

- **-help**
  Displays syntax and parameters for this utility.

**Results**

If the `pin_clean` utility does not 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_collect**

Use this utility to collect the balance due for accounts that use credit card and direct debit payment methods. See "Running the pin_collect Utility to Collect BRM-Initiated Payments".

**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 [file_name.log]]
[-help]
```

**Parameters**

- **-pay_type payment_method**
  Specifies the payment method:
  - 10003 for credit card
  - 10005 for direct debit
  - 10018 for SEPA

- **-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 not applicable for SEPA payment type.

- **-active | -close | -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 [file_name.log]**
  Displays information about successful or failed processing as the utility runs.

- **-report**
  Displays more information than the verbose parameter. Requires the verbose option.

- **-rebill**
  Collects any outstanding bills for a given account status.
-start [mm/dd/yy or yyyy | number_of_days]
Start date. Collects payments on the day utility is run and the day before the utility is run.

If a start date is specified, the entire day is included. The end date is automatically the current date if you do not specify a value for the -end parameter.

-end [mm/dd/yy or yyyy | number_of_days]
End date.

If an end date is specified, that entire day is included, ending at, but not including, the 0th (first) second of the next day (00:00:00 a.m.). The end date cannot be a future date.

-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.

When it is called by the pin_bill_day script, the pin_collect utility logs error information in the pin_mta.pinlog file.
pin_deposit

Use this utility to deposit 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 "Using More Than One Payment Processor" for more information.

Important: For multischema systems, you must run the utility separately against each database schema in your system. See "Running Non-MTA Utilities in Multischema 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[filename.log]]
[-help]

Parameters

-pay_type payment_method
Specifies the payment method. There are two possible values:

- 10003 for credit card
- 10005 for direct debit

-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.

See "Setting Up Merchant Accounts" 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.

If a start date is specified, the entire day is included. The end date is automatically the current date if you do not specify a value for the -end parameter.

If an end date is specified, that entire day is included, ending at, but not including, the 0th (first) second of the next day (00:00:00 a.m.). The end date cannot be a future date.

-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.

-**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.

When it is called by the **pin_bill_day** script, the **pin_deposit** utility logs error information in the **pin_billd.pinlog** 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.

Important: For multischema systems, you must run the utility separately against each database schema in your system. See "Running Non-MTA Utilities in Multischema Systems" in BRM System Administrator’s Guide.

Location

BRM_home/bin

Syntax

```
pin_recover -pay_type payment_method
-vendor payment_processor_name
[-rfr | -resubmit batch_ID | -recover_payment]
-verbose[<file_name.log>] [-test] [-help]
```

Parameters

- **-pay_type payment_method**
  Specifies the payment method. There are two possible values:
  - 10003 for credit card
  - 10005 for direct debit

- **-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.
  See "Setting Up Merchant Accounts" 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 Failed Deposits and Conditional Deposits" 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, ensure 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.

-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.
pin_sepa

Use this utility to generate and process SEPA request and response files.
See "Implementing SEPA Payment Processing".

Location

BRM_home/apps/pin_sepa

Syntax

To generate SEPA request XML files:

```
pin_sepa [-sdd_req | -sct_req | -sepa_rev] [-verbose [file_name.log]] [-help]
```

To process SEPA response XML files:

```
pin_sepa -sepa_resp [-verbose] [-help]
```

To generate SEPA request XML files and process SEPA response XML files:

```
pin_sepa -all [-verbose] [-help]
```

Parameters

- `-sdd_req`
  Generates SEPA Direct Debit request XML files.

- `-sct_req`
  Generates SEPA Credit Transfer request XML files.

- `-sepa_rev`
  Generates SEPA Direct Debit reversal request XML files.

- `-sepa_resp`
  Processes SEPA response XML files for SEPA Direct Debit and SEPA Credit Transfer.

- `-all`
  Generates SEPA request XML files for SEPA Direct Debit, SEPA Credit Transfer, and SEPA Direct Debit reversal and processes SEPA response XML files for SEPA Direct Debit and SEPA Credit Transfer.

- `-verbose [>filename]`
  Displays information about successful or failed processing as the utility runs. `filename` specifies the file to redirect the output to.

  **Note:** This parameter is always used in conjunction with other parameters and commands.

- `-help`
  Displays the syntax and parameters for this utility.
Results

The **pin_sepa** utility uses the following file naming convention for the request XML files:

\textit{typedbno-YYYYMMDD-X.xml}

where:

- \textit{type} is SDD for SEPA Direct Debit, SCT for SEPA Credit Transfer, or SDD-REV for SEPA Direct Debit reversal.
- \textit{dbno} is the database number.
- \textit{YYYYMMDD} is the year, month, and day on which the file was generated.
- \textit{X} is a unique, eight-digit sequence number.

If the **pin_sepa** utility does not notify you that it was successful, look in the log file (javapcm.log is the default log filename) 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 \texttt{Infranet.properties} configuration file.
This appendix describes Oracle Communications Billing and Revenue Management (BRM) payment attributes

See also:
- BRM Concepts
- List of Payment Processing Features

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 >= PIN_PYMT_SUCCESS and < PIN_PYMT_SUSPENSE. The numeric range for successful payments is 0-14.
- **Suspended payments** have a value >= PIN_PYMT_SUSPENSE and < 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 >= PIN_PYMT_FAILED and < PIN_PYMT_STATUS_MAX. The numeric range for financially failed payments is 30-44.
- Payments with a status >= 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 Payment Status Codes**

The following values described in Table A–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 BRM Managing Accounts Receivable.</td>
</tr>
</tbody>
</table>
| PIN_PYMT_SUSPENSE    | 15           | The payment:  
  - Arrived in BRM as invalid but meets the criteria for suspending a payment. The payment is saved to the payment suspense account.  
  - Arrived in BRM as valid but was manually suspended before or after it posted to the customer account.  

This status code is not used for recycled payments. The Payment Suspense Manager feature must be enabled. See “Managing Suspended Payments”. |
| PIN_PYMT_FAILED_SUSPENSE | 16       | 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.  

This status code is used only for payments that originally post to the suspense account. It is not used for recycled payments. The Payment Suspense Manager feature must be enabled. See “Managing Suspended Payments”. |
Table A–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_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. 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>