Administration Guide for Oracle Siebel ePayment Manager

Version 4.7, Rev. A

October, 2010

ORACLE

Contents

1 Preface

About Customer Self-Service and eaSuite™

Oracle has developed the industry's most comprehensive software and services for deploying Customer Self-Service solutions. **eaSuite**™ combines electronic presentment and payment (EPP), order management, knowledge management, personalization and application integration technologies to create an integrated, natural starting point for all customer service issues. eaSuite's unique architecture leverages and preserves existing infrastructure and data, and offers unparalleled scalability for the most demanding applications. With deployments across the healthcare, financial services, energy, retail, and communications industries, and the public sector, eaSuite powers some of the world's largest and most demanding customer self-service applications. eaSuite is a standards-based, feature rich, and highly scalable platform, that delivers the lowest total cost of ownership of any self-service solution available.

eaSuite consists of four product families:

- Electronic Presentment and Payment (EPP) Applications
- Advanced Interactivity Applications
- Enterprise Productivity Applications
- Development Tools

Electronic Presentment and Payment (EPP) Applications are the foundation of Oracle's Customer Self-Service solution. They provide the core integration infrastructure between organizations' backend transactional systems and end users, as well as rich e-billing, e-invoicing and e-statement functionality. Designed to meet the rigorous demands of the most technologically advanced organizations, these applications power Customer Self-Service by managing transactional data and by enabling payments and account distribution.

- **eStatement Manager** is the core infrastructure of enterprise Customer Self-Service solutions for organizations large and small with special emphasis on meeting the needs of organizations with large numbers of customers, high data volumes and extensive integration with systems and business processes across the enterprise. Organizations use eStatement with its data access layer, composition engine, and security, enrollment and logging framework to power complex Customer Self-Service applications.
- **ePayment Manager** is the electronic payment solution that decreases payment processing costs, accelerates receivables, and improves operational efficiency. ePayment Manager is a complete payment scheduling and warehousing system with real-time and batch connections to payment gateways for Automated Clearing House (ACH) and credit card payments, and payments via various payment processing service providers.

Oracle's **Development Tools** are visual development environments for designing and configuring Oracle's Customer Self-Service solutions. The Configuration Tools encompass data and rules management, workflow authoring, systems integration, and a software development kit that makes it easy to create customer and employee-facing self-service applications leveraging eaSuite.

About This Guide

ePayment Manager plugs into eStatement Manager to provide a generic, plug-and-play platform that is capable of supporting a wide range of electronic payment methods including ACH, credit card, and CheckFree CDP transactions.

This guide describes how to configure ePayment Manager in the Command Center, and how to define payment gateways. It also describes how to use ePayment Manager in a production environment.

Related Documentation

This guide is part of the ePayment Manager documentation set. For more information about implementing your payment application, see one of the following guides:

Print Document	**Description**
Installation Guide for Oracle Siebel ePayment Manager	How to install and configure ePayment Manager on your system.
Customizing and Extending Oracle Siebel ePayment Manager	How to develop ePayment Manager applications, and extend the functionality of ePayment Manager Command Center jobs.

2 Overview of ePayment Manager

ePayment Manager is the electronic payment solution that decreases payment processing costs, accelerates receivables, and improves operational efficiency. ePayment Manager is a complete payment scheduling and warehousing system with real-time and batch connections to payment gateways for Automated Clearing House (ACH) and credit card payments, and payments via various payment processing service providers.

ePayment Manager Benefits

- Decreases payment processing costs and improves efficiency by providing complete electronic payment flexibility and low cost payment options to automate the payment process.
- Accelerates receivables and decreases float by automating payments online.
- Increases satisfaction and reduces customer support costs by allowing customers to easily and conveniently make payments on their accounts at the organization's Website.
- Minimizes IT costs by eliminating "hard wired" links to payment providers and having to support changing/emerging standards.
- Integrates with accounts receivables systems to automate electronic payment remittance postings.

ePayment Manager Key Features

- **Connections to payment networks**

  Real-time and batch interfaces to ACH, Credit Card, and proprietary networks, using a cartridge based approach that yields complete payment flexibility.

- **Advanced warehousing and scheduling**

  Full payment warehousing to manage all of the scheduling, transaction, and business logic. Make one-time instant payments, schedule future payments, set up recurring and "auto-pay" payments, utilize threshold functionality, and cancel/change payments.

  Supports ACH Notification of Changes (NOC), ACH addenda records, and multiple billers in one ACH file. Demand deposit account (DDA) verification before a payment is submitted via pre-notes.

  Once ePayment Manager retrieves an invoice from eStatement Manager, it keeps it in the ePayment Manager database. That allows customers to view invoices to make payments and view payment history.

- **Integration with your existing infrastructure**

  Updates Accounts Receivables systems with remittance info and supports reconciliation processes. Includes XML based API's for integration into backend systems.

- **Front-end GUIs**

  Includes fully functional front-end Web pages, which can also be used as templates, enabling you to fully brand and customize your front-end interface.

  Account history and access to details of past payments, providing an integrated view of all transactions, regardless of payment type or who initiated them

  Payment reminders and a variety of customizable email templates available to the administrator as well as the end-user. Examples of email notification include enrollment status, recurring payment scheduling, and bill payment status.

- **Easy-to-use administration tools**:

  Web-based configuration

  Integration with the eStatement™ Command Center

  Customer information management

  Monitor system activities and generate reports

- **Database optimization for high-performance and scalability**

- **Rich SDK**

  Enables you to fully extend the solution, including API's for two-way access and customizable front-end screens, jobs, and processes

Transaction Manager and Payment Cartridges

ePayment Manager provides a payment transaction manager and several payment cartridges. The payment transaction manager provides generic payment transaction processing capabilities. Depending on the payment type, the payment transaction manager communicates with a payment cartridge using different payment objects such as a check or credit card.

A payment cartridge communicates with a payment gateway, such as ACH or CheckFree. The cartridge translates a payment object to a format understandable by the payment processors.

