Oracle® Communications Billing and Revenue Management

Configuring and Collecting Payments

Release 15.2
G35885-01
January 2026

Oracle Communications Billing and Revenue Management Configuring and Collecting Payments, Release 15.2

G35885-01

Contents

About This Content

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

Audience

This guide is intended for operations personnel.

1
Running Payment Collection Utilities

Learn how to use the **pin_collect** and **pin_deposit** utilities to collect and deposit your customers' payments in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

See also "Overview of Payments" in *BRM Concepts*.

About the Payment Collection Utilities

You use the following utilities to collect payments from your customers:

- **pin_collect** collects the balance due for bills that are paid by credit card or direct debit.
- **pin_deposit** deposits pre-authorized credit card payments and PINless debit payments into your account.

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

You typically run the **pin_collect** and **pin_depost** utilities in the same scripts that run other billing utilities. See "Running Billing Scripts" in *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

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 01/01/23, payments are collected from 00:00:00 a.m. on 12/31/22 to 00:00:00 a.m. on 01/02/23.

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

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

Use the **-pay_type** parameter to collect payments for only one payment method, such as credit card or SEPA.

You can configure payment collection:

- To increase billing performance, you run multiple threads of the **pin_collect** utility. See "Tuning the Number of Children for Billing Utilities" 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 "Billing Accounts By Using the pin_make_corrective_bill Utility" in *BRM Configuring and Running Billing*.

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

When to Run the pin_collect Utility

Run the **pin_collect** utility at the following times:

- In the **pin_bill_day** script for all accounts. 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.

Note

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 (this is 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 to **0**. For example:

```
pin_collect -start 0
```

You can also specify exact start and end dates, and you can specify the 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.

Note

For open item accounting, the end date of the bill is not used to determine whether the bill falls within the specified range and qualifies for collection: only the start date is used.

Running the pin_deposit Utility to Deposit BRM-Initiated Payments

The **pin_deposit** utility deposits pre-authorized credit card payments and PINless debit 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 or Customer Center.

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

For direct debit, the **pin_deposit** utility searches for all unpaid PINless debit transactions, and then sends the authorization codes, transaction dates, trace number, and method of payment (MOP) to the debit network for depositing.

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

For information about the **pin_deposit** utility's syntax, see "pin_deposit".

When to Run pin_deposit

Run the **pin_deposit** utility daily as part of the **pin_bill_day** script.

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.

Note

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** parameters 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**.
- **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:

   ```
   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:

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

   Note

   There might be several sets of entries for each payment processor. Be sure to add new entries for each set of existing entries.

2
Processing Credit Card and Debit Card Payments with Paymentech

Learn how Oracle Communications Billing and Revenue Management (BRM) uses Paymentech to process your customers' credit card and direct debit payments.

Topics in this document:

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**.

About Credit Card Validation and Authorization

Credit card validation validates a customer's address by checking the ZIP code and street address. Credit card validation occurs during account creation, and when customers change their credit card number. If you use the Address Verification System (AVS), Paymentech gives you a discount for each credit card transaction charge.

Note

AVS supports addresses in the United States and Canada only. For information about changing the AVS validation results, see "Changing How BRM Handles Paymentech Address Validation Return Codes".

Credit card authorization validates the customer's credit card by checking the card number, expiration date, credit limit, and so forth. Authorization occurs at the following times:

- At account creation, or when customers change 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, such as a purchase fee.

Credit card validation and authorization occur 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. You typically install this at the same time as the rest of the BRM software. If you did not, see "Installing Individual BRM Components" in *BRM Installation Guide*.

2. Contact Paymentech to establish a link with Paymentech. See "Exchanging Connection Information with Paymentech".
3. Configure **dm_fusa** to send batch payment transactions to Paymentech through SFTP or TCP/IP.
   - For SFTP, see: Using SFTP for Batch Payment Transactions
   - For TCP/IP, see: Using TCP/IP for Batch Payment Transactions
4. Configure **dm_fusa** to send online transactions to Paymentech through TCP/IP. See "Configuring Online Payment Transactions".
5. Configure merchant accounts. See "Setting Up Merchant Accounts".
6. Set up BRM to process PINless debit payments. See "Configuring PINless Debit Payment Processing".
7. Set up the Paymentech HeartBeat application. See "Monitoring the Paymentech Connection".
8. Specify Paymentech configuration options; for example, enabling direct-debit processing and enabling fraud protection. See "Paymentech Configuration Options".
9. Configure performance options. See "Configuring Paymentech Processing Performance".
10. 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 2-1 lists the information that Paymentech needs from you.

Table 2-1   BRM Default Values for Paymentech

Paymentech Information	BRM Default
The IP address and port number for the machine running the Paymentech Data Manager (**dm_fusa**).	None. This is required only to use the Paymentech HeartBeat application, which is integrated with the Paymentech Data Manager. For more information, see "Monitoring the Paymentech Connection".
Is this for an existing Presenter ID (PID)?	No
What is the application software that formats the file?	Written by in-house programmers
What is the communications software that sends the file?	Customized by the software vendor
What is the online data communications protocol used to send the online authorization transaction?	TCP/IP Berkley Socket Interface
What is the batch data communications protocol used to send the batch file?	TCP/IP Berkeley Socket Interface
What online format to use for sending online authorizations?	For information about compatible Paymentech online processing format specifications, see "Additional BRM Software Requirements" in *BRM Compatibility Matrix*.
Will you load balance online authorizations between Paymentech's data centers, or will you use one data center as primary and one as backup?	Primary and Backup
What batch format to use for sending batch files?	For information about compatible Paymentech batch processing format specifications, see "Additional BRM Software Requirements" in *BRM Compatibility Matrix*.
Will you receive the batch reply file by sending an RFR (Request For Response) record or not?	1 Call (IA) - No RFR record sent to pick up the reply file.
Will you send authorizations separately from deposits OR will you send conditional deposits that will result in a deposit upon authorization approval?	Separate authorizations and deposits *and* conditional deposits.
What is the average size of your files be in production? (How many records/transactions?)	None. This number should be based on your company's projected customer account creation growth and billing rate.
What is the projected submission schedule?	Daily.
Number of times per day?	Once.

Table 2-1 (Cont.) BRM Default Values for Paymentech

Paymentech Information	BRM Default
What Paymentech functionality do you intend to test?	This list reflects a typical prepaid services company. • Online Credit Card Authorization • Online Electronic Check Processing (ECP) Verification • PINless Direct Debit Purchase Authorizations • Batch Electronic Check Processing (ECP) Validate & Deposit • Batch Deposits • Batch Conditional Deposits (for authorization & settlement) • Batch Refunds • Full AVS (Address Verification Service) • Zip only AVS • No AVS • Visa CVV2 • Amex CID • MasterCard CVC2 • Discover CID • ECI Indicator (also called Transaction Type) • International Currencies (specify) • Merchant Descriptor (requires Risk approval) • Switch/Solo Cards

Using SFTP for Batch Payment Transactions

You can configure the Paymentech Data Manager (**dm_fusa**) to use SFTP for sending batch payment transactions to the Paymentech payment processor. Batch payment transactions include batch payments, multiple verifications, multiple authorizations, deposits, and refunds.

To send batch payment transactions through SFTP, follow these steps:

- Setting Up Authentication Between dm_fusa and Paymentech
- Configuring Your SSH Client Configuration File
- Sending Batch Payment Transactions Through SFTP

Setting Up Authentication Between dm_fusa and Paymentech

Set up authentication between **dm_fusa** and Paymentech by generating a public key on the **dm_fusa** machine and copying it to the Paymentech server. To test payment processing with the Paymentech Simulator, also copy the public key to your Paymentech Simulator server.

To setup up authentication between **dm_fusa** and Paymentech:

1. Generate an RSA public key.

   For example, this command generates an RSA public key with a key length of 4096 bits:

   ```
   ssh-keygen -t rsa -b 4096
   ```

2. Copy the RSA public key to the Paymentech SFTP server:

   ```
   ssh-copy-id -i pathPublicKey  userName@hostNamePaymentech
   ```

where:

- *pathPublicKey* is the path and file name of the public key generated in step 1.
- *userName* is name of the **dm_fusa** user.
- *hostNamePaymentech* is the host name of the Paymentech SFTP server.

3. Copy the RSA public key to the Paymetech Simulator server:

   ```
   ssh-copy-id -i pathPublicKey  userName@hostNameSimulator
   ```

   where *hostNameSimulator* is the host name of the Paymentech Simulator server.

4. Verify the connection:

   ```
   sftp -i pathPublicKey  userName@hostName
   ```

   where *hostName* with the appropriate server host name (Paymentech or the Simulator).

Configuring Your SSH Client Configuration File

Connect **dm_fusa** to the Paymentech SFTP server by setting parameters such as server, port, algorithms, and ciphers in the SSH client configuration file, located at **~/.ssh/config**.

On your **dm_fusa** machine, add entries to the SSH file to meet your business requirements. For example:

```
Host CServer
   HostName SftpHost
   User UserName
   Port PortNumber
   IdentityFile pathPrivateKey
   #KexAlgorithms kexAlgorithms
   #Ciphers cipherNames
   #HostKeyAlgorithms hostKeyAlgorithms
   #MACs macAlgorithm
   #LogLevel DEBUG3
   #LogLevel VERBOSE
```

where:

- *sftpHost:*Host name of the Paymentech's SFTP server.
- *userName*: User name for accessing the Paymentech SFTP server.
- *portNumber*: SFTP port number for the Paymentech server.
- *pathPublicKey*: Path and file name of the private key that matches the public key you generated in "Setting Up Authentication Between dm_fusa and Paymentech".
- *kexAlgorithms*: List of key exchange algorithms, in order of preference, separated by commas.
- *cipherNames*: List of encryption algorithms for data transfer, separated by commas.
- *hostKeyAlgorithms*: List of public key algorithms the SSH server uses to authenticate itself to the client, separated by commas.
- *macAlgorithm*: List of MAC algorithms for data verification, separated by commas.

Sending Batch Payment Transactions Through SFTP

You can configure the Paymentech DM (**dm_fusa**) to send batch payment transactions to the Paymentech payment processor using SFTP. Batch payment transactions include batch payments, multiple verifications, multiple authorizations, deposits, and refunds. By default, **dm_fusa** uses TCP/IP for batch payment transactions.

To send batch payment transactions through SFTP:

1. Open the Paymentech DM configuration file (*BRM_home*/**sys/dm_fusa/pin.conf**).
2. Set the connection type between **dm_fusa** and Paymentech for batch payment transactions:

   ```
   - dm_fusa batch_proto sftp
   ```

3. If using SFTP for batch transactions, uncomment and configure the following entries:

   ```
   - dm_fusa sftp_host hostName
   - dm_fusa sftp_pkey_pwd passPhrase
   - dm_fusa sftp_indir inputPath
   - dm_fusa sftp_outdir outputPath
   - dm_fusa sftp_rfrfile rfrFileName
   - dm_fusa sftp_proc_pat pattern
   - dm_fusa sftp_retrys retryValue
   - dm_fusa sftp_retry_interval intervalValue
   ```

   Table 2-2 describes the values to use with the **pin.conf** entries.

   Table 2-2 Variables for the pin.conf Entries

Parameter	Description
hostName	The host section name in the SSH client configuration file.
passPhrase	The SFTP private key passphrase. If omitted, the passphrase is retrieved from the wallet.
inputPath	The absolute path to the input directory where batch files are uploaded using SFTP.
outputPath	The absolute path to the output directory where response files are downloaded using SFTP.
rfrFileName	The name of the RFR file to retrieve from the output directory. If this parameter is not specified, **dm_fusa** downloads any unprocessed file found in remote output directory.
pattern	The extension appended to response files.
retryValue	The number of times to check for a response file in the remote output directory.
intervalValue	The time interval, in seconds, to check for a response or RFR file in the remote output directory.

4. Save your changes.
5. Stop and restart the Paymentech DM for the settings to take effect.

Using TCP/IP for Batch Payment Transactions

To send batch payment transactions through TCP/IP, follow these steps:

To connect **dm_fusa** to the Paymentech payment processor:

1. Open the Paymentech DM configuration file (*BRM_home*/**sys/dm_fusa/pin.conf**).
2. Enable or disable TLS connections from **dm_fusa** to Paymentech:

```
- dm_fusa fusa_tls_enabled 0|1
```

where **0** disables TLS, and **1** enables TLS. The default is **1**.

3. Specify the connection type between **dm_fusa** and Paymentech for batch payment transactions:

   ```
   - dm_fusa batch_proto socket
   ```

4. Specify the Paymentech host name and port for batch payment transactions:

   ```
   - dm_fusa batch_srvr batchSrvr
   - dm_fusa batch_port batchPort
   ```

   where *batchSrvr* and *batchPort* are the Paymentech TCP/IP server and port.

5. Save your changes.
6. Stop and restart the Paymentech DM for the settings to take effect.

Configuring Online Payment Transactions

You configure the Paymentech Data Manager (**dm_fusa**) to send online payment transactions to the Paymentech payment processor using TCP/IP. Online payment transactions include single authorization transactions and verification transactions.

To configure **dm_fusa** to send online payment transactions through TCP/IP, follow these steps:

1. Open the Paymentech DM configuration file (*BRM_home***/sys/dm_fusa/pin.conf**).
2. Enable or disable TLS connections from **dm_fusa** to Paymentech:

   ```
   - dm_fusa fusa_tls_enabled 0|1
   ```

   where **0** disables TLS, and **1** enables TLS. The default is **1**.

3. Specify the server status for online payment transactions:

   ```
   - dm_fusa online_proto socket|linkdown
   ```

   where **socket** specifies that the online server is running, and **linkdown** specifies that the online server is offline. The default is **socket**.

4. Specify the Paymentech host name and port for online payment transactions:

   ```
   - dm_fusa online_srvr onlineSrvr
   - dm_fusa online_port onlinePort
   ```

   where:

   - *onlineSrvr* is the IP address or host name of the Paymentech server for online payment transactions.
   - *onlinePort* is the port where Paymentech receives online payment transactions.

5. Save your changes.
6. Stop and restart the Paymentech DM for the settings to take effect.

Configuring Paymentech Processing Performance

Configure the following options:

- Handling Concurrent Online Paymentech Requests

- Setting the Connection Timeout Length and Retries

To improve performance during account creation, see "Increasing Account Creation Speed When Paymentech Is Offline" in *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:
   - 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/pin.conf**).
2. Increase the number of times that **dm_fusa** retries a connection to Paymentech:

   ```
   - dm_fusa connect_retrys value
   ```

   The default is **2**, but you can enter any number.
3. Change the amount of time, in seconds, that **dm_fusa** waits for a response from Paymentech for online authorizations:

   ```
   - dm_fusa fusa_timeout value
   ```

   The default is **600**.

4. Change the amount of time, in seconds, that **dm_fusa** waits for a response from Paymentech for batch scripts:

   ```
   - dm_fusa fusa_batch_timeout value
   ```

   The default is **600**.

5. Save the file.
6. 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.

> **Note**
>
> If BRM stops receiving HeartBeat messages and is in the middle of a transaction, the connection does 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 runs automatically each time you process BRM-initiated payments.

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

```
Received (20) chars: Heartbeat request [HO19999999813123258^M]
Sending Heartbeat response [HI19999999813123300^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() :
HI19999999813123300^M
```

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:

```
E Tue Aug 08 10:51:24 2006  elm  dm_fusa:2994  qbe_fusa.c(1.13):645
1:elm:dm_fusa:2991:1:0:1155059471:0
Socket read error in dm_fusa_respond_heartbeat() recv() returned (0)
E Tue Aug 08 10:51:24 2006  elm  dm_fusa:2994  qm_back.c(7):299
1:elm:dm_fusa:2991:1:0:1155059471:0
Error(7) processing heartbeat monitor fd(5)
```

3

Paymentech Configuration Options

Learn how to configure the way that Paymentech processes credit card and direct debit data in Oracle Communications Billing and Revenue Management (BRM).

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

You can use soft descriptors to add information to your customers' credit card or checking account statements, including:

- Your "doing business as" (DBA) name
- The product or 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.

An asterisk separates the DBA and product names on the customer's statement. The entry is truncated on the statement if it is longer than 22 characters (including spaces). In this 22-character-maximum line, the asterisk delimiter can appear in positions 4, 8, or 13.

To add soft descriptor information to your customers' statements:

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. Specify the soft descriptor information in the following entries:

```
- dm_fusa     sd_merchantName_dba          DBA
- dm_fusa     sd_merchantName_pdt          productName
- dm_fusa     sd_merchantName_phone        phoneNumber
```

where:

- *merchantName* is the merchant name. It must match the value set in the CM pin.conf file's merchant entry.
- *DBA* is your "doing business as" name that the customer knows you by.
- *productName* is the name of your product.
- *phoneNumber* is the phone number for your customer support line.

For example, if the merchant name is **psi**, the DBA name is **BRM**, the product name is **InternetSVC**, and the customer service number is **800-555-1234**, you would use the following entries:

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

4. Change the other related entries according to the instructions in the file.
5. Save and close the file.
6. 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 "Customizing the Minimum Amount to Charge" in *BRM Opcode Guide*.

Changing How BRM Handles Paymentech Authorization Return Codes

The Paymentech authorization codes BRM uses 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 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 it.
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, customize the PCM_OP_PYMT_POL_VALIDATE policy opcode. See "Processing Paymentech Address Validation Return Codes" in *BRM Opcode Guide*.

Specifying the Batch Mode Encryption Key

If you process multiple credit card transactions simultaneously, 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 card number and security code is encrypted.

You specify the encryption method and key in the Paymentech configuration file. The default encryption method is OZT. For more information, see "Encrypting Data" in *BRM Developer's Guide*.

Note

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**) in a text editor.
2. Edit the **-crypt** entry:

   ```
   - crypt Encrypt_method| BRM_home/lib/Encryption_library "Secret_key"
   ```

   where:

   - *Encrypt_method* is the type of encryption method, which is either **aes** or **ozt**.
   - *Encryption_library* is the path and file name of the encryption library. The prefix for the library is **lib** for Linux or **null ""** for Windows. The extension for the library is **.so** for Linux and **.dll** for Windows.
   - *Secret_key* is your encrypted AES or OZT key.

   For example:

   ```
   - crypt ozt|BRM_home/lib/libpin_crypt_ozt4dm64.so "&ozt|encryptedKey"
   ```

   Note

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

3. Save and close the file.

4. Stop and restart the Paymentech DM.

Obtaining Card Type Indicator Information from Paymentech

You can configure BRM to request Card Type Indicator (CTI) details from Paymentech, which specify the type of card that consumers use for payments.

To set up **dm_fusa** to request CTI details during account creation with online and batch transactions, customize the PCM_OP_PYMT_POL_SPEC_VALIDATE and PCM_OP_PYMT_POL_PRE_COLLECT policy opcodes according to the comments provided in the source files. When Paymentech returns CTI details, BRM stores them in the PIN_FLD_TYPE_STR field of **/event/billing/charge/cc** and **/event/billing/validate/cc** objects in the BRM database.

Note

- Paymentech supports two versions of CTI records: **01** and **02**. BRM supports version **02** because it is the superset of version 01.
- Batch transactions are used only for deposits and conditional deposits.

See "BRM-Initiated Payment Processing" in *BRM Opcode Guide* for more information.

If BRM sends the Format Indicator and the MOP does not equal **AX**, **CR**, **CZ**, **DD**, **DI**, **IM**, **JC**, **MC**, **MR**, **VI**, or **VR**, Paymentech rejects the transaction with Response Reason Code 241 (Illegal Action).

If it sends the Format Indicator and the Action Code does not equal **AU**, **BI**, **PP**, or **VF**, Paymentech rejects the transaction with Response Reason Code 225 (Invalid Field Data).

If this Format Indicator is sent, the MOP equals **AX**, and the Action Code does not equal **PP**, Paymentech rejects the with Response Reason Code 225 (Invalid Field Data).

Since action code **PP** is not used in BRM, customizations enabling the Card Type Indicator must ensure that CTI is not sent for American Express Cards.

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, use the **pin_bus_params** utility to enable the **CidRequired** and **Cvv2Required** business parameters. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

Note

The CVV2 and CID numbers are stored in BRM with a **NULL** value for security reasons. If you have the **Cvv2Required** business parameter entry enabled, the information is sent directly to Paymentech for validation without being stored in the database. (Even if your Connection Manager (CM) does not require this additional fraud prevention, Paymentech still accepts the information if it is sent).

To require CVV2 and CID numbers:

1. Go to *BRM_home***/sys/data/config**.
2. Use the following command to create an editable XML file from the accounts receivable instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsAR bus_params_AR.xml
   ```

   This command creates an XML file named **bus_params_AR.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.
3. In **bus_params_AR.xml.out**, set **CidRequired** and **Cvv2Required** to **enabled**:

   ```
   <CidRequired>enabled</CidRequired>

   <Cvv2Required>enabled</Cvv2Required>
   ```

   The default value for both of the above business parameters is **disabled**.

   Caution

   BRM uses the XML in this file to overwrite the existing instance of the **/config/business_params** object. Use care when updating parameters in the file.
4. Save and exit the file.
5. Rename the **bus_params_AR.xml.out** file to **bus_params_AR.xml**.
6. Use the following command to load your changes into the **/config/business_params** object:

   ```
   pin_bus_params bus_params_AR.xml
   ```

   You should run this command from the *BRM_home***/sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide.*
7. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

The new value is effective immediately. You do not need to restart the CM.

Specifying the Maximum Number of Digits Allowed for CVV2 Verification

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

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

- For Customer Center: Use the Configurator application provided with Customer Center SDK to modify the maximum number of CCV2 digits allowed by Customer Center. You enter the information in the **CVV2 Number - maximum digits allowed** field of the Payment Configurator.
- For BRM server: Customize the PCM_OP_CUST_POL_VALID_PAYINFO policy opcode to validate the number of digits passed in the PIN_FLD_SECURITY_ID input flist field of the PIN_FLD_CC_INFO array.

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.)

To enable direct debit processing:

1. Open the Connection Manager (CM) configuration file (*BRM_home*/**sys/cm/pin.conf**).
2. Change the value of dd_validate to **1** to enable direct debit validation:

   For example:

   ```
   - fm_pymt_pol dd_validate 1
   ```
3. Save and exit the file.
4. Go to *BRM_home*/**sys/data/config**.
5. Use the following command to create an editable XML file from the accounts receivable instance of the **/config/business_params** object:

   ```
   pin_bus_params -r BusParamsAR bus_params_AR.xml
   ```

   This command creates an XML file named **bus_params_AR.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.
6. In **bus_params_AR.xml.out**, set **DDcollect** to **enabled**:

   ```
   <DDcollect>enabled</DDcollect>
   ```

   Caution

   BRM uses the XML in this file to overwrite the existing instance of the **/config/business_params** object. Use care when updating parameters in the file.
7. In **bus_params_AR.xml.out**, set **DDRevalidationInterval** to the number of seconds applicable:

   ```
   <DDRevalidationInterval>value</DDRevalidationInterval >
   ```

   The default value is **3600** seconds (one hour).

8. Save and exit the file.
9. Rename the **bus_params_AR.xml.out** file to **bus_params_AR.xml**.
10. Use the following command to load your changes into the **/config/business_params** object:

    ```
    pin_bus_params bus_params_AR.xml
    ```

    You should run this command from the *BRM_home***/sys/data/config** directory, which includes support files used by the utility. To run it from a different directory, see "pin_bus_params" in *BRM Developer's Guide*.
11. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

    For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide.* For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

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

4 Supported Paymentech Functionality

Learn how Oracle Communications Billing and Revenue Management (BRM) supports Paymentech functionality such as account verification and batch processing.

Topics in this document:

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 of $**0.00**. Paymentech validates the transaction against various validation files. If account verification is successful, Paymentech responds with 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 of $**0.00**. If account verification passes, Paymentech responds with 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 of $**0.00**. If account verification is successful, Paymentech responds with Response Reason Code **104** (No Reason to Decline).

For more information, see the Paymentech documentation.

About Stored-Credential Transactions

Credit card networks, such as VISA, MasterCard, Diners, Discover, JCB, and American Express, support the stored credential framework, which allows merchants to use stored credentials for transactions. A stored credential is a cardholder's payment information, such as an account number or a payment token, that is stored by a merchant or its agent, a payment facilitator, or a staged digital wallet operator for processing future transactions. Paymentech card processors also support customer-initiated or merchant-initiated transactions with stored credentials.

BRM supports payment transactions with stored credentials for VISA, MasterCard, Diners, Discover, JCB, and American Express cards. You can also customize BRM to support stored-credential transactions for other card networks. When VISA, MasterCard, Diners, Discover, JCB, and American Express cards are used for payments, the PCM_OP_PYMT_COLLECT opcode sends the following information required for card transactions to the Paymentech Data Manager (**dm_fusa**) and stores the responses received from the Paymentech DM for future transactions:

- Type of charge, such as recurring, one-time, or installment
- Transaction type
- Information on whether the card details are stored for future use
- A unique ID (TXID) obtained from a previous verify/charge transaction of the same type

You can override the information sent to the Paymentech DM based on your business requirements. If you do not want to store credentials for future transactions, you can remove this information from the input passed to the PCM_OP_PYMT_COLLECT opcode. See "Storing Card Credentials for Future Transactions" in *BRM Opcode Guide*.

For information about purging card credentials, see "Purging Card Credentials" in *BRM Opcode Guide*.

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 supports 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. For more information about obtaining the certification, see the Paymentech website.
- 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 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**.

5 Configuring PINless Debit Payment Processing

Learn how Oracle Communications Billing and Revenue Management (BRM) uses Paymentech to process PINless debit payment transactions.

Topics in this document:

About PINless Debit Payments

You can configure BRM to support PINless debit payments from your customers. PINless debit payments are debit card payments made without the card's magnetic stripe or your customer's PIN. Supporting PINless debit payments can reduce your transaction processing costs, because transactions are handled by domestic debit networks that tend to have lower fees than Visa, MasterCard, and other brands.

In BRM, these payments are processed through Paymentech as PINless debit BillPay.

About PINless Debit Validation and Authorization

BRM validates all debit card payments using Paymentech's Account Verification functionality. For more information, see "About Paymentech Account Verification ".

BRM authorizes both online and batch PINless debit payments by sending a Purchase Authorization (PA) request to Paymentech that includes the following:

- Action code set to **PA**
- Method of Payment (MOP) set to **DP**
- (For online only) Format indicator set to **02**, with the billing reference number set to the customer's account number
- (For batch only) Product record set to **PDE001**

Paymentech checks if the customer's account has enough funds and, if it does, debits the customer's account.

The response from Paymentech includes the MOP set to the PINless debit network and a trace number, which BRM then stores in the **/event/billing/payment/cc** object. When **pin_deposit** is run, it includes both the MOP the PINless Debit transaction was processed as and the trace number in the deposit request it sends to Paymentech.

Limitations of PINless Debit Payments

Before configuring BRM to support PINless debit payments, consider the following limitations:

- Tokenization is not supported for PINless debit payment transactions. As a result, if a card number has already been tokenized in BRM, it cannot be changed to a PINless debit payment method.
- The Card Type Indicator (CTI) cannot be sent with PINless debit transactions.
- Stored credentials are not supported with PINless debit transactions.
- Refunds are not supported with PINless debit transactions.

Setting Up PINless Debit Processing

> **Note**
>
> Before Paymentech can start processing PINless debit transactions, your debit network must approve your merchant account and add all of your payment channels.

To set up BRM to support PINless debit payment transactions, do the following:

1. Enable the **PINlessDebitProcessing** business parameter in BRM. See "Enabling PINless Debit Payments in BRM".
2. Contact your Merchant Services Representative at Paymentech to have them enable PINless BIN File Management for your account.
3. Customize the PCM_OP_CUST_POL_PREP_PAYINFO policy opcode to identify PINless debit payment types. See "Identifying PINless Debit Payment Types".

Enabling PINless Debit Payments in BRM

To enable BRM to collect payments through PINless direct debit, use the **pin_bus_params** utility to change the **PINlessDebitProcessing** business parameter. For information about the utility's syntax and parameters, see "pin_bus_params" in *BRM Developer's Guide*.

To enable the collection of PINless debit payments:

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, set the **<PINlessDebitProcessing>** element to **enabled**:

   ```
   <PINlessDebitProcessing>enabled</PINlessDebitProcessing>
   ```

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.

Identifying PINless Debit Payment Types

BRM determines whether a payment card supports PINless debit transactions by reading the first character in the **/payinfo/cc** object's PIN_FLD_TYPE_STR field. By default, this field is not set by BRM. You must customize the PCM_OP_CUST_POL_PREP_PAYINFO policy opcode to identify whether a payment card supports PINless debit and then set the PIN_FLD_TYPE_STR in **/payinfo/cc** appropriately.

To identify and mark PINless debit payment types, customize the PCM_OP_CUST_POL_PREP_PAYINFO policy opcode to do the following:

1. Validate that the PIN_FLD_TYPE_STR field in the **/payinfo/cc** object is not already tokenized.

   If the field contains a token, it cannot be changed to a PINless debit payment type. The customer's debit card must be re-registered to be used for PINless debit transactions.

2. Customize the **fm_cust_pol_is_pinless_debit()** function to use the Paymentech BIN files to determine whether a payment card supports PINless debit.
3. Customize the **fm_cust_pol_is_pinless_debit()** function to return **true** when the card is PINless debit and **false** when it is not.

   Depending on the value returned by **fm_cust_pol_is_pinless_debit**(), the policy opcode sets the **/payinfo/cc** object's PIN_FLD_TYPE_STR field as follows:

   - If **True** is returned, the first character of the PIN_FLD_TYPE_STR output flist field is set to **Y**.
   - If **False** is returned, the first character of the PIN_FLD_TYPE_STR output flist field is set to **N**.

For more information about the PCM_OP_CUST_POL_PREP_PAYINFO policy opcode, see "Customizing Payment Method Data Preparation" in *BRM Opcode Guide*.

6

Setting Up Merchant Accounts

Learn how to set up merchant accounts for BRM-initiated payments in Oracle Communications Billing and Revenue Management (BRM). Payment processors, such as Paymentech, use merchant accounts to identify the companies who send them payments.

Topics in this document:

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

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 (DM) 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 -1     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 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 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. 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:

   ```
   - dm_fusa   mid_merchantName_currencyID    merchant_account_number
   ```

   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**).

7

Masking Credit Card Numbers by Using Tokens

Learn how to mask credit card numbers by using tokens in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

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.

Note

BRM does not tokenize PINless debit card numbers.

By default, credit card tokenization is enabled. When 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 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 card number) 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 processed credit cards before enabling tokenization, your BRM database includes untokenized card numbers. Do the following to tokenize existing card numbers stored in the BRM database:

1. Replace existing untokenized card numbers in the BRM database with tokens. See "Replacing Credit Card Numbers with Tokens".
2. Purge the database of old untokenized card 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.

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 "Understanding Conversion Manager" in *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.