Payment uses the following three payment cartridges:

- **ACH -** The ACH payment cartridge transforms bill payments to National Automated ClearingHouse Association (NACHA 2001) file formats. This payment cartridge supports the batch-oriented electronic funds transfer system governed by the ACH operating rules. ACH payment cartridges generate outbound files that are sent to ACH and processes inbound files that are returned from ACH. Payment supports PPD, CCD, and WEB formats.

  Part of configuring an ACH payment gateway involves creating inbound and outbound directories to store files, then specifying the pathnames to these directories in the ACH payment gateway configuration form.

- **Credit Cards -** Each credit card processor requires a specific credit card payment cartridge to process credit card payments. ePayment Manager provides a payment cartridge for VeriSign, which processes real-time online authorization and payment using VeriSign's PayflowPro.

- **CheckFree CDP -** CheckFree CDP (CheckFree Direct Payment) consists of a batch-mode transfer of a series of outbound files that contain payment transactions and the subsequent receipt of a series of inbound files that contain the results of these payment transactions. Although CheckFree offers several payment methods, CDP is the only CheckFree payment method supported at this time.

  Part of configuring a CheckFree payment gateway involves creating an inbound and outbound directory for CheckFree inbound and outbound files, then specifying the pathnames to these directories on the CheckFree payment gateway configuration page. Using the CheckFree payment cartridge you can also schedule future payments.

  CheckFree uses FTP over the Internet to transfer files, so ePayment Manager must also have access to the Internet from the production environment. CheckFree also uses PGP file encryption for security; you must manage PGP encryption.

ePayment Manager Jobs

ePayment Manager jobs transfer information between a biller and a payment gateway, and perform general maintenance tasks associated with an online billing system. You configure ePayment Manager jobs in the Command Center.

When naming a payment application, you must adhere to the following conventions. The application name:

- Can contain alpha-numeric characters only
- Cannot contain any spaces
- Cannot begin with a number
- Cannot contain dashes or hyphens

You can create and configure these **regular payment jobs** for use with ePayment Manager:

Job Type	Description
pmtSubmitEnroll	Submits enrollment information to a payment gateway. Currently this only applies to the ACH payment gateway.
pmtConfirmEnroll	Activates pending payment accounts after three days (by default), as long as there has been no error returned. Applies to ACH only.
pmtNotifyEnroll	Sends email notification to customers about the status of their payment account activation, plus any changes to their payment account.
pmtARIntegrator	Creates a file in a variety of formats that can be read by an Accounts Receivable system.
pmtCheckSubmit	Submits scheduled check/debit payment transaction requests to an ACH or CheckFree payment gateway.
pmtCheckUpdate	Updates a check's status according to the response from a payment gateway. For ACH it also processes check returns, prenote returns and NOC returns.
pmtCreditCardSubmit	Submits scheduled credit card payments to a credit card gateway.
pmtCreditCardExpNotifiy	Sends emails to users whose use a credit card for payments to warn them that their credit card is about to expire.
pmtPaymentReminder	Sends payment reminder email notifications to customers.
pmtRecurringPayment	Schedules payments on a recurring basis, as defined by the user.
pmtAllCheckTasks	Runs all the ePayment Manager jobs sequentially, in the required order.
pmtCustom	This job must be customized using the procedures described in the Customizing and Extending Oracle Siebel ePayment Manager.

You can create and configure these **external payment job types** if you enable payments to **external billers**:

Job Type	**Description**
pmtCheckSubmit	Submits scheduled check/debit payment transaction requests to an ACH or CheckFree payment gateway.
pmtCreditCardSubmit	Submits scheduled credit card payments to a credit card gateway.

See Chapter 9, *Configuring ePayment Manager Jobs* for details on configuring ePayment jobs.

Scheduling ePayment Manager Jobs

Schedule payment jobs to run when there is not much customer activity, such as at midnight.

If two jobs access the same table, schedule them to run at different times. For example, run pmtCheckSubmit, pmtCheckUpdate and pmtPaymentReminder at different times, and allow enough time between jobs so they won't both be trying to access the database at the same time. Failing to do this could cause a database access error. pmtSubmitEnroll, pmtConfirmEnroll and pmtNotifyEnroll should also run at different times. The best solution is to use the pmtAllCheckTasks job, or the job chain feature of eStatement that creates a chain of jobs. Both ensure that no two jobs run at the same time.

For check jobs, run pmtRecurringPayment, pmtCheckSubmit, pmtCheckUpdate and pmtPaymentReminder in order. For enroll jobs, run pmtSubmitEnroll, pmtConfirmEnroll and pmtNotifyEnroll in order. There is only one credit card job, pmtCreditCardSubmit, which should run before the pmtPaymentReminder job.

ePayment Manager Job Status Monitoring

When a payment job completes, an email can be sent to the administrator about the status of the job. This feature is enabled in the Payment Settings for each payment gateway.

ePayment Manager Job Plug-In

The following payment jobs support plug-ins, which allow you to extend core ePayment Manager functionality:

- pmtCheckUpdate
- pmtPaymentReminder
- pmtRecurringPayment
- pmtCreditCardExpNotify

Auditing, Reporting and Logging

ePayment Manager stores transactional information in the following ways:

- **Audit Trace** - ePayment Manager keeps an audit trace for each payment transaction as the transaction goes through multiple states.

- **Payment History** - User payments are logged, which the user can view using the Payment History and External Payment History options in the sample application.
- **Command Center Reports** - ePayment Manager logs warnings and errors that can be viewed through Command Center reports. Based on the format of the logging messages, ePayment Manager can hook up with other monitor systems, such as EMC patrol.

Audit Trace

Check audit traces are stored in the *check_payments_history* table (note that even though the table name contains the word history, it is not used for payment history, it is used for auditing trace). For each check payment in *check_payments* table, there may be multiple entries in the *check_payments_history* table, each corresponding to a state the check was in. For example: a check is scheduled, then processed and then paid. Those state changes create three entries in the *check_payments_history* table: the first has status 6, the second 7, and the third has 8. All entries have the same paymentID.

Credit card audit traces are kept in the *creditcard_payments_history* table.

The information in the history tables is not used by the sample UI or payment jobs. It is only used for auditing or debugging purposes, to allow an administrator to find out what’s going on with a particular payment. For example, the administrator may want to know when the payment is processed and when it is returned.

Payment History

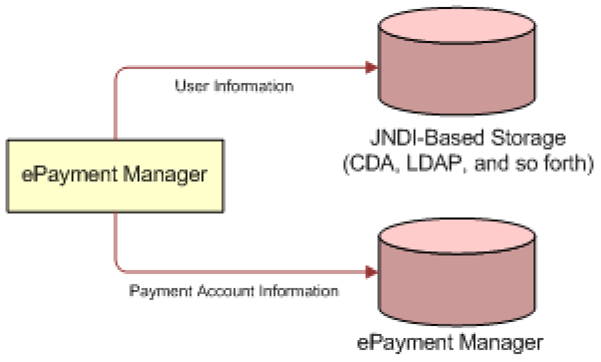
Check payments history is stored in the *check_payments* table, and credit card payments history is stored in the *creditcard_payments* table. Users can view their payment history from the sample application.

3 Enrollment

Enrollment Overview

Different enrollment models are used, depending on how the enrollment data is stored and retrieved.

By default, user information is stored in the eStatement/ePayment database or an external repository. By contrast, payment account information is always stored in the *payment_accounts* table. The following diagram shows the default enrollment model:



Single- and Multiple-DDN Models

Choose single or multiple Data Definition Name (DDN, also referred to as biller or statement provider) based on whether the user will have only one biller with one account, or multiple billers with multiple accounts.

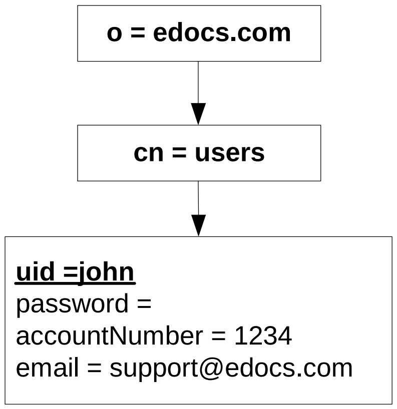
These models are described here, and can both be demonstrated by the paymentComplex sample application, which is packaged as *ear-payment-complex.ear* and can be accessed as http://host:port/paymentComplex/Payment?app=Payment.

Single-DDN Model

In the single-DDN model, a single user can have only one account number per DDN.

This model is compatible with the eStatement Sample application. Therefore, a user who enrolls through the Sample application should be able to login to the paymentComplex application, and vice versa.

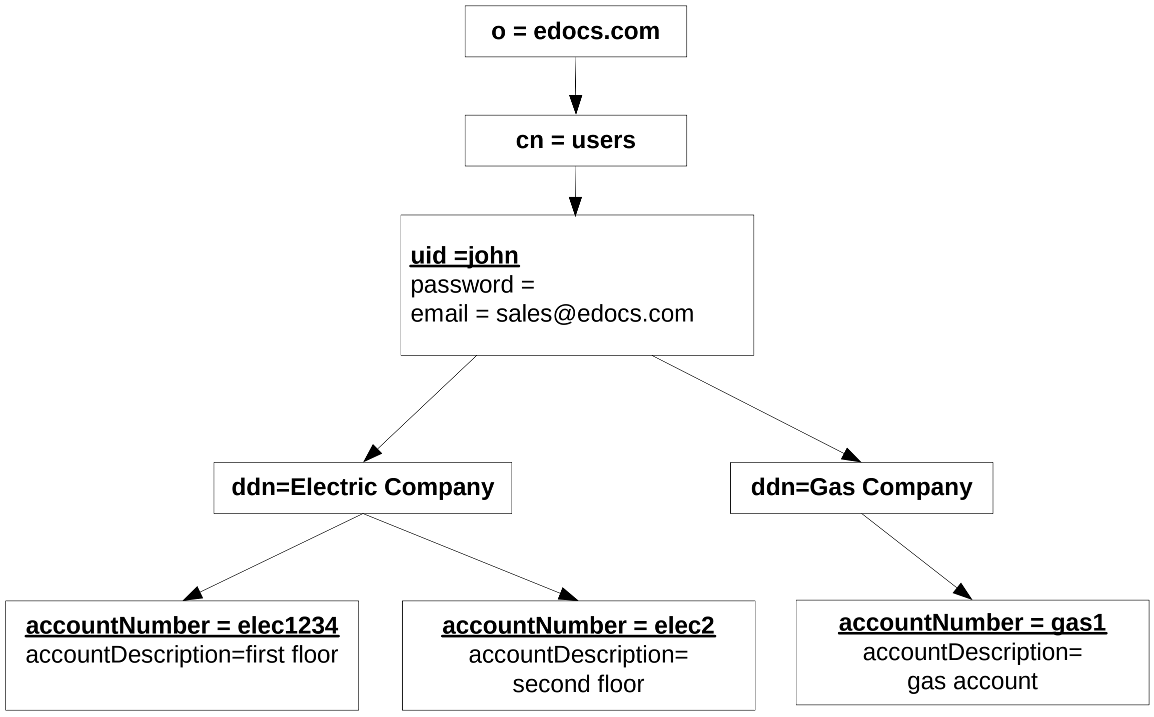
There are different ways to implement this enrollment model. This model is based on eStatement's CDA enrollment and its CDA schema, which is shown the following diagram:



Multiple-DDN Model

In the multiple-DDN model, a single user can enroll with multiple billers and can have more than one account with each biller. This essentially implements a hierarchy of user accounts.

There are different ways to implement this enrollment model. The paymentComplex implementation is based on eStatement CDA enrollment, and its CDA schema is shown the following diagram:



Consult Oracle Professional Services to design the enrollment model that best suits your needs.

ePayment Manager Enrollment

ePayment Manager requires that an existing user be enrolled with eStatement Manager. eStatement Manager presents bills or account statements to that user, and ePayment Manager provides the ability to make payments against those accounts. ePayment also enrolls users in the payment system.

The payment enrollment cycle for each account type is described in the sections describing that account.

A user can enroll with ePayment without specifying an account. That allows the user to make instant credit card payments.