- To replace all credit card numbers with tokens, run the following command:

  ```
  pin_cc_migrate -vendor paymentProcessorName
  ```

  where *paymentProcessorName* 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 **/payinfo/cc** objects, run the following command:

  ```
  pin_cc_migrate -vendor paymentProcessorName -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 paymentProcessorName -account accountPOID
  ```

  where *accountPOID* 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 paymentProcessorName -start_date mm/dd/yy -end_date mm/dd/yy
```

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 about the utility's syntax and parameters.

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 trail objects immediately after you run the **pin_cc_migrate** utility. You can purge the old event and audit trail objects by using BRM utilities or purging scripts:

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

Note

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**.

8 Testing Paymentech Credit Card Processing

Learn how to test Paymentech credit card processing in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

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 an invalid credit card.

  Note

  The Paymentech simulator does not check for the credit card's expiration date.

The payment processing simulator is located in *BRM_home*/**bin**. It includes two utilities: **answer_s** and **answer_b**.

Caution

Use the **answer_s** and **answer_b** utilities only in test environments. In production environments, *uninstall* these utilities to prevent sensitive data from being used for verification.

- The **answer_s** utility simulates online transactions.

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

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.

Table 8-1 Example Credit Card Expiration Data

Credit Card Number	Expiration Date
4444 1111 2222 3333	0999
4101 0000 0000 0000	any expiration date

Setting Up the Paymentech Simulator

Setting up the Paymentech Ssimulator involves the following tasks:

- Defining the Credit Card Functionality to Test
- Using SFTP for Testing Batch Payment Transactions
- Using TCP/IP for Testing Batch Payment Transactions
- Specifying an IP Address for the Paymentech Simulator
- Returning Specific Values for Card Type Indicator

Defining the Credit Card Functionality to Test

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

Note

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

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.

Using SFTP for Testing Batch Payment Transactions

You can configure the Paymentech Data Manager (**dm_fusa**) to use SFTP for sending batch payment transactions to the **answer_b** server. Batch payment transactions include batch payments, multiple verifications, multiple authorizations, deposits, and refunds.

To test batch payment transactions through SFTP, follow these steps:

- Setting Up Authentication Between dm_fusa and answer_b
- Configuring Your SSH Client Configuration File for answer_b
- Testing Batch Payment Transactions Using SFTP

Setting Up Authentication Between dm_fusa and answer_b

Set up authentication between **dm_fusa** and **answer_b** by generating a public key on the **dm_fusa** machine and copying it to your Paymentech Simulator server.

To set up authentication between **dm_fusa** and **answer_b**:

1. Generate an RSA public key.

   For example, this command generates an RSA public key with a key length of 4096 bits:

   ```
   ssh-keygen -t rsa -b 4096
   ```

2. Copy the RSA public key to the Paymentech Simulator server:

   ```
   ssh-copy-id -i pathPublicKey  userName@hostNameSimulator
   ```

   where:

   - *pathPublicKey* is the path and file name of the public key generated in step 1.
   - *userName* is name of the **dm_fusa** user.
   - *hostNameSimulator* is the host name of the Paymentech Simulator server.

3. Verify the connection:

   ```
   sftp -i pathPublicKey  userName@hostNameSimulator
   ```

Configuring Your SSH Client Configuration File for answer_b

Connect **dm_fusa** to the Paymentech Simulator server by setting the following in the SSH client configuration file, located at **~/.ssh/config**:

```
Host CServer
   HostName sftpHost
   User userName
   Port portNumber
   IdentityFile pathPrivateKey
```

where:

- *sftpHost:*Host name of the Paymentech's SFTP server.
- *userName*: User name for accessing the Paymentech SFTP server.
- *portNumber*: SFTP port number for the Paymentech server.
- *pathPrivateKey*: Path and file name of the private key that matches the public key you generated in "Setting Up Authentication Between dm_fusa and answer_b".

Testing Batch Payment Transactions Using SFTP

You can configure the Paymentech DM (**dm_fusa**) to test how it sends batch payment transactions to the Paymentech Simulator. Batch payment transactions include batch payments, multiple verifications, multiple authorizations, deposits, and refunds.

To test batch payment transactions using SFTP:

1. Open the Paymentech DM configuration file (*BRM_home*/**sys/dm_fusa/pin.conf**).
2. Set the connection type between **dm_fusa** and Paymentech Simulator for batch payment transactions:

   ```
   - answer_b batch_proto sftp
   ```

3. Uncomment and configure the following entries:

   ```
   - answer_b sftp_host hostName
   - answer_b sftp_pkey_pwd passPhrase
   - answer_b sftp_indir inputPath
   - answer_b sftp_outdir outputPath
   ```

   Table 2-2 describes the values to use with the **pin.conf** entries.

   Table 8-2 Variables for the pin.conf Entries

Parameter	Description
hostName	The host section name in the SSH client configuration file.
passPhrase	The SFTP private key passphrase for the **answer_b** server. If omitted, the passphrase is retrieved from the wallet.
inputPath	The absolute path to the input directory where batch files are uploaded to the **answer_b** server using SFTP.
outputPath	The absolute path to the output directory where response files are downloaded from the **answer_b** server using SFTP.

4. Save your changes.
5. Stop and restart the Paymentech DM for the settings to take effect.

Using TCP/IP for Testing Batch Payment Transactions

To configure the Paymentech DM to test batch payment transactions over TCP/IP:

1. Open the Paymentech DM configuration file (*BRM_home*/**sys/dm_fusa/pin.conf**).
2. Set the connection type between **dm_fusa** and Paymentech Simulator for batch payment transactions:

   ```
   - answer_b batch_proto socket
   ```

3. Update 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 5678.

   ```
   - answer_b batch_srvr batchSrvr
   - answer_b batch_port batchPort
   ```

4. Save and close the configuration file.
5. 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_*** utility's **pin.conf** file.

To specify an IP address for the Paymentech Simulator:

1. Open the simulator configuration file (*BRM_home*/**apps/fusa_server/pin.conf**).
2. Do one of the following:
   - To enable Paymentech to listen to any IP address located on the machine where the **answer_*** utility is running, add the following entry to the file:

     ```
     - answer answer_name -
     ```

   - To assign a specific IP address for the **answer_*** utility, add the following entry to the file:

     ```
     - answer answer_name IP_address
     ```

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

     For example:

     ```
     - answer answer_name 192.0.2.150
     ```

Returning Specific Values for Card Type Indicator

You can define specific values to return for Card Type Indicator (CTI).

1. Open the simulator configuration file (*BRM_home*/**apps/fusa_server/pin.conf**).
2. Add the following parameters:

   ```
   - answer_s cti_resp USANYNYNYNYN
   ```

```
- answer_b cti_resp USANYNYNYNYN
```

3. Save and close the file.

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:

```
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. In the **answer_b** configuration file (*BRM_home*/**apps/fusa_server/pin.conf**), change the **v_code** entry to **502**:

   ```
   - answer_b v_code 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**.

   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 directories you specify in the **dm_fusa pin.conf** file. The **answer_b sftp_indir** entry specifies the input directory, and the **answer_b sftp_outdir** entry specifies the output directory.
6. (For SFTP only) 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".

9

Resolving Failed BRM-Initiated Payment Transactions

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

Topics in this document:

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

About Failed BRM-Initiated Payment Transactions

Failed credit card or direct debit payment transactions occur when BRM connects to a credit card processor, sends a transaction, but 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. Checkpoint records have a Paymentech result code of **888** or **999**. To resolve failed transactions, you must resolve all checkpoint records. Transactions usually have a checkpoint record for the following reasons:

- A transaction batch was received by the credit card processor, but it wasn't processed completely. To resolve this error, you 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 results from the Paymentech credit card processor (which are provided in a Paymentech RFR file).

After BRM retrieves the Paymentech RFR file, it sets the table row's value to **1000**. 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 does not have any values over **999**.

The following Paymentech result codes are used by BRM:

- PASS = 0
- FAIL_NO_ANS = 1
- FAIL_ADDR_AVS = 2
- FAIL_ADDR_LOC = 3
- FAIL_ADDR_ZIP = 4
- FAIL_CARD_BAD = 5
- SRVC_UNAVAIL = 6
- FAIL_DECL_SOFT = 7
- FAIL_DECL_HARD = 8
- FAIL_NO_MIN = 9
- INVALID_CMD = 10
- FAIL_SELECT_ITEMS = 11
- CVV_BAD = 12
- NO_CREDIT_BALANCE = 13
- FAIL_LOGICAL_PROBLEM = 14
- FAIL_FORMAT_ERROR = 15
- FAIL_INVALID_CONTENT = 16
- FAIL_TECHNICAL_PROBLEM = 17
- DEPOSIT_PENDING = 777
- AUTH_PENDING = 888
- CHECKPOINT = 999
- OFFSET = 1000

Unsuccessful transactions (result code 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 Cleared but Payment Events Not Created" to fix these transaction errors.

Note

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**.

Checking for Transaction Errors

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

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

   ```
   pin_clean -summary
   ```

   The **pin_clean** utility writes output to **stdout**.

2. Writes the output to a file.

Afterward, you can manually resolve the transaction errors. See "Resolving Transaction Errors Manually".

Resolving Transaction Errors Manually

To resolve failed transactions manually:

1. Retrieve the list of transactions that failed. See "Checking for Transaction Errors".
2. If there are many checkpoint records, limit the number of records found by running the following command:

   ```
   pin_clean -summary -search_count_limit n
   ```

3. Review the results. This 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
   ```

4. 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 you can repeat the transaction later.

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

Table 9-1 Types of Failed Credit Card Transactions

Record Type	Error	Action
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.
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.
conditional deposit	The connection was lost while running the "pin_collect" utility.	See "Resolving Failed Deposits and Conditional Deposits".
deposit	The connection was lost while running the "pin_deposit" utility.	See "Resolving Failed Deposits and Conditional Deposits".
refund	The connection was lost when a refund was made.	See "Resolving Failed Refund Transactions".

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.

5. Use the **pin_recover** utility to resubmit the batch. See "Resubmitting Transactions to Paymentech and BRM".

Resolving Failed Deposits and Conditional Deposits

To resolve failed deposits ("pin_deposit") and conditional deposits ("pin_collect"):

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 to Paymentech and BRM".

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. Go to the *BRM_home*/**bin** directory and run the **pin_recover** utility:

   ```
   pin_recover -rfr
   ```

   > **Note**
   >
   > You cannot use the **-rfr** option for online transactions such as a charge or refund made using Billing Care or Customer Center.

3. After the **pin_recover** utility has finished, run it again. Paymentech sometimes posts multiple RFR files, and you must process all of them before continuing.

   > **Note**
   >
   > Regardless of the number of times you run the **pin_recover** utility with the **-rfr** option, accounts are charged only once for each transaction.

   ```
   pin_recover -rfr
   ```

4. Run the **pin_clean** utility to check for any remaining transaction errors:

   ```
   pin_clean -summary
   ```

   If you still find transaction errors, resubmit the same batch and process the transactions that didn't go through. See "Resubmitting Transactions to Paymentech and BRM".

> **Note**
>
> 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.

Resolving Failed Refund Transactions

If the network goes down while processing a refund, verify whether the refund was processed successfully in Paymentech and BRM:

- If the refund failed in both BRM and Paymentech, 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.
- If the refund was successful in Paymentech but not in BRM, reprocess the RFR file in BRM. See "Reprocessing Failed Transactions in BRM".
- If there is no RFR file, ensure the credit card number used is the same in the failed transaction and the account's current payment method. Resubmit the batch to Paymentech and BRM. See "Resubmitting Transactions to Paymentech and BRM".

> **Note**
>
> If the credit card details used in the failed transaction are not used in the account's current payment method, running the **pin_recover -resubmit** command fails.

Reprocessing Failed Transactions in BRM

If a transaction batch was processed successfully by Paymentech but one or more payment transactions failed in BRM, reprocess the RFR file in BRM.

To reprocess failed transactions in BRM:

1. (For SFTP batch payment transactions only) Verify that the RFR file is present in the Paymentech server's output directory.

   You configure the name of the response file to recover in the **sftp_rfrfile** entry of the **dm_fusa pin.conf** file.

2. Go to the *BRM_home*/**bin** directory and run the following command:

   ```
   pin_recover -rfr
   ```

   > **Note**
   >
   > You cannot use the **-rfr** option for online transactions, such as a charge or refund made using Billing Care or Customer Center.

3. Run the **pin_clean** utility to check for remaining transaction errors:

   ```
   pin_clean -summary
   ```

   If errors remain, resubmit the same batch and process the transactions that did not complete. See "Resubmitting Transactions to Paymentech and BRM".

Resubmitting Transactions to Paymentech and BRM

If reprocessing an RFR file in BRM does not resolve all transactions in a batch, resubmit the batch to Paymentech so it can process the transactions that didn't go through. However, it's important that you resubmit transactions to Paymentech promptly, as any delay might lead to the need for reauthorization. VISA authorizations, for example, last only seven days.

> **Note**
>
> 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.

To resubmit transactions to Paymentech and BRM:

1. Find the batch ID for the batch you are resubmitting. To do so, run the "pin_clean" utility 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:
   - Enter **3** to display transactions made by running the "pin_collect" utility.
   - Enter **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. Enter **3** to quit the **pin_clean** utility.
5. Resubmit the unprocessed transactions to Paymentech:

   ```
   pin_recover -resubmit batch_ID
   ```

   For example:

   ```
   pin_recover -resubmit T,2f
   ```
6. Run the **pin_clean** utility in summary mode again:

   ```
   pin_clean -summary
   ```

   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 are probably 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, 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*.

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**)
- **/var/tmp**
- **/tmp**

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.

Configuring Delay Intervals for Resolving Payments

If the **pin_recover** utility is running in parallel with your custom payment application, duplicate transaction IDs can occur. To prevent this, configure the utility to search through charge events (**/event/billing/charge/cc**) whose creation date is older than a specified delay interval.

To configure a delay interval for resolving payments:

1. Open the billing utility configuration file (*BRM_home*/**apps/pin_billd/pin.conf**) in a text editor.
2. Search for the **event_search_delay** entry.
3. Specify the delay interval:

   ```
   pin_recover event_search_delay value
   ```

   where *value* is the delay interval in seconds. For example, if you set it to 300, **pin_recover** searches only through events that are older than 5 minutes.
4. Save and close the file.

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

The following details some problems you might encounter while trying to resolve failed credit card transactions and provides information on how to fix them:

- Cannot Remove Checkpoints After Using an RFR File
- Checkpoints Cleared but Payment Events Not Created
- Paymentech Doesn't Have an RFR File and Never Received the Payment Batch

Cannot Remove Checkpoints After Using an RFR File

If checkpoints still exist after running the **pin_recover** utility, resubmit the batch. See "Resubmitting Transactions to Paymentech and BRM" 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:

1. Delete the transactions.
2. Run the **pin_recover** utility with the **-resubmit** option:

   ```
   pin_recover -resubmit
   ```

3. Run the **pin_clean** utility with the **-summary** option to select and delete batches:

   ```
   pin_clean -summary
   ```

   Be sure to note the batch ID.
4. Run the **pin_recover** utility with the **-resubmit** option and provide the batch ID:

   ```
   pin_recover -resubmit batch_ID
   ```

Checkpoints Cleared but Payment Events Not Created

A credit card number can be reported as charged in BRM and Paymentech but not be recorded as paid in BRM. This uncommon scenario can occur when the network connection drops after Paymentech responds but before BRM allocates the payment.

To see if this has occurred, use the **testnap** utility to search the database for Paymentech result codes with a value of **1000**. For information about **testnap**, see "Testing Your Applications and Custom Modules" in *BRM Developer's Guide*.

> **Note**
>
> You cannot use the **pin_clean** utility to search for these records because the utility searches for result codes of **999** or less.

If the database contains result codes of 1000, you must create payment events for those credit card charges. To do so, run the following command:

> **Note**
>
> Because the charge has been made, this command does not affect the customer's credit card.

```
pin_recover -recover_payment
```

The payment event (**/event/billing/payment**) objects are inserted into rows in the EVENT_T and EVENT_BILLING_PAYMENT_CC_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, BRM allocates the money to open items. However, you may need to allocate the payment manually.

When a payment recovery is successful, the EVENT_BILLING_PAYMENT_CC_T value is set to **0**.

Paymentech Doesn't 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 to Paymentech and BRM" 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.

10

Processing Payment Batches in Billing Care

Learn how to use Billing Care to process externally-initiated payments, such as payments made by checks, in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

About Batches

Batches are collections of individual payments processed together for efficiency and convenience. Customers can send their payments directly to your bank or payment processor. The payment information from various customers are bundled together to include multiple payment records to be processed as a batch. Batches streamline the process of handling numerous transactions simultaneously. Batches can be of three types, **Payment**, **Refund**, and **Reversal**.