1. A new customer enrolls for payment services by completing an enrollment form in the user interface. Then the customer adds bank account information selecting the enrollment status as "pnd_active". Payment saves the information in the payment_accounts table with an enrollment status of "pnd_active".
2. The pmtSubmitEnroll job runs to submit the enrollment information to the payment gateway. It changes the enrollment status to "pnd_wait".
3. The pmtConfirmEnroll job runs. This job updates the status of the customer enrollment to "active" if there are no problems after a specified number of days (by default, three days).

   If the payment enrollment information is not correct, the pmtConfirmEnroll job updates the customer enrollment status to "bad_active". An exception report is created, which can be viewed from the Command Center.
4. The customer may optionally receive an email about enrollment status from the pmtNotifyEnroll job.

4 Check Payments

Check Payment Overview

ePayment Manager supports check payments through ACH or CheckFree payment gateways. This section describes check enrollment, and then check payment.

Adding a Check Account

The following actions describe the process to enroll a new user with ePayment Manager who specifies a check account at enrollment time:

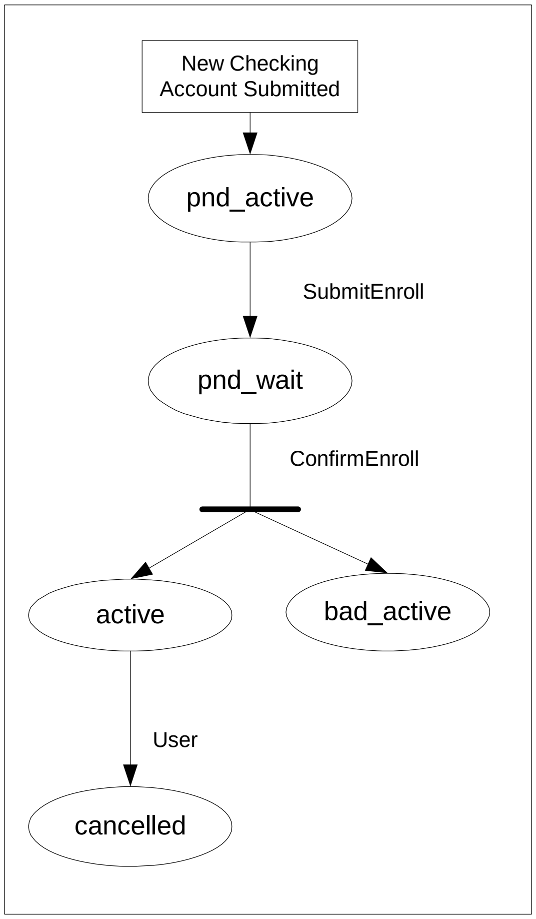
1. A new customer enrolls for check payment services by completing an enrollment form in the user interface. ePayment Manager saves the information in the *payment_accounts* table with an enrollment status of "pnd_active".
2. The pmtSubmitEnroll job runs to submit the enrollment information to the payment gateway. It changes the enrollment status to "pnd_wait". If the check cannot be submitted, its status is changed to "failed".

   For ACH only, pmtSubmitEnroll sends customer enrollment information, which is contained in a zero amount check called a prenote, to an ACH payment gateway for verification. To send a prenote, the pmtSubmitEnroll job creates a zero amount check with status of "prenote_scheduled", and immediately inserts the check into the *check_payments* table with a status of "prenote_processed". This means that the status "prenote_scheduled" is transitory, and so is not visible in the *check_payments* table. A summary report is created, which can be viewed from the Command Center.
3. After receiving the customer enrollment information, the ACH payment gateway responds with a *return* file only if there are errors in the customer enrollment information. If there are no errors, ACH does not send a *return* file, or any other form of acknowledgement.
4. The pmtConfirmEnroll job runs. This job updates the status of the customer enrollment status to "active "if there are no problems after a specified number of days (by default, three days).

   If the payment enrollment information is not correct, the pmtConfirmEnroll job updates the customer enrollment status to "bad_active". An exception report is created, which can be viewed from the Command Center.
5. The customer may optionally receive an email about enrollment status from the pmtNotifyEnroll job.

Check Account Enrollment Status Flow

The following diagram shows the status changes that a **new check account** goes through for enrollment, depending on customer actions and the pmtSubmitEnroll and pmtConfirmEnroll jobs. The status is kept in the *account_status* field in the *payment_accounts* table.

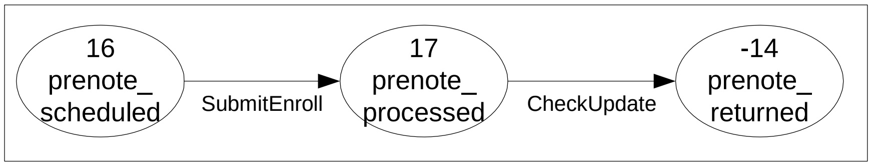


The following table describes each status:

Enrollment Status	Description
pnd_active	A new check account is enrolled, pending approval.
pnd_wait	The check account has been sent to the bank for verification
active	The check account has been activated for payment.
bad_active	The check account failed to be activated.

ACH Prenote Status Flow

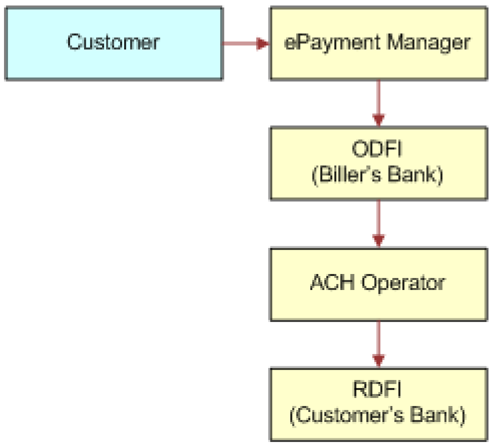
The following diagram shows the status an ACH prenote goes through, due to the pmtSubmitEnroll and pmtCheckUpdate jobs. The table following the diagram describes each state as it changes in the *check_payments* table.



Transaction Status	**Description**
prenote_scheduled (16)	ePayment Manager scheduled a prenote, and it is ready to send to ACH. You cannot see this state in the *check_payments* table; it is used internally by ePayment Manager only.
prenote_processed (17)	ePayment Manager processed a prenote and sent it to ACH.
prenote_returned (-14)	ACH returned a prenote.

Check Payment Transactions

The following diagram shows the entities in an ACH payment transaction:



The following steps describe a typical ACH check payment transaction cycle (excluding transfers between the ODFI, ACH operator and RDFI):

1. A customer logs in and schedules a new payment from the list of defined checking accounts. ePayment Manager inserts a check into the database with a status of "scheduled".

If the customer later cancels the payment, the check status is changed to "cancelled", but the payment remains in the database for the customer to view as a cancelled payment.

2 The pmtCheckSubmit job runs, selects all the checks that are due for payment, creates a batch file of selected checks, and sends the batch file to the payment gateway (ODFI). It also changes the status of each selected check to "processed" in the eStatement/ePayment database.

   If the check cannot be submitted, the status is changed to "failed". A summary report log is generated, which can be viewed from Command Center.

3 The payment gateway (ODFI) processes the received check payment through the ACH operator to the RDFI. If there is an error clearing the check, ACH creates a file containing a code that indicates why the check was returned, and sends the file to ePayment Manager.

4 The pmtCheckUpdate job runs. If there is no return code, and five business days (default) have passed, pmtCheckUpdate changes the status of the check from "processed "to "paid".

   If the payment gateway returns the check, the pmtCheckUpdate job updates the check’s status to "returned", and saves the reason code in the *txn_err_msg* field of the *check_payments* table. An exception report is generated to summarize the information in the returned file, which can be viewed from Command Center.

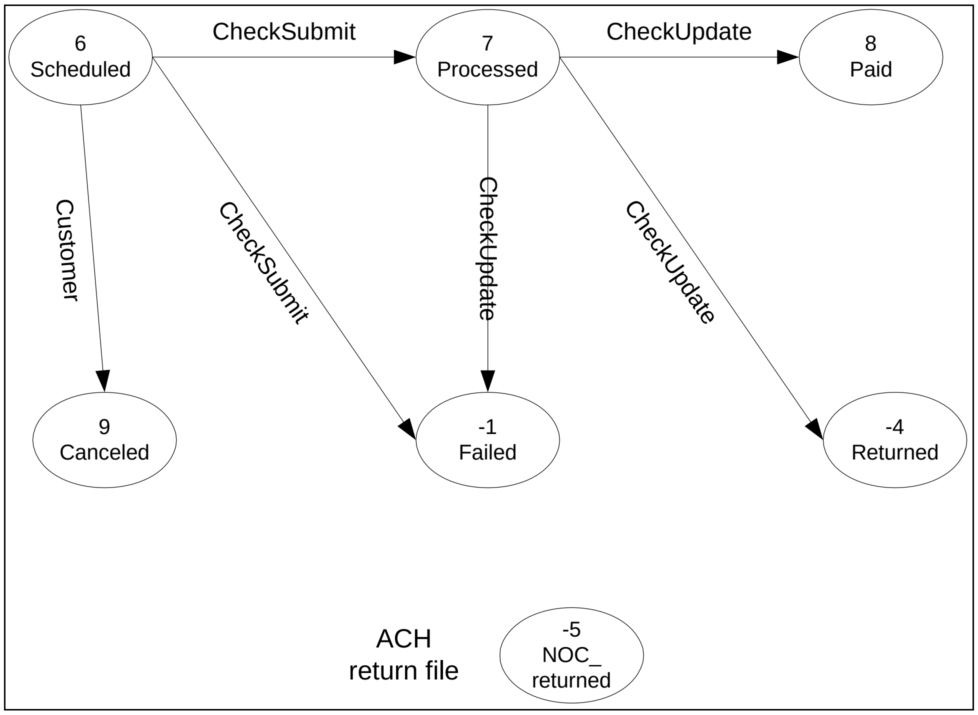
   If there is an error other than "returned", pmtCheckUpdate changes the check status to "failed".

   An ACH payment gateway may return an NOC in response to a prenote. For an NOC, the pmtCheckUpdate job reads the NOC, and inserts a new zero amount check into the *check_payments* table with a status of "noc_returned". The *payment_accounts* table may be updated, depending on the setting of auto-update for NOC in Payment Settings.

5 If configured, the pmtPaymentReminder job sends email to the customer about the status of the check payment.

Check Payment Status Flow

The following diagram shows the states that a check can be in, and the jobs that change the state:



The "NOC_returned" state (-5) is not a check state; it is inserted into the *check_payments* table by the pmtCheckUpdate job when an ACH return file contains a NOC record.

The following table lists the statuses that can occur during a check payment transaction cycle. The values in parentheses () are the actual values saved in the ePayment Manager database.

Transaction Status	**Description**
Scheduled(6)	A customer scheduled a new check payment.
Processed(7)	ePayment Manager processed a check and sent it to the ACH or CheckFree payment gateway.
Paid(8)	ACH paid or cleared a check.
Cancelled(9)	The customer cancelled a check.
Failed(-1)	ACH failed to pay a check failed for a reason other than "returned".
Returned(-4)	ACH returned a check.
noc_returned(-5)	This customer’s payment account information needs to be changed.

Credit

Credit reversals are supported.

User Options

The user interface you create for ePayment Manager can offer a variety of check payment options. Some of those options require you to configure fields in Payment Settings for a check payment gateway.

A user can schedule a check payment using any of the unlimited number of accounts per user that are available for check payment gateways (ACH or CheckFree).

A payment must be scheduled at least 24 hours before the due date. The JSP for check payments enforces the allowable schedule time. The pmtCheckSubmit job submits checks to be paid that are scheduled for payment by the next day, but this setting can be changed by updating the Number of days before a check's pay date for it to be submitted field in the pmtCheckSubmit job. The JSP that verifies check payment dates must be updated when this field is updated.

Payments can be scheduled for a single future date, or payments can be scheduled to recur at a user-defined interval.

Payment invoices allow a customer to select from a list of invoices, and make a payment against the invoices in the list. Invoices that are paid by check can be viewed from the future payments or payment history pages.