You can upload batch payment (**.pmt**) files created externally either by placing them in the appropriate directory or by uploading them manually. Billing Care validates the files against batch payment templates that are installed during Billing Care installation and process the batch file, automatically or manually, based on configuration. You can validate, manually suspend, allocate payments in the batch as well as view batch payment processing history, reverse failed batch payments, view suspended payments or imported file status.

You can create or edit batches, import and edit batch files, and validate and submit these batches in the Billing Care UI. Batches are stored in the database. You can create batches directly using the Billing Care UI and save them in your local file system, or you can import batch files manually into Billing Care. Batch files are text files containing payment information, such as account number, payment amount, and payment date, in delimiter-separated rows.

In Billing Care, the records in a batch are displayed in a table where you can add, edit, and delete records. See "Working with Individual Records in Batches" for more information.

When you import a batch file to create a new batch, Billing Care validates the imported batch file against an existing batch template. If there are no matching templates, you can edit and use an existing one or create a new template.

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)

> **Note**
>
> Fixed width is only supported when you import batches using the method in "Importing and Submitting Batch Payment Files into Billing Care Without Editing".

- 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 also specify the payment method, payment type, and qualifier, and whether to treat consecutive delimiters as one in the given data.

You can have the bank deliver the file electronically, and you can use Billing Care to import data directly from the file. See "Importing and Submitting Batch Payment Files into Billing Care Without Editing".

> **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.

Importing and Submitting Batch Payment Files into Billing Care Without Editing

You can import data from text files into Billing Care in batch payment format. When you import files using this method, you cannot edit them. 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. Each batch payment file that you import into Billing Care must have a unique file name.

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
```

After you import the batch, you can process it to validate and submit the records to BRM. If any of the records fail, you are given the option to create an exception batch. If you choose this option, a new batch is created with the failed records, and the failed records are removed from the original batch.

Creating or Importing and Editing Batches

In Billing Care, you can create a batch in the UI or you can import a batch from an existing batch file. If you import a file, make sure that the batch name in your imported batch **.pmt** file is the same as the template name. In either case, you can edit the batch in the same way. You can set the batch's characteristics, like the type of batch (payment, refund, or reversal), the number of records, and the payment method. See "Creating, Editing, and Processing Payment, Refund, and Reversal Batches" in *Billing Care Online Help* for more information. You can also edit a table containing the individual records. See "Working with Individual Records in Batches" for more information.

You can also edit failed records during submission of the batch and create a new batch with the failed records. This function allows you to identify and revise errors within a batch during its processing. See "Submitting a Batch" in *Billing Care Online Help* for more information.

Working with Individual Records in Batches

You can add, edit, manually suspend, allocate, or delete records in a table in the Create Batch page for the batch. You can import fields in the batch such as first name, last name, due amount from the file edit all of the payment, refund, or reversal information for the records in the batch, including account number, bill number, payment amount, due amount, receipt date. When you import a batch, fields such as due amount, status, first name, last name are imported from the file and are not editable in the batch. Those fields are auto-populated with the latest information when the batch is validated.

You can validate batches and the status of the validated records in the table are displayed. You can edit the failed records through this table. See "Validating a Batch" in *Billing Care Online Help* for more information on validation.

You can allocate payments to bills or items. See "Allocating Records to Bills or Items" in *Billing Care Online Help* to learn how to allocate payments.

You can manually suspend payments if you find a successful payment in a batch that you suspect contains incorrect data or requires special handling, so that it can be carefully examined before it is posted to the account. You can also remove suspension from manually suspended payments or delete records through this table.

You can also search for records through account or bill details to easily add them to the batch. See "Searching for Records to Add to a Batch" in *Billing Care Online Help* to learn how to search for any records.

Validating and Submitting Batches

After a batch is created, you can validate it. See "Validating a Batch" in *Billing Care Online Help* for instructions. After validation, a summary table is displayed with details such as validated total, suspended total, total entered in the batch, expected total and the remaining total. If any of the records fail validation, you can edit them to correct them. See "Managing Batch Entries that Fail Validation" for more information.

After validation is complete, you can:

- Edit the unvalidated entries so that they can pass validation.

> **Note**
>
> A payment can fail validation if it is less or more than the amount for a specific bill. If you have the correct permission in Billing Care (noManualAllocationMandatory), you can select to ignore this error and allocate the payment at the account level rather than the bill level.

- Export the batch to your system. See "Exporting a Batch to a File" in *Billing Care Online Help*.
- Submit the batch. See "Submitting a Batch" in *Billing Care Online Help*. This completes the processing of a batch. After a batch is submitted, you cannot make any further edits to the records. You can edit the failed records and create a new batch with those records.

Managing Batch Entries that Fail Validation

When records fail validation, and with Payment Suspense Manager enabled, their status indicates suspension upon batch submission. You can edit these records, employing the same editing process used for pre-validation changes. See "Modifying a Record in a Batch" in *Billing Care Online Help* for instructions. You must then validate the batch again before submitting it.

About Batch Templates

Batch templates are used by Billing Care to process batches of payments, refunds or reversals. Billing Care provides some default templates which are stored in the database.

Batch templates can be created, edited, or duplicated from an existing template directly in the UI.

Note

- If you have existing **.pit** files, you can enter the data from your files to the Billing Care UI. See "Creating a Batch Template" in *Billing Care Online Help* to know more.
- Make sure that the batch name in your imported batch **.pmt** file is the same as the template name.

Sample Batch Templates

BRM server contains batch payment templates which are used to process batch files.

Table 10-1 contains a list of standard batch payment templates in the Billing Care server.

Table 10-1 Standard Batch Templates

Template Type	Payment Code
Cash Payment Batch	10011
Check Payment Batch	10012
Wire-Transfer Payment Batch	10013
Inter Bank Payment Order Payment Batch	10014
Postal Order Payment Batch	10015
Failed Payment Batch	10017
Cash Refund Batch	10011
Check Refund Batch	10012
Failed Refund Batch	10017
Inter Bank Payment Order Refund Batch	10014
Postal Order Refund Batch	10015
Wire-Transfer Refund Batch	10013
Check Reversal Batch	10012
Credit Card Reversal Batch	10002
Direct Debit Reversal Batch	10005
Inter Bank Payment Order Reversal Batch	10014
Postal Order Reversal Batch	10015
Wire-Transfer Reversal Batch	10013

11

Allocating Payments

Learn 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

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 the 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, Customer Center, or your CRM application.

  – **Payment allocated to accounts with multiple bill units:** Payment is distributed to different bill units of the account based on distribution logic implemented in the PCM_OP_PYMT_POL_MBI_DISTRIBUTE policy opcode. See "Allocating Externally Initiated Payments by Due Amount" in *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 "Allocating Account Payments to Multiple Bill Units" in *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 or Customer Center, 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.

A payment batch can contain either bill-level allocations or item-level allocations, but not both. You must choose the allocation level before you create a payment batch.

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.

Allocating Payments Later

You can create a batch for only those payments that need allocations. See "Managing Batch Entries that Fail Validation".

Working with Multiple Currency Types in Billing Care

When you allocate payments with Billing Care or Customer Center, you choose the currency to use for each batch of payments. You can allocate a payment to an account in any currency. The amount is 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 "pin_bus_params" in *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.

12 Processing Atypical Payments

Learn how to configure how Oracle Communications Billing and Revenue Management (BRM) handles payments 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

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 "Payment Opcode Workflows" in *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.

See "Reversing Payments" in *BRM Opcode Guide* to create a custom application for reversing payments.

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

> **Note**
>
> 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.

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.

- 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 "Adjusting Accounts, Subscription Services, and Member Services" in *BRM Opcode Guide*.

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 "Configuring Unconfirmed Payment Processing" in *BRM Opcode Guide*.

13

Configuring Payment Methods

Learn how to configure payment methods for Oracle Communications Billing and Revenue Management (BRM) payments.

Topics in this document:

- About Payment Methods
- Default Payment Methods

To create a custom payment method, see "Adding a Custom Payment Method" in *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.

Note

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.** 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 Concepts*.

  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 "About Credit Card Payment Confirmation Numbers" 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 Concepts*.
- **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.
- **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 Charge Sharing Groups" in *BRM Managing Customers*.
- **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. 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.

Payment methods are stored in the **/config/payment** object and defined in the *BRM_home***/include/pin_pymt.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 lists the standard payment methods and element IDs.

Table 13-1 Payment Methods and Element IDs

Payment method	Element ID
PIN_PAY_TYPE_UNDEFINED Used during account creation.	0
PIN_PAY_TYPE_PREPAID Used to keep negative balances.	10000
PIN_PAY_TYPE_INVOICE Used for monthly invoices.	10001
PIN_PAY_TYPE_DEBIT Used for checking account debit.	10002
PIN_PAY_TYPE_CC Used for credit cards.	10003
PIN_PAY_TYPE_DD Used for US/Canadian direct debits.	10005
PIN_PAY_TYPE_SMARTC Used for smartcards.	10006
PIN_PAY_TYPE_SUBORD Used to roll up balances to the parent account.	10007
PIN_PAY_TYPE_BETA For use by beta testers only. Billing utilities ignore this.	10008
PIN_PAY_TYPE_INTERNAL Used for internal employees. Used the same way as guest accounts.	10009
PIN_PAY_TYPE_GUEST Used for guest accounts. It is not charged, but credit limits apply.	10010
PIN_PAY_TYPE_CASH Used for cash.	10011

Table 13-1 (Cont.) Payment Methods and Element IDs

Payment method	Element ID
PIN_PAY_TYPE_CHECK Used for personal bank checks.	10012
PIN_PAY_TYPE_WTRANSFER Used for wire transfers.	10013
PIN_PAY_TYPE_PAYORDER Used for inter-bank payment orders.	10014
PIN_PAY_TYPE_POSTALORDER Used for postal payment orders.	10015
PIN_PAY_TYPE_VOUCHER Used for payment vouchers.	10016
PIN_PAY_TYPE_FAILED Used for unconfirmed payments that failed.	10017
PIN_PAY_TYPE_SEPA Used for SEPA payments.	10018

14

Setting Up Payment Installments

Learn how to set up your customers to pay their bills in installments in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

About Installments

Your customers can pay the amount they owe in multiple installments over time. The amount your customer owes can be divided into equal or non-equal individual installments. For example, a $1,200 bill could be split equally into six monthly individual installments of $200 or unequally into one $600 installment and two monthly installments of $300.

About Defining the Terms for an Installment Plan

You define the terms for each installment plan your company offers by creating an installment specification. An installment specification defines the minimum amount for an individual installment, the types of customers and bills eligible for the installment plan, its validity dates, and the number of unequal individual installments allowed. For example, an installment specification could specify the following:

- The minimum amount for an individual installment is $100.
- Only customers from California (or a specific area) who owe more than $1000 are eligible for an installment plan.
- The installment plan is valid for a certain period, such as from January through June.
- Up to two unequal individual installments are allowed but must be scheduled at least 20 days apart.

About Creating Installments for Customers

An installment schedule is an agreement of payment between you and your customer. The customer agrees to pay the due amount in installments at certain intervals. Based on the number of installments, the due amount is divided into appropriate individual installment amounts.

The installment schedule is created on the bill or bill items using an installment specification.

By default, the total amount owed for the installment schedule is divided equally into the specified number of monthly installment payments. However, the CSRs can manually adjust the installment's effective date and amount owed for the individual installments. By default, the current date is considered the effective date for the first installment. The due date is calculated based on the interval specified in the customer’s payment agreement. The installment charge is billed immediately or is billed in the next billing cycle.

> **Note**
>
> If the individual installment amounts are changed, ensure that the total installment amount of the installment schedule is equal to the total amount owed. For example, if the bill amount is $1000, you create an installment schedule of $200 X 5 installments. The customer can decide to pay $500 in one installment, $200 in the next, $100 in the next, and $200 in the last installment. BRM allows this as long as the final bill amount adds up to $1000. However, you get an error if the installment amounts do not add up to the final bill amount.

The following scenarios explain how the installment schedule can be created:

- A customer purchases a device for $1,200 and wants to pay the amount in installments. The customer creates an installment schedule to pay for the amount owed in 12 equal individual installments of $100 per month.
- A customer has a bill amount of $2,400 and wants to pay the bill amount in non-equal installments. Based on the eligibility criteria, the customer can pay $1200 as the initial payment amount and then pay the remaining $1200 amount in 3 equal installments of $400 each.

About Installment Status

When your customer creates an installment schedule, Billing Care creates an installment schedule and an individual installment for each scheduled installment payment. For example, if a customer creates an installment schedule with three installment payments, Billing Care creates one installment schedule and three individual installments.

Billing Care indicates the status of an installment schedule with the following:

- **Open:** The customer has created an installment schedule and owes a balance.
- **Broken:** The customer missed all of the installment payments. That is, all individual installments have a **Broken** status.
- **Canceled:** The installment schedule is canceled.
- **Finished:** The customer has paid off the full amount.

Billing Care indicates the status of an individual installment with the following:

- **Open:** The individual installment has been scheduled and not charged.
- **Charged:** The installment charge has been added to the customer’s bill.
- **Broken:** The customer missed paying an installment payment. The customer’s bill is sent to collections.
- **Paid:** The individual installment is paid in full.
- **Canceled:** The individual installment is canceled.

About Partial Installment Payments

When a customer pays a partial installment amount, the individual installment remains in **Charged** status and the customer still owes the remaining balance by the due date. If the payment for the individual installment is not paid in full by the due date, the individual installment status is changed to **Broken**.

About Missed Installment Payments

When a customer misses paying an installment payment by the due date, the individual installment status changes to **Broken**. If it is the last installment, the overall installment schedule status also changes to **Broken**. If the payment is not made, the collection process takes care of the next steps as per the defined rules.

About Canceling an Installment Schedule

If a customer decides to pay any remaining amount for the installment in one shot, CSRs can cancel the remaining installments. CSRs can cancel any installment schedule with **Open** status. When an installment schedule is canceled, the **pin_installment_status_change** utility changes the status of the installment schedule from **Open** to **Canceled**. The utility also changes the status of all **Open** individual installments to **Canceled**. However, the status of any **Charged** or **Paid** individual installments is not changed and the customer still owes the amount from **Charged** individual installments by their original due date. BRM automatically applies the remaining balance from all the **Canceled** individual installments to the customer’s bill in progress as a one-time charge.

For example, a customer has a bill due amount with 6 installment payments. After paying a few (or none) of the installments, the customer decides to cancel the remaining installments by paying the remaining due amount in full. When a customer raises such a request, the CSR can cancel the installment schedule and the installment status is changed to **Canceled**.

Note

CSRs can only cancel the installments that are in **Open** status.

The following table shows an example where a customer has an open installment schedule with four individual installments. If the customer decides to cancel the installment schedule on April 10, the status of the installment schedule is changed to **Canceled**, and the status of the individual installments are changed as shown in Table 14-1.

Table 14-1 Sample Individual Installment Status

Installment Number	Effective Date	Billing Due Date	Amount Owed	Status Before Cancellation	Status After Cancellation
1	March 1	March 15	$100	Paid	Paid
2	April 1	April 15	$100	Charged	Charged
3	May 1	May 15	$100	Open	Canceled
4	June 1	June 15	$100	Open	Canceled

In this case, the customer is still required to pay the second installment of $100 by April 15th. BRM adds a $200 charge for the canceled installments to the customer’s bill in progress.

Setting Up Installments on Your System

To set up BRM to support installments, do the following:

1. Create installment specifications in Billing Care. See "Installments" in *Billing Care Online Help*.
2. Create nightly cron jobs to run the following utilities:
   - **pin_installments**

     See "Applying Installment Charges".
   - **pin_installment_status_change**

     See "Updating an Installment Status".
3. You can run the **pin_gen_notifications** utility to generate reminder messages for your customers a specified amount of time before their installment payment is due or their installment plan ends. See "Sending Messages to Customers through External Notification Applications" in *BRM Managing Customers*.
4. Perform any customization to the installment process by using the policy opcodes. See "Installment Opcode Workflows" in *BRM Opcode Guide*.
5. Create an installment schedule in Billing Care for customers who want to pay a bill in installments. See "Creating an Installment Schedule" in *Billing Care Online Help*.

Applying Installment Charges

The **pin_installments** utility applies the charges for an individual installment to your customer's bills on the specified effective date. If the installment flag is set to **Bill Now Bill**, billing is triggered immediately. If that flag is not set, the installment charges are applied to the next regular bill.

For example, a customer's installment due date is April 1st. On the scheduled date, **pin_installments** changes the installment status from **Open** to **Charged**.

Example for the utility command:

```
pin_installments -verbose
```

Updating an Installment Status

The **pin_installment_status_change** utility changes the status of the individual installment of your customer's bills, on the specified effective date.

When the payment is not received by the installment due date or when a customer makes only a partial payment of the bill amount, **pin_installment_status_change** changes the installment status from **Charged** to **Broken**. The customer's billing profile is sent to collections management to take any further actions.

pin_installment_status_change changes the status of an open installment to **Canceled**. Any remaining installment amount is included in the next bill.

When all the installments are made and no more payments due, the utility changes the installment status to **Finished**.

Once the installment status is changed from **Open** to **Charged**, **pin_installment_status_change** changes the status to **Broken**, **Canceled**, or **Paid**.

Example for the utility command:

```
pin_installment_status_change -verbose
```

Customizing Installments

You can customize installments in the following ways:

- Prepare the installment schedule specifications by using the PCM_OP_INSTALLMENT_POL_PREP_SCHEDULE_SPEC policy opcode.
- Validate the installment schedule specifications by using the PCM_OP_INSTALLMENT_POL_VALID_SCHEDULE_SPEC policy opcode.
- Validate the installment by using the PCM_OP_INSTALLMENT_POL_VALID_INSTALLMENT policy opcode.
- Prepare the installment by using the PCM_OP_INSTALLMENT_POL_PREP_INSTALLMENT policy opcode.

For more information, see "Installment Opcode Workflows" in *BRM Opcode Guide*.

15 Customizing Payment Applications

Learn how to customize payment details displayed in Oracle Communications Billing and Revenue Management (BRM) client applications.

Topics in this document:

- Customizing the Date Format for Payment Center
- Improved Performance of Searches for Payment Events in Payment Center

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 window 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:

   ```
   paymentsearch.stepsize=100
   ```
3. Save the file.

16
Implementing SEPA Payment Processing

Learn how to use Oracle Communications Billing and Revenue Management (BRM) to process Single Euro Payments Area (SEPA) payments.

Topics in this document:

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 their 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.*

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 or Customer Center 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 16-1.

   Table 16-1 Elements in the CREDITOR_INFO Child Element

Element	Description
ADDRESS	Your business street address
BIC	Your Business Identifier Code
CITY	The city where your business is located
COUNTRY	The country where your business is located
CREDITOR_ID	Your creditor identification number
CURRENCY	Your currency
IBAN	Your International Bank Account Number
NAME	Your business name
REF_PARTY	The name of your reference party
REF_PARTY_ID_CODE	The identification code of your reference party
ZIP	The postal code where your business is located

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 in Developer Center to verify that the creditor configurations are loaded.

   See "Using the testnap Utility to Test BRM" in *BRM Developer's Guide* for general instructions on using the **testnap** utility.
6. Stop and restart the Connection Manager (CM).

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 "Amending Creditor Information" in *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.

Note

- 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 16-2.

   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 16-2 pin_sepa Infranet.properties Configuration Entries

Entry	Description
infranet.connection	Specifies the connection information to connect to the BRM database.
infranet.login.type	Specifies whether a login name and password is required to connect to the BRM database. The default is **1**.
infranet.log.level	Specifies the error reporting level. The default is **1**. • **0**: No logging • **1**: Log error messages only • **2**: Log error messages and warnings • **3**: Log error, warning, and debugging messages.
infranet.log.file	Specifies the file name used to log errors. The default is **pin_sepa.pinlog**.
infranet.threadpool.size	Specifies the number of threads. The default is **3**.
infranet.threadpool.maxsize	Specifies the maximum number of threads. The default is **5**.
infranet.threadpool.fetchsize	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**.
infranet.sepa_dd_req_dir.path	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**.
infranet.sepa_ct_req_dir.path	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**.
infranet.sepa_rev_req_dir.path	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**.
infranet.sepa_resp_dir.path	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.
infranet.sepa.sddrequest.ReqdColltnDt.pattern	Specifies the date pattern for the SEPA Direct Debit request.

Table 16-2 (Cont.) pin_sepa Infranet.properties Configuration Entries

Entry	Description
infranet.sepa.sddrequest.ReqdColltnDt.value	Specifies the date on which to collect the money from the customer.
infranet.sepa.sddrequest.InitgPty.Nm	Specifies the name of the party initiating the SEPA Direct Debit request.
infranet.sepa.sddrequest.InitgPty.OrgId	Specifies the ID of the party initiating the SEPA Direct Debit request.
infranet.sepa.sddrequest.PmtInf.PmtMtd	Specifies the SEPA Direct Debit payment method. This entry must be set to **DD**.
infranet.sepa.sddrequest.InstrPrty	Specifies the instruction priority for the SEPA Direct Debit request. The default is **NORM**.
infranet.sepa.sddrequest.ChrgBr	Specifies the party who pays for the charges. The default is **SLEV**. According to the SEPA Rulebook, the only value allowed for this entry is **SLEV**.
infranet.sepa.sddrequest.PmtTpInf.LclInstrm	Specifies the Local instrument code for SEPA Direct Debit request. The default is **CORE**. • **CORE**: Core Scheme • **B2B**: Business to Business Scheme
infranet.sepa.sddrequest.PmtTpInf.SvcLvl	Specifies the service level for the SEPA Direct Debit request. The default is **SEPA**.
infranet.sepa.sctrequest.PmtInf.PmtMtd	Specifies the SEPA Credit Transfer payment method. This entry must be set to **TRF**.
infranet.sepa.sctrequest.ReqdExctnDt.pattern	Specifies the date pattern for the SEPA Credit Transfer request.
infranet.sepa.sctrequest.ReqdExctnDt.value	Specifies the date on which to credit the money to customer account.
infranet.sepa.sctrequest.InstrPrty	Specifies the instruction priority for SEPA Credit Transfer request. The default is **NORM**.
infranet.sepa.sctrequest.ChrgBr	Specifies the party who pays for the charges. The default is **SLEV**. According to the SEPA Rulebook, the only value allowed for this entry is **SLEV**.
infranet.sepa.sctrequest.InitgPty.Nm	Specifies the name of the party initiating the SEPA Credit Transfer request.
infranet.sepa.sctrequest.InitgPty.OrgId	Specifies the ID of the party initiating the SEPA Credit Transfer request.
infranet.sepa.sctrequest.PmtTpInf.LclInstrm	Specifies the Local instrument code for SEPA Credit Transfer request. The default is **CORE**. • **CORE**: Core Scheme • **B2B**: Business to Business Scheme
infranet.sepa.sctrequest.PmtTpInf.SvcLvl	Specifies the service level for the SEPA Credit Transfer request. The default is **SEPA**.
infranet.sepa.sddreversal.InitgPty.Nm	Specifies the name of the party initiating the SEPA Direct Debit Reversal request.
infranet.sepa.sddreversal.InitgPty.OrgId	Specifies the ID of the party initiating the SEPA Direct Debit Reversal request.

3. Save and close the file.

17

Configuring Payment Fees

Learn how to implement payment fees in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

For information on customizing payment fees, see "Customizing Payment Fees" in *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 such as *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 events to charge customers for payment fees and 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 is 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 payment fees are created. You can also extend the **/event/billing/payment/failed** storable class to include the added fields. See "Customizing Payment Fees" in *BRM Opcode Guide*.

Payment fees can be charged only for failures that occur due to financial reasons, and not for failures that occur due to communication errors between BRM and the payment transaction service. Communication errors are considered unresolved transactions. You run the "pin_clean" utility to find all unresolved credit card and direct debit payments recorded in the BRM database.

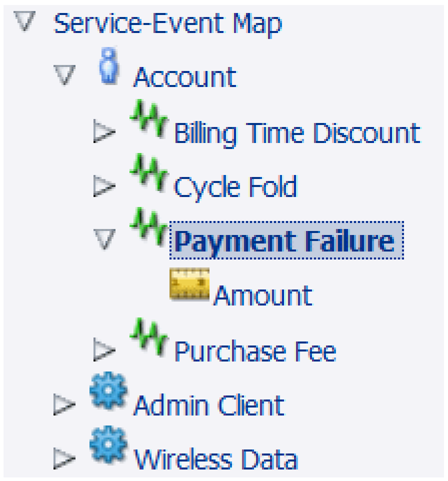
Note

If you use Payment Suspense Manager to handle failed payments, any failed payment is posted to the payment suspense account and receives a payment fee. When the payment is later fixed and posted to the correct account, the payment fee is allocated along with the payment.

Creating Payment Fees in PDC

To create a payment fee in PDC, create a payment fee charge offer:

1. In the PDC service-event map, create a Payment Failure event in the Account events:



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 "Creating and Managing Customer Segments" in *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.

Creating Payment Fees in Pricing Center

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

Note

You should be familiar with real-time rating before you begin. For information on creating rates and pricing plans, see Pricing Center Help.

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

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

   Note

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

2. Apply the product at the account level and define the purchase and ownership information.
3. In the **General Product Info** tab, type **1** in **Priority**.
4. Under Event Map, click **Add**.
   a. In the **Event** column, select **Failed Payment Fee Event**.
   b. In the **Measured By** column, select **Occurrence**.
   c. In the **Rate Plan Structure** column, select **Rate Plan Selector**.
5. Set up the rate plan for the $5 fee.
   a. Under **Rate Plan Selector**, type a name for the payment fee.
   b. Click **Edit Plans** and click **New**.
   c. Define the **Plan Details** and **Rate Plan Structure**.
   d. In the **Balance Impacts** tab, select **US Dollars [840]** as the **Resource ID** and type **5.00** in **Fixed Amount**.
   e. Click **OK**.
6. Set up the rate plan for the $7 fee. Repeat step 5, but in the **Balance Impacts** tab, type **7.00** in **Fixed Amount**.

7. Set up the rate plan selector.
   a. Click **...+** in the first column, select **Event**, choose **PIN_FLD_FAILED_PAYMENT_FEE.PIN_FLD_PAY_TYPE** from the attributes list, and click **OK**.
   b. Click **+** in the **row** column to create a row for each payment method, except Cash payments. Omitting the cash payment method from the rate plan selector excludes it from being rated and no fees are applied.
   c. In the first column of each row, type the element ID for each payment method.
   d. For the credit card, debit card, direct debit, and wire transfer payment methods, select the **$5 payment fee** rate plan.
   e. For the invoice, check, and postal order payment methods, select the **$7 payment fee** rate plan.
8. Click **OK** and **Apply**.

You can exempt accounts from receiving fees and define thresholds at which to suppress payment fees. For information, see Pricing Center Help.

18

Configuring Payment Incentives

Learn how to implement payment incentives in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

For information on customizing payment incentives, see "Customizing Payment Incentives" in *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 in PDC by creating charge offers that grant the incentives, and in Pricing Center by defining a product and rate plan.

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.

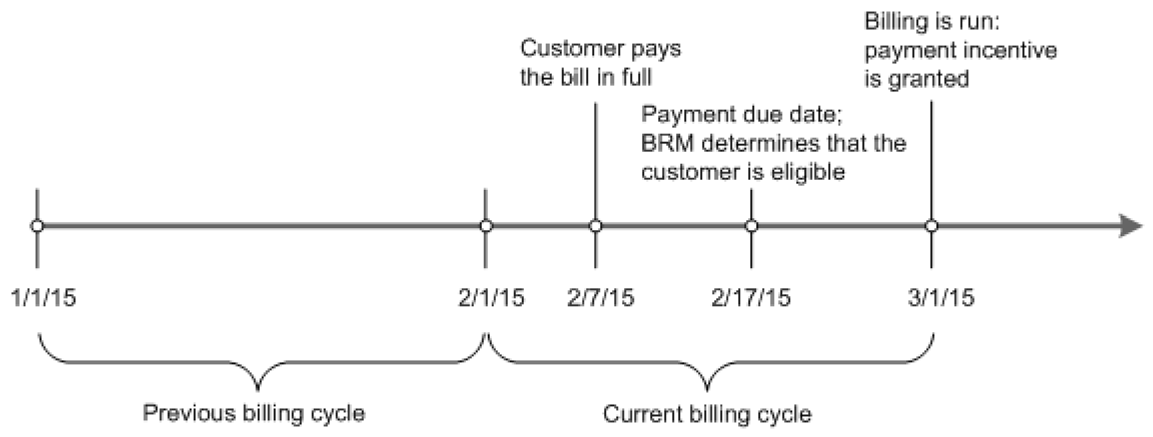
You can customize payment incentives by editing the PCM_OP_PYMT_POL_GRANT_INCENTIVE policy opcode and extending the **/event/billing/incentive** storable class. See "Customizing Payment Incentives" in *BRM Opcode Guide*.

In addition to creating incentives in Pricing Center or PDC, you must also 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 18-1 shows how a payment incentive is granted.

Figure 18-1 Payment Incentive Time Line


Customer pays the bill in full
Billing is run: payment incentive is granted
Payment due date; BRM determines that the customer is eligible
1/1/15
2/1/15
2/7/15
2/17/15
3/1/15
Previous billing cycle
Current billing cycle

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 "Reversing Payment Incentives" 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 "pin_bus_params" in *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**:

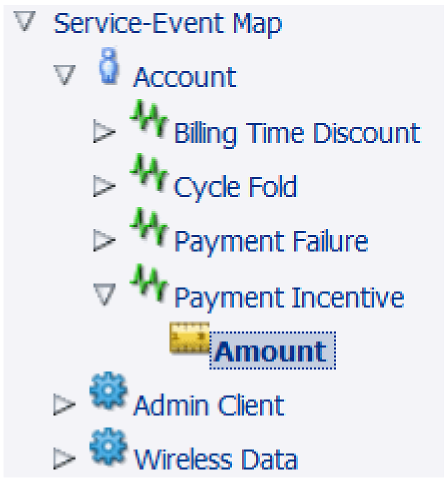
   ```
   <PaymentIncentive>enabled</PaymentIncentive>
   ```
4. Save the file as **bus_params_AR.xml**.
5. Load the XML file into the BRM database:

   ```
   pin_bus_params bus_params_AR.xml
   ```
6. Stop and restart the CM.

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 "Creating and Managing Customer Segments" in *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.

Creating Payment Incentives in Pricing Center

You create payment incentives in Pricing Center by defining products and rate plans. The rate plans you set up for payment incentives are based on the fields defined in the **/event/billing/incentive** event. You use these fields to create attribute combinations in Pricing Center that BRM compares with the actual event to determine whether it should rate the payment incentive and, if so, which rate to use.

Note

You should be familiar with real-time rating before you begin. For detailed information on creating rates and pricing plans, see Pricing Center Help.

This example uses a rate plan selector to define payment incentives based on combinations of two attributes: the payment method and customer segment:

- The payment incentive awards a 15% reduction on the total bill amount for all customers in the “Platinum Subscriber” customer segment who pay by credit card.
- It also awards a $10 reduction on the total bill amount plus 30 free minutes for customers in the “Silver Subscriber” customer segment who pay by cash.

This example includes a restriction for customers in the “Silver Subscriber” customer segment. These customers do not qualify for a payment incentive unless their total bill is over $100.

Note

A rate plan selector can only contain fields defined in the payment incentive event. For example, you can apply a payment incentive based on a customer segment, payment channel, or payment method, or a combination of these attributes.

1. Start Pricing Center and begin creating a **System** product.
2. Apply the product at the account level and define the purchase and ownership information.
3. In the **General Product Info** tab, type **1** in **Priority**.
4. Under Event Map, click **Add**.
   a. In the **Event** column, select **Payment Incentive Event**.
   b. In the **Measured By** column, select **Current Bill Total**.
   c. In the **Rate Plan Structure** column, select **Rate Plan Selector**.
5. Set up a rate plan for the 15% total bill reduction.
   a. Under **Rate Plan Selector**, type **15% Reduction** as the name for the payment incentive.
   b. Click **Edit Plans** and click **New**.
   c. Define the **Plan Details** and **Rate Plan Structure**.
   d. Define the **Quantity Discount Bracket** for the new rate as **Based on: Rate Dependent**.
   e. In the **Balance Impacts** tab, select **US Dollars [840]** as the **Resource ID** and type **–0.15** in **Scaled Amount**. Because the value for **Scaled Amount** is negative, this results in a 15% reduction. (A positive number would result in a fee.)
   f. Click **OK**.
6. Set up a second rate plan for the $10 total bill reduction plus the 30 free minutes. This reduction is applied only if the total for the current bill is over $100.
   a. Under **Rate Plan Selector**, type $10 Reduction + 30 Minutes as the name for the payment incentive.
   b. Click **Edit Plans** and click **New**.
   c. Define the **Plan Details** and **Rate Plan Structure**.
   d. Define the **Quantity Discount Bracket** for the new rate as **Based on: Rate Dependent**.
   e. In the **Balance Impacts** tab, deselect **Minimum** and type **100** in the associated entry box.
   f. Select **US Dollars [840]** as the **Resource ID** and type **–10** in **Fixed Amount**.
   g. Add a row to the balance impacts table that sets **Free Domestic Minutes** as the **Resource ID** and enter **30** in **Fixed Amount**.
   h. Click **OK**.
7. Set up the rate plan selector.
   a. Click **...+** in the first column, select **Event**, choose **PIN_FLD_INCENTIVE.PIN_FLD_PAY_TYPE** from the attributes list, and click **OK**.

b. Click **...+** in the next column, select **Event**, choose **PIN_FLD_INCENTIVE.PIN_FLD_CUSTOMER_SEGMENT** from the attributes list, and click **OK**.
c. Click **+** in the **row** column to create a row for each of the two payment method/ customer segment combinations. The system product does not provide incentives to any customers who do not meet one of these two criteria.
d. For the first row, type **10003** to define credit card as the payment method and **Platinum Subscriber** to define the customer segment. Select the **15% Reduction** rate plan.
e. For the second row, type **10011** to define cash as the payment method and **Silver Subscriber** to define the customer segment. Select the **$10 Reduction + 30 Minutes** rate plan.
f. Click **OK** and **Apply**.

19 Configuring Top-Ups

Learn how to configure and implement top-ups in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Important

Most top-up implementation tasks require a custom application. See "Managing Top-Ups" in *BRM Opcode Guide*.

About Standard Top-Ups

A standard top-up is a top-up that a customer makes to his or her account. BRM supports the following types of 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** are initiated by ECE. They occur when a resource balance falls below a specified threshold amount.

  To use 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.

- **Recurring standard top-ups** are initiated by the **pin_balance_transfer** utility when run with the **-standard** parameter.

  To use recurring standard top-ups, an account must have one or more services that are configured for top-ups. In addition, a recurring standard top-up payment method, the top-up resource, the interval between recurring top-ups, and the number of recurring top-ups to make must be set for the account.

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

- **Cash** or **check** (manual standard top-ups for topping up currency balances only)
- **Credit card** or **direct debit** (manual, automatic, and recurring standard top-ups for topping up both currency and noncurrency balances)

When a customer uses a payment card to top up his account, BRM interacts with a credit card agency or direct debit company to collect payment.

- **Voucher** (manual and recurring standard top-ups for topping up both currency and noncurrency balances)

  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.

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.

Performing Recurring Standard Top-Ups

Recurring standard top-ups are initiated by the **pin_balance_transfer** utility. When run with the **-standard** parameter, the utility finds all accounts in your system with a recurring top-up and a top-up date of today.

To perform recurring standard top-ups, go to the *BRM_home***/apps/pin_balance_transfer** directory and run the following command:

```
pin_balance_transfer -standard
```

For more information about the utility, see "pin_balance_transfer".

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** 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** 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".

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

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.

Note

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 19-1.

Table 19-1 Sponsored Top-Up Group Member Statuses

Status	Description
Active	Member account can receive top-ups from the group owner account.
Inactive	Member account's top-ups from the owner account are suspended, but member *cannot* join another sponsored top-up group.
Closed	Member account no longer receives top-ups from the group owner account. Member *can* join another sponsored top-up group.

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 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 example **crontab** entry runs **pin_balance_transfer** at 1:00 a.m. daily:

```
0 1 * * * BRM_home/bin/pin_balance_transfer &
```

For more information about the utility, see "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" ;
  STR
      ID = 4 ;
      VERSION = 1 ;
      STRING = "Sponsored Topup. Sponsor Debit" ;
      EVENT-GLID
          "/event/billing/adjustment/account"    105 ;
      EVENT-GLID-END
  END
  ```

- **Sponsored top-up** credit **reason code 5 and domain ID 8**

  ```
  DOMAIN = "Reason Codes-Credit Reasons" ;
  STR
      ID = 5 ;
      VERSION = 8 ;
      STRING = "Sponsored Topup. Sponsoree Credit" ;
      EVENT-GLID
          "/event/billing/adjustment/account"    105 ;
      EVENT-GLID-END
  END
  ```

The following definitions for these new reason codes and domain IDs 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/msgs/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 and creating new strings for it, see "Loading Localized or Customized Strings" and "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*.

Note

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:

```
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**:

```
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 is 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 19-2 lists the default error messages that are displayed in Customer Center when errors associated with the corresponding error type and field name occur.

Table 19-2 Default Error Messages in Customer Center for Top-Ups

Error Message	Error Type	Field Name
Voucher has already been used	ERR_NOT_FOUND	PIN_FLD_EXTENDED_INFO
Invalid voucher ID/PIN combination	ERR_NOT_FOUND	PIN_FLD_POID
Voucher has already been used or has expired	ERR_BAD_VALUE	PIN_FLD_STATE_ID
Voucher has expired	ERR_BAD_VALUE	PIN_FLD_EXPIRATION_T
Invalid voucher ID/PIN combination	ERR_BAD_ARG	PIN_FLD_VOUCHER_PIN
Voucher has already been used or has expired	ERR_BAD_ARG	PIN_FLD_STATE_ID

20 Configuring Loans

Learn how to configure and implement loans in Oracle Communications Billing and Revenue Management (BRM).

Topics in this document:

Note

Most loan implementation tasks require a custom application. See "Loan Opcode Workflows" in *BRM Opcode Guide.*

About Loans

Prepaid customers can opt-in to receive the following types of loan:

- **Dynamic loans** are granted when a customer's credit limit is crossed during rating. For example, if a customer's balance is $5, their credit limit is set to $0, and they try to purchase a subscription for $10, they can be granted a loan for the remaining $5. You enable dynamic loans when configuring subscription charge offers in PDC.
- **Offered loans** are offered when a customer's balance falls below a threshold set in their credit profile and are granted after the customer confirms. Loans can be offered either when the threshold is crossed during subscription or usage charging.
- **Channel loans** are granted when a customer uses an external purchase channel, such as an external app, an Unstructured Supplementary Service Data (USSD) gateway, or interactive voice response (IVR) system to use a loan to purchase a package. These channels send the purchase and loan request to BRM, and BRM checks eligibility and grants the loan without considering or consuming the customer's current balance.
  You must configure a policy opcode to grant channel loans after rating is complete. See "Granting a Channel Loan" in *BRM Opcode Guide* for more information.

You recover loans from customers from the next top-up or balance transfer they make.

When configuring loans and customer loan profiles, you can:

- Charge service fees, taxes, and late repayment fees.
- Configure loan offer thresholds.
- Set the number of days before repayment is considered late and specify what happens after; you can charge a late fee or pull the amount due from the customer's available balance.

- Configure what happens if a top-up amount is less than the full amount due for the loan, including service fees, taxes, and late fees.
- Set the maximum number of active loans for a customer.

Although loans are always for currency resources, you can optionally grant noncurrency resources along with the currency loan. You can also calculate the maximum amount to loan based on the previous month's consumption of a noncurrency resource.

Loan amounts are temporarily added to the customer's credit limit so that they do not count against them in credit checks.

About Loan Thresholds

You can set loan thresholds either as a percent of the balance, or a fixed amount. When the customer's balance falls below their loan threshold, the **/event/notification/threshold_loan** event is generated and a notification is sent to the customer to offer a loan. The notification is then resent each day until the balance is no longer below the threshold.

You configure loan thresholds for all customers when creating packages in PDC or plans in BRM using the XML Pricing Interface. When a customer purchases the package, the thresholds are stored in the **/config/credit_profile** associated with the customer's balance group. You can update the loan offer thresholds for individual customers when creating or modifying the customer's loan profile. See:

- "Setting Loan Thresholds for Packages" in *PDC Creating Product Offerings*
- "Setting Loan Thresholds for Plans" in *BRM Setting Up Pipeline Rating*
- "Creating and Modifying a Customer's Loan Profile" in *BRM Opcode Guide*

You must configure your charging application to send the notification generated for the **/event/notification/threshold_loan** event, manage the customer's response by USSD, IVR, or other self-care application, and request the loan from BRM with the PCM_OP_LOAN_APPLY_LOAN opcode or BRM REST APIs if the customer accepts the loan.

About Recovering Loans

You automatically recover loans granted to your customers the next time they top up their balance, either by making a standard or sponsored top-up. You can also customize the events for recurring charges or adjustments to call the PCM_OP_LOAN_RECOVER_LOAN opcode in the **pin_notify** file.

Any top-up balance remaining after the loan and all associated service fees and taxes have been recovered is credited to the main account balance.

If the top-up amount is less than the amount due for the loan, you can:

- Partially recover the loan using a percentage of the top-up balance, credit the remaining amount to the main account balance, and notify the customer of the remaining amount due.
- Reject the top-up and notify the customer that they must pay the full amount.

If the payment associated with a top-up is reversed, the loan recovery is also reversed.

Configuring Loans

At a high level, configuring loans involves the following:

1. Configure loans in **/config/loan** configuration objects and load them to the BRM database by using the **load_config** utility. See "Loading Your Loan Information into the BRM Database".
2. To grant noncurrency resources with loans, create charge offers for the **/event/billing/loan_grant** event. See "Configuring Charge Offers" in *PDC Creating Product Offerings*.
3. To apply loan service fees, if you did not configure the amount in the **/config/loan** object, create charge offers for the **/event/billing/loan_fee** event. You can use the loan amount as a RUM for the charge. See "Configuring Charge Offers" in *PDC Creating Product Offerings*.
4. Configure what happens during loan recovery if the top-up amount is less than the loan amount due:
   - To allocate a percentage of a top-up to loan repayment, set the **LoanRepaymentPercent** subscription business parameter. Set this parameter to a number between 0 (none of the top-up goes to repaying the loan) and 100 (all of the top-up goes to repaying the loan). The default is 100. See "Configuring BRM by Using the pin_bus_params Utility" and "business_params Reference" in *BRM System Administrator's Guide* for more information about configuring business parameters.
   - To reject the top-up completely and notify the customer of the amount due, customize the PCM_OP_LOAN_POL_PRE_RECOVER_LOAN policy opcode. This is an empty policy hook that can be called during payment and balance transfer flows for top-ups.
5. When creating subscription charge offers in PDC, specify that a loan can be applied if the customer has insufficient credit. See "Allowing Customers to Exceed Their Credit Limit" in *PDC Creating Product Offerings*.
6. When using the XML Pricing Interface to create packages in PDC or plans in BRM, set balance thresholds for offering loans to customers. See "Setting Loan Thresholds for Packages" in *PDC Creating Product Offerings* or "Setting Loan Thresholds for Plans" in *BRM Setting Up Pipeline Pricing*.
7. Configure whether customers want to receive loans, and how loans are implemented for the customer by creating loan profiles (**/profile/loan** objects). Loan profiles store the details for the current loan cycle. See "Creating a Customer's Loan Profile" in *BRM Opcode Guide*.
8. Set loan thresholds for customers in **/config/credit_profile** objects. See "Setting a Customer's Automatic Loan Threshold" in *BRM Opcode Guide*.

Loading Your Loan Information into the BRM Database

Your loan information includes parameters such as maximum and minimum amounts, minimum eligible account age, eligible locations, service fees, and tax codes. You load the loan information into **/config/loan** objects in the BRM database. You can create new **/config/loan** objects, or update existing ones.

To set up and load loan information:

1. Open the *BRM_home***/sys/data/config/config_loan.xml** file in an XML or text editor, where *BRM_home* is the directory in which the BRM server software is installed.
2. Provide the values listed in Table 20-1.

Table 20-1 Fields in the Loan Configuration Object

Field	Description
DESCR	(Optional) The loan configuration object's description.
NAME	(Required) The name of the loan configuration object. If a **/config/loan** object with this name already exists, its fields are updated with the new values provided. If it doesn't exist, a new object is created.
CREDIT_AMOUNT	(Required) The default loan amount to grant. The account's primary currency is used. If there is no amount specified on a loan request, this value is used.
LOAN_FIXED_FEE	(Optional) The fixed amount to charge as service fee. If you specify this, you cannot specify LOAN_PERCENT_FEE.
LOAN_PERCENT_FEE	(Optional) The percent of the loan to charge as a service fee. For example, if the requested amount is 10, and you set PERCENT_FEE to 10, the service fee is $1. If you specify this, you cannot specify LOAN_FIXED_FEE.
AGE	(Optional) The minimum number of days, months, or years the account must be active to be eligible for a loan.
UNIT	(Optional) The unit for the **AGE** field. Can be: • **4** (days) • **5** (months) • **6** (years)
LOAN_MINIMUM	(Optional) The minimum amount for the loan. Used only for dynamic loans. If the specified amount on the loan request is less than the minimum, the loan is rejected.
FIXED_MAXIMUM	(Optional) The maximum fixed amount to grant for each loan cycle. This amount can represent a total amount from multiple loans. For example, if the amount specified in a loan request is 10, and FIXED_MAXIMUM is set to 30, that loan could be granted a maximum of three times in the cycle. You configure the loan cycle in the **/profile/loan** object. See "Creating a Customer's Loan Profile" in *BRM Opcode Guide*. If you specify this, you cannot specify SCALED_MAXIMUM.
SCALED_MAXIMUM	(Optional) The maximum scaled amount to grant. This is a percent of the resource specified in RESOURCE_ID, which represents consumption of a noncurrency resource in the previous calendar month. For example, if you set SCALED_MAXIMUM to 50% and use the resource ID for GB of data, the maximum loan amount is calculated by determining the cost of 50% of the GB used by the customer in the previous month. If a customer pays $20 monthly for 10GB of data, and used the full 10GB in the previous month, the maximum loan amount would be $10. If you specify this, you cannot specify FIXED_MAXIMUM.
RESOURCE_ID	(Optional) The ID of a noncurrency resource to use when calculating the loan when SCALED_MAXIMUM is specified. This resource ID is unrelated to noncurrency resources you can grant with loans by the **/event/billing/loan_grant** event. Those are specified in the loan request.
MAX_QUANTITY	(Optional) The maximum number of times the loan can be granted in each loan cycle.
TAX_CODE	(Optional) The tax code for the loan service fee.
LOAN_TAX_CODE	(Optional) The tax code for the loan.
REASON_ID	(Optional) The A/R reason code to map to the G/L ID.