CheckFree Direct Pay (CDP)

CheckFree Files and Directories

CheckFree File Names

Files for CheckFree must conform to file naming specifications. CheckFree will not process files that do not adhere to these rules, and the pmtCheckUpdate process will not process incorrectly named files.

The filename format is:

(system).(ftp-username).(type).dat.(datestamp)

Where:

- **system** is either test or prod depending on whether these are test files or actual production files with actual financial transactions.
- **ftp-username** is the CheckFree-provided FTP user name for the deployment
- **type** is one of: "debit", "confirm", "transumm", "tranjrnl", "setlment", "returns". ePayment Manager creates the debit file; the other files are created by CheckFree and returned to ePayment Manager.
- **dat** is always dat.
- **datestamp** in *CCCCMMDDHHMMSS* format.

CheckFree Directory Structure and Handling of Outbound and Inbound Files

The configuration screen specifies the file output and input paths. However, CheckFree files require additional handling beyond ePayment Manager's processing. Files in the input and output directories must be transferred to and from the payment gateway. For CheckFree, the files must first be encrypted using PGP, and then transferred to CheckFree over the Internet using FTP. After CheckFree processes the files, the results are retrieved from CheckFree via FTP, and PGP-decrypted for use by ePayment Manager.

The following directory structure shows a sample setup:

/payments/out/bd - Contains output files produced by ePayment Manager.

/payments/out/ftp - Contains copies of the output files produced by ePayment Manager.

/payments/out/history - A work directory that contains PGP-encrypted versions of the output files produced by ePayment Manager, ready to be transferred to CheckFree.

/payments/in/ftp - Contains PGP-encrypted info received from CheckFree.

/payments/in/history_pgp - Contains copies of the PGP-encrypted files received from CheckFree.

/payments/in/bd - Contains decrypted files ready for processing by ePayment Manager.

The following steps are required to process incoming files:

1. FTP files from CheckFree to */payments/in/ftp*, using the FTP user name provided by CheckFree.
2. Decrypt the files in */payments/in/ftp*. There are now both PGP-encrypted and non-encrypted files in that directory.
3. Move the PGP files to */payments/in/history_pgp.*
4. Move the rest of the files to */payments/in/bd* for processing by pmtCheckUpdate.

The steps required to process outgoing files are:

1. pmtCheckSubmit creates output files in */payments/out/bd.*
2. Copy the contents of */payments/out/bd* to */payments/out/history.*
3. Encrypt the files in */payments/out/bd* (overwriting the originals).
4. Move the contents of */payments/out/bd* to */payments/out/ftp.*
5. FTP the contents of */payments/out/ftp* to CheckFree using the CheckFree-supplied FTP user name.
6. Delete the contents of the */payments/out/ftp* directory.

CheckFree Transactions

Scheduling Payments

CheckFree requires payment transactions to be received by noon EST, and only on weekdays. When using CheckFree as a payment gateway, be sure to allow enough time to run the pmtCheckSubmit job and PGP encrypt the resulting transaction files in time for CheckFree's deadline.

Payment Files

CheckFree files contain the following information:

- **Confirmation** (*confirm*) - An acknowledgement of the debit file received, plus some high-level summary information on the file (total transactions, total value of the transactions).
- **Transaction Summary** (*transumm*) - A more detailed summary of the debit file. Payment does not process this file.
- **Transaction Journal** (*tranjrnl*) - Details for each record received in the debit file, and includes a status for each transaction.
- **Settlements** (*setlment*) - Details on expected deposit amounts, listed by payment date. This file is not processed by Payment.
- **Returns** - Information on any transactions that were returned by the bank, usually due to insufficient funds.

CheckFree returns any payment with the following status:

- **Error Out** - The payment had invalid bank account or routing numbers, invalid payment dates prior to the current date, or the CheckFree-specific information provided in the ePayment Manager configuration is incorrect. These types of transactions are marked as errors in the transaction journal file returned by CheckFree.
- **Accepted** - The transaction data has no errors, and the transaction can be sent to the proper bank for further action.
- **Returned** - The target bank has rejected the transaction, usually due to insufficient funds. This transaction will be in the returns file, and may be returned as long as five days after initially being submitted.

CheckFree does not send any notification that a payment has been made successfully. Their convention is that if a transaction has not been returned within five business days, then it is assumed to have completed successfully.

ACH

Supported SEC Codes

For ACH We Support the following SEC Codes (Standard Entry Class Codes):

- **Web** - Internet initiated entry (default for ePayment Manager).

  Debit entries are originated (either single or recurring) from a customer's account using Web-based authorization.

- **PPD** - Pre Arranged Payment and Deposit Entry. Under PPD the following types are included:
  - Direct Deposit: The credit application transfers funds into the customer's account.
  - Preauthorized Bill Payment: This is a debit application, where billers transfer electronic bill payment entries through the ACH network.

- **CTX** - Corporate Trade Exchange

  Supports multiple addenda record based on ANSI ASC X12 standards. Can be used either with the credit or debit application.

ACH Change Codes (NOC)

The following table lists some of the ACH change codes (also known as NOC codes) that may appear in the returns file after running the pmtCheckUpdate job if previously valid payment information is now incorrect or out-of-date.

Code	ACH Change Code Description
C01	Incorrect DFI Account Number
C02	Incorrect Routing Number
C03	Incorrect Routing Number and Incorrect DFI Account Number
C05	Incorrect Transaction Code
C06	Incorrect DFI Account Number and Incorrect Transaction Code
C07	Incorrect Routing Number, Incorrect DFI Account Number, and Incorrect Transaction Code

Additional information about these and additional ACH change codes are available from www.nacha.org.

ACH Return Codes

The following table lists some of the ACH return codes that may appear in the returns file after running the pmtCheckUpdate job.