Table 20-1 (Cont.) Fields in the Loan Configuration Object

Field	Description
REASON_DOMAIN_ID	(Optional) The A/R reason domain to map to the G/L ID.
LOCATION_MODE	(Optional) Whether the locations specified in the LOCATIONS array are eligible for the loan (**0**) or ineligible (**1**).
LOCATIONS	(Optional) An array of locations. If LOCATION_MODE is set to 0, only the locations specified in the array are eligible for the loan. If you don't specify any locations in the array, all locations are eligible. If LOCATION_MODE is set to 1, the array is required. The locations specified in the array are not eligible for the loan.
LOCATION	(Optional) A location, specified in an array under LOCATIONS. This can be a ZIP code, city, state, or ISO-3166 country code.

3. Save and close the file.
4. Run the following command, which loads the contents of the file into the **/config/loan** object:

   ```
   BRM_home/apps/load_config/load_config -v config_loan.xml
   ```

   The **load_config** utility validates the contents using the **config_loan.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 in Developer Center to verify that the creditor configurations are loaded.
   See "Using the testnap Utility to Test BRM" in *BRM Developer's Guide* for general instructions on using the **testnap** utility.
6. Stop and restart the Connection Manager (CM).

See "Loan Configuration Examples" for examples of how to use the loan configuration fields.

Loan Configuration Examples

The following examples show how to configure different kinds of loan.

Example: Configuring an offered loan

This example shows how to configure a loan for $10 and:

- Charge a fixed fee of $0.50
- Grant the loan only to accounts in the USA that have been active for at least 6 months
- Grant a maximum worth 50% of the data used from the previous month
- Restrict customers to one loan per cycle
- Grant 1GB of data with the loan

It also shows how to set up the customer's loan profile to:

- Resets the loan cycle every 30 days.
- Offer a loan when the customer's balance reaches 5.
- Require the customer to pay the loan back 14 days after it is granted. **Repayment details**:

- Pull any outstanding balance after 14 days from the customer's available balance.
- Have a maximum of 3 active loans at a time.

To do this:

1. Create a **/config/config_loan** object and import it using the **load_config** utility. The following shows the sample XML for the **/config/config_loan** object:

```
<ConfigObject>
    <DESCR>Loan for $10</DESCR>
    <NAME>10DollarLoan</NAME>
    <LOAN_INFO>
        <CREDIT_AMOUNT>10</CREDIT_AMOUNT>
        <LOAN_FIXED_FEE>.50</LOAN_FIXED_FEE>
        <AGE>6</AGE>
        <UNIT>5</UNIT>
        <RESOURCE_ID>1000100</RESOURCE_ID>
        <MAX_QUANTITY>1</MAX_QUANTITY>
        <SCALED_MAXIMUM>50</SCALED_MAXIMUM>
        <TAX_CODE>purchase</TAX_CODE>
        <LOAN_TAX_CODE>usage</LOAN_TAX_CODE>
        <REASON_ID>2</REASON_ID>
        <REASON_DOMAIN_ID>Reason Codes - Loan Reasons</REASON_DOMAIN_ID>
        <LOCATION_MODE>0</LOCATION_MODE>
        <LOCATIONS>
            <LOCATION>US</LOCATION>
        </LOCATIONS>
    </LOAN_INFO>
</ConfigObject>
```

2. When creating subscription charge offers in PDC, specify that a loan can be applied if the customer has insufficient credit. See "Allowing Customers to Exceed Their Credit Limit" in *PDC Creating Product Offerings*.
3. Create a charge offer with a charge for the **/event/billing/loan_grant** event that has a credit balance impact of 1GB of data.
4. Create the customer's loan profile to opt-in for loans, define loan thresholds, and other settings by using the following input flist for the PCM_OP_CUST_CREATE_PROFILE opcode:

```
0 PIN_FLD_POID                        POID      [0] 0.0.0.1 /profile/loan -1 0
0 PIN_FLD_ACCOUNT_OBJ                          POID      [0] 0.0.0.0 /account
235101 0
0 PIN_FLD_SERVICE_OBJ                          POID      [0] 0.0.0.0 /service/
telco/gsm -1 0
0 PIN_FLD_SERVICE_ID                           STR       [0] "0049100098"
0 PIN_FLD_PROFILES                             ARRAY     [0] allocated 20,
used 7
1    PIN_FLD_PROFILE_OBJ                       POID      [0] 0.0.0.1 /profile/
loan -1 0
1    PIN_FLD_INHERITED_INFO                    SUBSTRUCT [0] allocated 20,
used 5
2        PIN_FLD_LOAN_INFO                     SUBSTRUCT [0] allocated
20,used 4
3            PIN_FLD_EXTERNAL_ELIGIBILITY      INT       [0] 1
3            PIN_FLD_PROFILE_NAME              STR       [0] "Loan
```

```
Configuration 1"
3            PIN_FLD_FREQUENCY            INT       [0] 30
3            PIN_FLD_UNIT                 ENUM      [0] 4
3            PIN_FLD_OPT_LOAN             INT       [0] 1
3            PIN_FLD_MAX_ACTIVE_LOANS     INT       [0] 3
3            PIN_FLD_REPAYMENT_DAYS       INT       [0] 14
3            PIN_FLD_PULLBACK             INT       [0] 1
3            PIN_FLD_LOAN_THRESHOLDS_FIXED  STR     [0] "5"
```

See *BRM Opcode Guide* for more information about using opcodes.

When the customer's balance drops below $5, they are automatically sent a notification offering them the loan. If they accept and the charging system sends the request to BRM, the following happens:

1. If the current date is later than the reset date specified in the loan profile for the loan cycle, the number of loans and credit amount for the loan cycle in the loan profile is reset.
2. BRM checks whether the customer is eligible for the loan. In this scenario, it confirms that:
   - The customer has opted in to receive loans
   - The customer's maximum number of active loans hasn't been met
   - The amount requested does not make the CREDIT_AMOUNT in the loan profile exceed the scaled maximum amount for this loan cycle. In this example, the amount must be less than the value of half of the customer's data for the previous month. If the customer currently pays $20 a month for 10GB of data, and they used all of their data in the previous month, the maximum amount of loans they can receive for this cycle is $10.
   - The number of times the loan can be granted for this cycle is not exceeded.
   - The customer's location is eligible.
   - The customer's account has been active long enough.
   - The customer's account status is active.
   - There is no missing or incorrect information, such as an incorrect resource ID.
3. The loan is granted to the eligible customer. A $10 credit is added to their account balance. Their credit limit is temporarily raised to include the credit, the fee, and tax.
4. A non-billable open item is created for the $0.50 fee.
5. General ledger information is recorded.
6. Because the request input flist contained a noncurrency resource ID, an **/event/billing/loan_grant** event is generated, and the charge offer granting the 1GB of data is purchased.
7. The customer's credit profile is updated to reflect the loan amount and number of loans granted this cycle.

Example: Configuring a dynamic loan

This example shows how to configure a dynamic loan for a currency resource and:

- Charge a fee of 10%
- Grant the loan to accounts that have been active for at least 10 days in all locations other than Alaska
- Restrict customers to two loans per cycle

To do this:

1. Create a **/config/loan** object and import it using the **load_config** utility. The following shows the sample XML for the **/config/loan** object:

   ```
   <LOAN_INFO>
       <CREDIT_AMOUNT>10</CREDIT_AMOUNT>
       <LOAN_PERCENT_FEE>10</LOAN_PERCENT_FEE>
       <AGE>10</AGE>
       <UNIT>4</UNIT>
       <LOAN_MINIMUM>2</LOAN_MINIMUM>
       <LOAN_MAXIMUM>20</LOAN_MAXIMUM>
       <MAX_QUANTITY>2</MAX_QUANTITY>
       <TAX_CODE>PURCHASE</TAX_CODE>
       <LOAN_TAX_CODE>RECURRING</LOAN_TAX_CODE>
       <REASON_ID>2</REASON_ID>
       <REASON_DOMAIN_ID>Reason Codes - Loan Reasons</REASON_DOMAIN_ID>
       <LOCATION_MODE>1</LOCATION_MODE>
       <LOCATIONS>
           <LOCATION>ALASKA</LOCATION>
       </LOCATIONS>
   </LOAN_INFO>
   ```

2. When creating subscription charge offers in PDC, specify that a loan can be applied if the customer has insufficient credit. See "Allowing Customers to Exceed Their Credit Limit" in *PDC Creating Product Offerings*.

When a customer is charged for a subscription that costs $20, but only has $12 in their account, the following happens:

1. If the current date is later than the reset date specified in the loan profile for the loan cycle, the number of loans and credit amount for the loan cycle in the loan profile is reset.
2. BRM checks whether the customer is eligible for the loan. In this scenario, it confirms that:
   - The customer has opted in to receiving loans
   - The customer's maximum number of active loans hasn't been met
   - The amount requested does not make the CREDIT_AMOUNT in the loan profile exceed the fixed maximum amount for this loan cycle.
   - The number of times the loan can be granted for this cycle is not exceeded.
   - The customer's location is eligible.
   - The loan requested is more than the minimum amount.
   - The customer's account has been active long enough.
   - The customer's account status is active.
   - There is no missing or incorrect information.
3. The loan is granted to the eligible customer. An $8 credit is added to their account balance. Their credit limit is temporarily raised to include the credit, the fee, and tax.

> **Note**
>
> Even though the CREDIT_AMOUNT configured for the loan was $10, the requested amount was only $8, so only $8 is granted. If the request did not contain an amount, the CREDIT_AMOUNT would have been used.

4. A non-billable open item is created for the $0.80 fee.
5. General ledger information is recorded.
6. The customer's credit profile is updated to reflect the loan amount and number of loans granted this cycle.

21
Managing Suspended Payments

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

Topics in this document:

Important

Only externally initiated payments can be suspended.

About Suspending Payments

A suspended payment is a payment that needs to be corrected. Payment Suspense Manager saves payments to a special *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 are 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 that are 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 repost it to the correct account.
- Allocate suspended payments to one or more accounts.

- Partially allocate a suspended payment so 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 you have realized but remain 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 Payment Center to manually suspend payments that were incorrectly posted to customer accounts and to correct suspended payments.

How you work with this application depends on whether you receive the payment as a batch file from the bank or use a payment gateway 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.

   Payment suspense can occur during account maintenance, after payments have been saved to the BRM database.
   - 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.

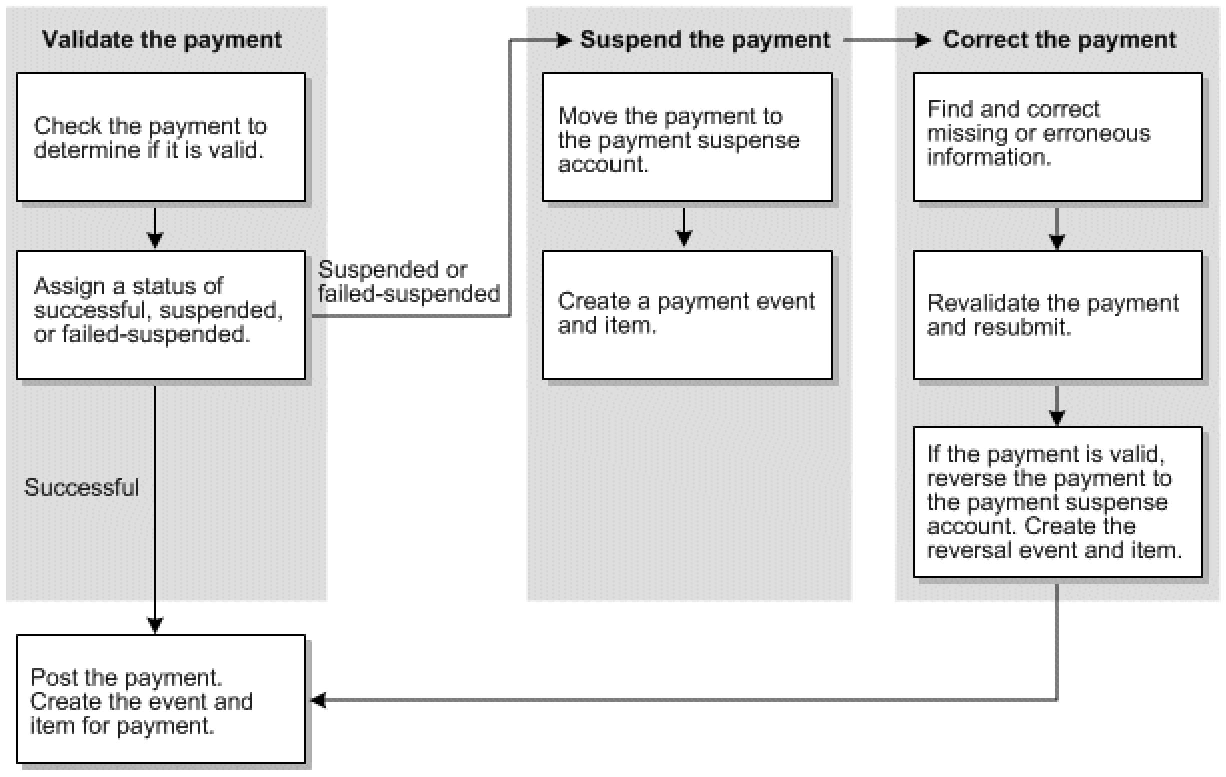
After payment analysts correct suspended payments and assign them to one or more accounts, the payments must be validated again. If the payment validation is successful, BRM posts the payments to the correct accounts. If the suspended payment is allocated completely (an amount does not remain in suspense), 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 21-1 shows the steps involved in payment suspension and the basic operations they perform.

Figure 21-1 Basic Operations and Steps Involved in Payment Suspension


Validate the payment
Check the payment to determine if it is valid.
Assign a status of successful, suspended, or failed-suspended.
Suspended or failed-suspended
Successful
Suspend the payment
Move the payment to the payment suspense account.
Create a payment event and item.
Correct the payment
Find and correct missing or erroneous information.
Revalidate the payment and resubmit.
If the payment is valid, reverse the payment to the payment suspense account. Create the reversal event and item.
Post the payment. Create the event and item for payment.

About Payment Suspense and Client Applications

The payment suspense process is initiated in one of three ways:

- Through original payments and suspended payments 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.

Payment center is the BRM client application that is used in the payment suspense process. Payment Center is used to investigate and correct suspended payments.

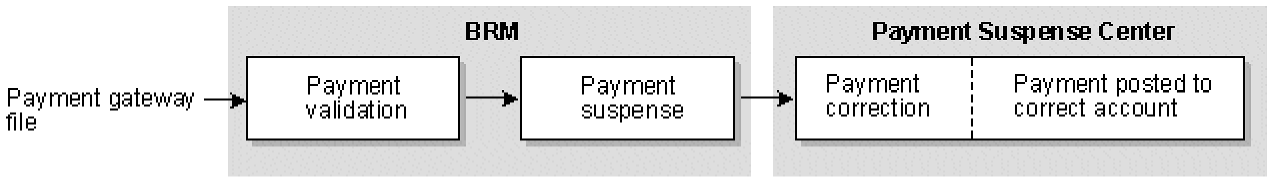
Use Payment Center for the following tasks:

- 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.

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 21-2 shows how the payment suspense process works if you use a payment gateway to process payments.

Figure 21-2 Payment Suspense Process Using Payment Gateway



In this case, the payment gateway directs BRM to perform the validation and suspense tasks. 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 information on Payment Center, see Payment Center Help.

Removing Unallocatable Payments from Suspense

Sometimes, 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 must partially distribute a suspended payment and remove the remaining suspended amount as unallocatable. If you 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 are 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 is 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 or Customer Center.
- 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

Payment Suspense Manager suspends payments exhibiting certain problems instead of failing or wrongly allocating them, and postpones them for later investigation. This enables the payment posting process to be completed without requiring immediate intervention to fix the errors. You can use the **pin_bus_params** utility to enable Payment Suspense Manager. For information about this utility, see "pin_bus_params" in *BRM Developer's Guide*.