Code	ACH Return Code Description
R01	Insufficient Funds
R02	Account Closed
R03	No Account/Unable to Locate Account

Code	**ACH Return Code Description**
R04	Invalid Account Number
R05	Reserved
R06	Returned per ODFI's Request
R07	Authorization Revoked by Customer (adjustment entries)
R08	Payment Stopped or Stop Payment on Item
R09	Uncollected Funds
R10	Customer Advises Not Authorized; Item Is Ineligible, Notice Not Provided, Signatures Not Genuine, or Item Altered (adjustment entries)
R11	Check Truncation Entry Return (Specify) or State Law Affecting Acceptance of PPD Debit Entry Constituting Notice of Presentment or PPD Accounts Receivable Truncated Check Debit Entry
R12	Branch Sold to Another DFI
R14	Representative Payee Deceased or Unable to Continue in that Capacity
R15	Beneficiary or Account Holder (Other Than a Representative Payee) Deceased
R16	Account Frozen
R17	File Record Edit Criteria (Specify)
R20	Non-Transaction Account
R21	Invalid Company Identification
R22	Invalid Individual ID Number
R23	Credit Entry Refused by Receiver
R24	Duplicate Entry
R29	Corporate Customer Advises Not Authorized
R31	Permissible Return Entry (CCD and CTX only)
R33	Return of XCK Entry

Additional information about these and additional ACH return codes are available from http://www.nacha.org/.

NOC Transactions

When a prenote is returned with a NOC, *TXN_MESSAGE* is populated with NOC information formatted as NOC_CODE::NEW_ADDENDA_INFO::OLD_ADDENDA_INFO.

NOC_CODE is the three-character code returned.
NEW_ADDENDA_INFO is the NOC information returned from ACH, which can include the corrected account number, routing and account type. **OLD_ADDENDA_INFO** is the existing addenda information.

ACH Effective Date

The Skip non-business days for batch effective entry date field on the Payment Settings page for an ACH check payment gateway controls how the effective entry date is calculated when the ACH batch file is created by pmtCheckSubmit.

If the field is set to **Yes**, then non-business days are not taken into consideration. The effective entry date is set to the payment date that the customer specified when scheduling the payment.

If the field is set to **No**, then non-business days are skipped, and the effective entry date is the next business day following the **computed** date. ePayment Manager checks the scheduled payment date to see if it is on or before the end of today. If it is, the computed date is the customer-scheduled date plus one. If it is not, then the computed date is the customer-scheduled date.

Non-business days are weekend days, plus the U.S. Federal holidays. The Federal holidays are listed in "*ACH Federal Holidays*" on page 65.

ACH Settlement Date

The ACH settlement date is not written to the ACH batch file by pmtCheckSubmit. That date is added by the ACH Operator when the payment is actually settled.

ACH Addenda Records

ePayment Manager supports ACH addenda records, which means you can append a list of addenda records after an entry detail record in an ACH file. Addenda records are biller-specific, so customization is required to support this feature. Theoretically, you can put any information into an addenda record, for example, the invoices of a payment. To add addenda records, you must write a plug-in for the pmtCheckSubmit job. Contact Oracle Professional Services or your development team for more information about supporting ACH addenda records.

Multiple DDNs

If you put multiple Document Definition Names (DDN's) in one ACH file, all the DDNs must use the same template. If you need to use different templates, contact Oracle Professional Services about creating a custom class.

ePayment Portal Feature

ePayment Manager can act as a payment portal to host the payments from different billers. For ACH, ePayment can put checks from different billers (DDNs) into one ACH file, and can process a return file with checks from different DDNs. See "pmtCheckSubmit Job" on page 88, and "pmtCheckUpdate Job" on page 75 for details.

5 Credit Card Payments

Credit card payments are supported for immediate or future (scheduled) payments and require two steps:

1. **Authorization** - Verifies the customer account and puts a hold on the account for the amount of the payment.
2. **Settlement** - Occurs when the payment is actually made.

ePayment Manager performs authorization and settlement in one transaction using the credit card gateway for credit card payments.

Credit card payments require an agreement with a credit card gateway to process credit card transactions. A cartridge for VeriSign is provided with ePayment Manager, which requires signing up with VeriSign Payment Services. An Oracle Professional Services representative can walk you through that process. In addition, other cartridges can be created by Oracle Professional Services to support other payment processors.

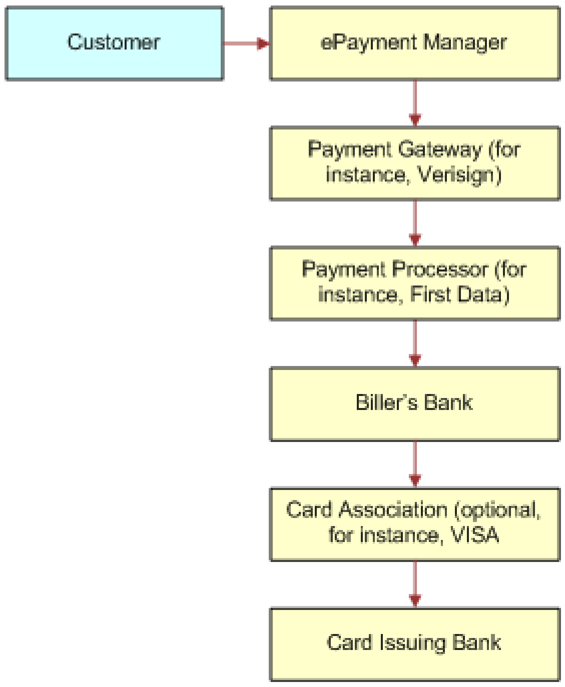
Credit Card Payment Status

The following table lists the statuses that can occur during a credit card payment transaction cycle. The values in parentheses () are the actual values saved in the ePayment Manager database.

Transaction Status	Description
Scheduled (6)	A customer has scheduled a new credit card payment.
Settled (8)	The credit card payment was authorized and settled successfully.
Failed-authorized (-4)	A credit card payment failed during authorization.
Cancelled (9)	A credit card payment was cancelled by the customer.
Failed (-1)	A credit card payment failed because of network problems. This state occurs only for instant payments. For scheduled payments or recurring payments, the state stays "scheduled" if there is a network problem, so is tried again. There is no need for ePayment Manager to retry an instant payment; the user sees the error message and optionally retries making the payment.