To enable Payment Suspense Manager:

1. Go to *BRM_home***/sys/data/config**.
2. Use the following command to create an editable XML file from the appropriate instance of the **/config/business_params** object:

```
pin_bus_params -r BusParamsAR bus_params_AR.xml
```

This command creates an XML file named **bus_params_AR.xml.out** in your current directory. If you do not want this file in your current directory, specify the path as part of the file name.

3. In the file, change **disabled** to **enabled**:

   ```
   <PaymentSuspense>enabled</PaymentSuspense>
   ```

   > **Caution**
   >
   > BRM uses the XML in this file to overwrite the existing instance of the **/config/business_params** object. Use care when updating parameters in the file.

4. Save and exit the file.
5. Rename the **bus_params_AR.xml.out** file to **bus_params_AR.xml**.
6. Run the following command, which loads the updated contents of the file into the BRM database:

   ```
   pin_bus_params bus_params_AR.xml
   ```

7. Read the object with the **testnap** utility or the Object Browser to verify that all fields are correct.

   For general instructions on using **testnap**, see "Using the testnap Utility to Test BRM" in *BRM Developer's Guide*. For information on how to use Object Browser, see "Reading Objects" in *BRM Developer's Guide*.

8. Stop and restart the Connection Manager (CM).

   For more information, see "Starting and Stopping the BRM System" in *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_PYMNT_POL_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.

Note

The CSR package you select must comply with these rules:

- The admin_client service should have been used when setting up the package.
- There can be *absolutely* no bundles or charge offers attached to the package.

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 Center**: Provides action owner lists that payment personnel can choose from when assigning a person to correct a payment or use as a criterion 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:

- 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.

Note

You should not assign G/L IDs for action owner codes, and there is no need to assign G/L IDs for the reason codes for suspended payments.

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

```
LOCALE = en_US
DOMAIN = "Reason Codes - Payment Suspense Management";
STR
  ID = 2001;
  VERSION = 14;
  STRING = "Account No not found.";
  HELPSTR = "Account Number not found in database"
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"
STR
  ID = 102,002;
  VERSION = 15;
  STRING = "Micheal Orden";
  HELPSTR = "Payments Processing department"
END

DOMAIN = "Reason Codes - Payment Suspense Management Reversal Reason";
STR
  ID = 4999;
  VERSION = 16;
  STRING = "Unable to correct payment";
  HELPSTR = "Unable to correct payment."
  EVENT-GLID
    "/event/billing/reversal"         112;
  EVENT-GLID END
STR
   ID = 110,000;
   VERSION = 16;
   STRING = "Payment recycled to suspense";
   HELPSTR = "Payment moved from wrong customer account to payment suspense account"
   EVENT-GLID
    "/event/billing/reversal" 113;
   EVENT-GLID END
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 A/R Actions" 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***/sys/msgs/reasoncodes** directory. You then use the **load_localized_strings** utility to load the contents of the file into the **/strings** and **/config/map_glid** objects.

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

```
load_localized_strings reasons.locale
```

Note

- If you are loading a localized version of this file, use the correct file extension for your locale. For a list of file extensions, see "Locale Names" in *BRM Developer's Guide*.
- 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 and creating new strings for it, see "Loading Localized or Customized Strings" and "Creating New Strings and Customizing Existing Strings" in *BRM Developer's Guide*.

22

Configuring Payment Channels

Learn how to define and load payment channel IDs and descriptions into the Oracle Communications Billing and Revenue Management (BRM) database.

Topics in this document:

About Payment Channel Information

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

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.

Setting Up Payment Channel Information

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

For BRM-initiated payment, the payment gateway must include the payment channel information with each payment. When BRM receives the payments, they are processed automatically using the correct channel ID.

For externally initiated payments, the payment gateway must map the external payment channel information to the BRM channel IDs within each payment file. It is essential that the payment channel information is already included in the imported payment batch. If a payment does not have a specific payment channel ID, the channel ID at the payment batch level is used for that transaction.

Note

By default, BRM does not verify the accurate mapping of payment channel IDs.

Defining Payment Channel Information in BRM

The payment channel information you load into the BRM database consists of payment channel IDs and the text strings that describe them. To define payment channel IDs, you edit the **payment_channel.en_US** sample file in the *BRM_home***/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" in *BRM Developer's Guide*.

For information about loading the **payment_channel.***locale* file and creating new strings for it, see "Loading Localized or Customized Strings" and "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).
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** is 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 to load the channel ID into BRM.

You can run the **testnap** utility to check that the payment channel IDs were loaded properly.

23 Customizing Payment Collection Dates for BRM-Initiated Payments

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

Topics in this document:

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 "Calculating Payment Collection Dates" in *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.

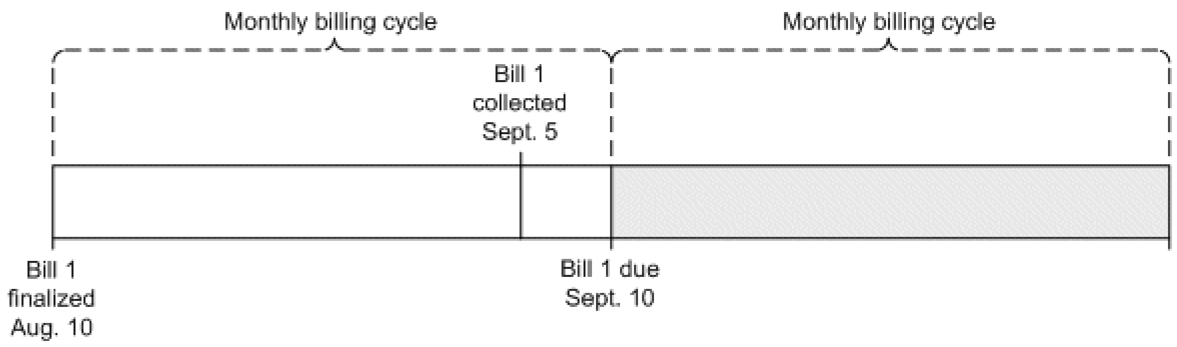
About Configurable Payment Collection Dates and On-Purchase Billing

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 23-1)
- Due date = September 10 (August 10 + 31 days)
- Payment collection date = September 5 (September 10 - 5 days)

Figure 23-1 Regular Billing Cycle Dates

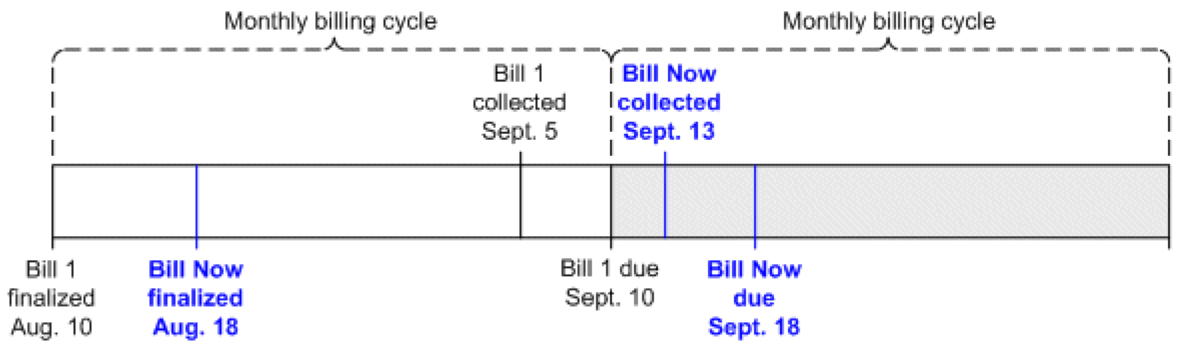


The Bill 1 payment collection date (September 5) is stored in the bill unit associated with Bill 1.

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

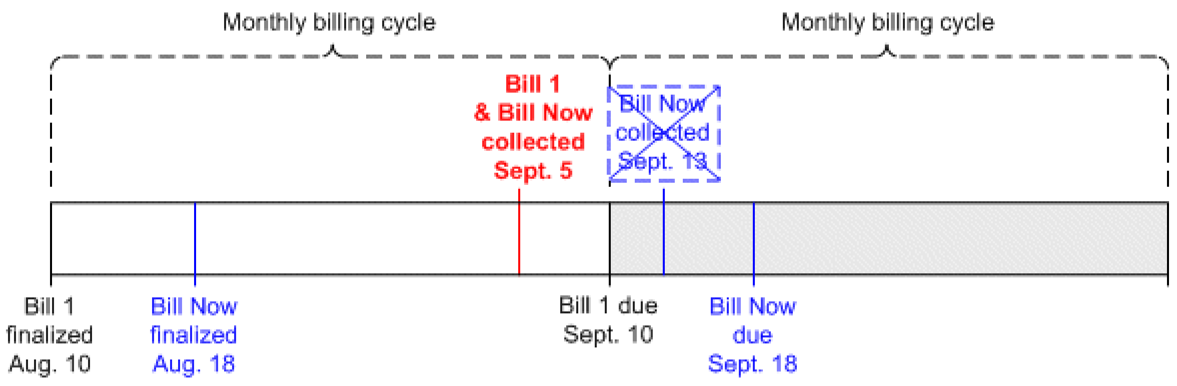
- Bill finalized = “Bill Now" (see Figure 23-2)
- Due date = September 18 (August 18 + 31 days)
- Payment collection date = September 13 (September 18 - 5 days)

Figure 23-2 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 23-3.

Figure 23-3 Bill Now Payment Collection Date



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" in *BRM Configuring and Running Billing* and "On-Purchase Billing" in *BRM Opcode Guide*.

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 occurs 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*.

24

Processing Payments in a Multischema System

Learn how to collect payments in an Oracle Communications Billing and Revenue Management (BRM) multischema system.

Topics in this document:

Processing Payments in a Multischema System

You can use Payment Center in a multischema environment. This support for multischema payment processing allows you to:

- Create and submit a single payment batch that includes payments made to customer accounts across multiple schemas.
- Create and submit a single reverse or refund payment batch for payments associated with customer accounts in various schemas.
- Move suspended payments to the designated payment suspense account set up for the connected schema.
- Recycle suspended payments to one or more customer accounts across multiple schemas using Payment Center. For example, if you are connected to schema 1, you can recycle payments from the payment suspense account for schema 1 to customers accounts located in different schemas.

To enable multischema support for payment processing, do the following:

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".

25 Payment Utilities

Learn about the Oracle Communications Billing and Revenue Management (BRM) payment utilities.

Topics in this document:

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 or recurring standard top-ups.

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

Location

BRM_home/**bin**

Syntax

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

Parameters

-test
Tests the utility, but does not affect accounts. Use this parameter to see which accounts receive automatic sponsored top-ups without actually transferring funds from group owner accounts to member accounts.

-start *mm/dd/yy* or *yyyy*
Specifies to perform top-ups scheduled on or after the specified start date.

-end *mm/dd/yy* or *yyyy*
Specifies to perform top-ups scheduled on or before the specified end date.

-standard
Performs standard recurring top-ups. See "About Standard Top-Ups".
The **-start** and **-end** parameters cannot be used with the **-standard** parameter.

-verbose
Displays information as the utility runs.

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".

Note

- Ensure that the outstanding payments for credit card accounts are closed before running this utility.
- The utility does not tokenize PINless debit card numbers.

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
Returns a list of the **/payinfo/cc** objects for which tokenization has been completed. This parameter must be used with the **-verbose** parameter.

-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".

Location

BRM_home/**bin**

Syntax

```
pin_clean   [-summary] [-search_count_limit n] [-auth_pending] [-schema 
schema_number]
            [-verbose] [file_name.log] [-help]
```

Parameters

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

-schema *schema_number*
Specifies the schema on which to run the utility on multischema BRM systems: **1** specifies the primary schema, **2** specifies the second schema, and so on. By default, the utility runs on all schemas.

-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/ddlyy* 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/ddlyy* 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 transactions and PINless 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.

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[file_name.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_installment_status_change

Use this utility to update a customer’s installment schedule status. This utility searches for installments with a due date on or before the current date and the installment schedule status **Charged**. When it finds an installment that meets the criteria, the utility changes the installment schedule status to one of the following:

- **Broken**: If the installment amount is not paid by the specified due date or if only a part of the installment amount is paid, this utility changes the installment schedule status to **Broken**.
- **Canceled**: If the customer decides to cancel an installment schedule, this utility is used to change the installment schedule status to **Canceled**. When this happens, any remaining installments are canceled, and any remaining installment amount gets reflected in the next bill.

Location

BRM_home/**bin**

Syntax

```
pin_installment_status_change  [-verbose] [-test] [-help]
```

Parameters

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

-test
Tests the utility but does not affect the installment. Use this parameter to check if the status is changed correctly.

-help
Displays the syntax and parameters for this utility.

Results

If the **pin_installments_status_change** utility does not notify you that it was successful, look in the utility log file (**pin_installments_status_change.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_installments

Use this utility to apply the installment charges to the customer's bills. This utility searches for installment schedules with an effective date on or before the current date and the installment status as **Open**. When it finds an installment that meets the criteria, the utility applies the installment charges to the customer's bill and changes the installment's status to **Charged**.

> **Note**
>
> To connect to the BRM database, this utility needs a configuration (**pin.conf**) file in the directory from which it is being run. For information about creating configuration files for BRM utilities, see "Connecting BRM Utilities" in *BRM System Administrator's Guide*.

Location

*BRM_home***/bin**

Syntax

```
pin_installments  [-help] [-verbose] [-test]
```

Parameters

-help
Displays the syntax and parameters for this utility.

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

-test
Tests the utility but does not affect the installment. Use this parameter to check if the installment charges are applied correctly.

Results

If the **pin_installments** utility does not notify you that it was successful, look in the utility log file (**pin_installments.pinlog**) to find any errors. The log file is either in the directory from which the utility was started, or in a directory specified in the configuration file.

pin_recover

Use this utility to resolve failed credit card and direct debit transactions. See "Resolving Failed BRM-Initiated Payment Transactions" for more information.

Location

*BRM_home***/bin**

Syntax

```
pin_recover   -pay_type payment_method -vendor payment_processor_name
             [-schema schema_number] [-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.

-schema *schema_number*
Specifies the schema on which to run the utility in multischema BRM systems: **1** specifies the primary schema, **2** specifies the second schema, **3** specifies the third schema, and so on. By default, the utility runs on all schemas.

-rfr
Retrieves the Request For Response (RFR) file from Paymentech, allocates successfully processed payments to the appropriate accounts' open items, and records the payments as completed in BRM. See "Reprocessing Failed Transactions in BRM" for more information.

-resubmit *batch_ID*
Resends the original batch with the same batch ID to Paymentech for processing. To find the batch ID, run the **pin_clean** utility. See "Resubmitting Transactions to Paymentech and BRM" 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 Custom Pay Types" 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:

*typedbno-YYYYMMDD-X***.xml**

where:

- *type* is **SDD** for SEPA Direct Debit, **SCT** for SEPA Credit Transfer, or **SDD-REV** for SEPA Direct Debit reversal.
- *dbno* is the database number.
- *YYYYMMDD* is the year, month, and day on which the file was generated.
- *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 **Infranet.properties** configuration file.

A

About Payment Statuses

Learn about Oracle Communications Billing and Revenue Management (BRM) payment attributes.

Topics in this document:

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

Table A-1 lists the status codes assigned by default.

Table A-1 Default BRM Status Codes and Descriptions

Value	Default Code	Description
PIN_PYMT_SUCCESS	0	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.
PIN_PYMT_WRITEOFF_SUCCESS	10	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*.
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 was 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

Value	Default Code	Description
PIN_PYMT_RECYCLED_SUSPENSE	17	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.
PIN_PYMT_RETURNED_SUSPENSE	19	The payment was distributed to a customer account from the payment suspense account but was then resuspended.
PIN_PYMT_FAILED	30	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.
PIN_PYMT_STATUS_MAX	45	Payments with a value equal to or greater than 45 are not supported by BRM.