Credit Card Payment Transactions

The following diagram shows the entities involved in a credit card payment transaction:


Customer
ePayment Manager
Payment Gateway (for instance, Verisign)
Payment Processor (for instance, First Data)
Biller's Bank
Card Association (optional, for instance, VISA
Card Issuing Bank

Credit card processing usually goes through the following steps:

1. A user enters credit card number and other card related information.
2. The card information is sent to the card-issuing bank for authorization. Authorization only guarantees that the money is available at the time of authorization.
3. The merchant issues a settlement request to issuing bank so that the money can be transferred. The merchant usually does this after fulfillment (sending out ordered goods). For bill payments, the biller does not send out ordered goods, so authorization and synchronization are combined into one operation; a credit card payment is settled at the same time it is authorized.
4. Since credit card is processing is real-time, real-time and not batch-based, the life cycle of credit card is simpler than check processing.

Instant Credit Card Payments

The following diagram shows the states for an instant credit card payment. For instant payments, there is no scheduled state:

```
Credit Card Payment → settled
                     |-> failed-authorize
                     |-> Failed
```

1. A user submits an instant credit card payment from the UI.
2. ePayment Manager sends the payment to credit card cartridge in real time.
3. If the card is authorized and settled, the credit card state is set to "settled".

   If the card failed to authorize, the state is set to "failed_authorize".

If there is a network problem, the state is set to "failed".

4. The card is inserted into *creditcard_payments* table.
5. The result of the transaction is presented to the user.
6. The pmtPaymentReminder job runs and (optionally) sends emails to users who have made an instant payment.

Scheduled Credit Card Payments

The following diagram shows the states for a scheduled credit card payment:

```
Credit Card Payment → Scheduled → Cancelled
                              |
                              | pmtCreditCardSubmit job
                              |→ settled
                              |→ failed-authorize
                              |→ scheduled
```

1. A credit card payment is scheduled by the customer through the user interface, and the payment is marked as "scheduled" in the *creditcard_payments* table.
2. Before the scheduled credit card payment is processed by pmtCreditCardSubmit, the user can modify or cancel it.
3. When the pmtCreditCardSubmit job runs, it selects all credit card payments that are scheduled to be paid at the time the job runs, opens a connection to the credit card payment gateway, and starts making payments. The Number of days before a credit card's pay date for it to be submitted field on the pmtCreditCardSubmit job determines how many days ahead to look when selecting payments to be made.

   If the *IVerisignCreditCardSubmitPlugIn* has been implemented in Payment Settings, this job modifies the credit card payments that are scheduled to be paid, or takes other actions related to the selected credit card payments. Functions in the plugin are called before and after credit card payment processing. For more information about the pmtCreditCardSubmit job and its *IVerisignCreditCardSubmitPlugin*, see page 91. For information about configuring job plugins, contact Professional Services.
4. The credit card gateway sends the transactions to the credit card processor. The credit card processor either authorizes and settles the credit card payment, or rejects it. The results are returned to the credit card gateway, which forwards the results to the pmtCreditCardSubmit job.
5. The pmtCreditCardSubmit job changes the status of the credit card payment in the database depending on the transaction status returned by the credit card processor, and optionally sends email to the customer about the status of the payment.

   If the card is authorized and settled, the credit card state is set to "settled".

   If the card fails to authorize, the state is set to "failed_authorize".

   If there is a network problem, the state remains "scheduled", so it will be processed the next time pmtCreditCardSubmit runs.

6 The pmtPaymentReminder job runs and (optionally) sends emails to users about the status of their scheduled payment.

Reversals

Credit reversals are supported.

User Options

The user interface to ePayment Manager can offer a variety of credit card payment options. Some of those options require you to configure fields in Payment Settings for a credit card payment gateway. See "Credit Card Payment Gateway Configuration (Regular Payee DDN Only)" on page 68 for configuration details.

Using VeriSign as a Payment Gateway

A cartridge for VeriSign is provided with ePayment Manager. Before configuring a VeriSign credit card payment gateway, you must obtain a digital certificate through VeriSign. When choosing VeriSign as the credit card gateway type, the Payment Settings page displays several VeriSign specific fields, which require information from your registration with VeriSign, and the path to the digital certificate. A test certificate through VeriSign is included with ePayment Manager.

You must also configure your application server to support a VeriSign payment gateway. See the *Installation Guide for Oracle Siebel ePayment Manager* for details.

AVS (Address Verification Service)

Address Verification Service (AVS) reduces the risk of fraudulent transactions by verifying that the credit card holder's billing address matches the one on file at the card issuer. The address is optional and does not affect whether the payment is accepted or rejected. However, using an address may get a lower rate from card issuer.

A merchant (also known as the biller) submits the AVS request through the payment process directly to the specific credit card association (for example, VeriSign) for address comparison. If AVS is turned on by the System Administrator, address information is passed into VeriSign as part of the VeriSign request. VeriSign then contacts the credit card issuing bank and passes along the address information. The credit card issuing bank verifies the credit card address information on record matches the address information passed in by VeriSign. The credit card issuing bank then replies back to VeriSign whether information matched (address and zip code are checked during AVS). "Y" means yes, "N" means no, and "X" means a match can not be determined. VeriSign then accepts or rejects (voids) the transaction based on the filter set through ePayment Manager (for both street address and zip code). There is also a filter option to set the international AVS code to determine if the AVS response was international, US or could not be determined. Some credit card issuing banks require city and state verification as well. ePayment Manager does not handle these by default, but the pmtCreditCardSubmit job has a plugin to allow custom code pass in the AVS values. For more

information about AVS functionality, see
http://www.verisign.com/support/payflow/genResources/avs.html.

If ePayment Manager does not send the address information to VeriSign, or the system administrator did not turn on AVS, and the AVS check level is set to Full, the transaction fails. If the card issuer address is sent to the payment gateway, but the address doesn't match the information on the gateway, then the gateway can send an AVS code. If an AVS code is received, ePayment Manager logs the AVS code in the audit tables.

ePayment Manager supports Payflow Pro, but Payflow Pro does not support turning AVS on or off per transaction. However, the lower capability Payflow Link can. You also must set up the AVS level with VeriSign as part of your Payflow Pro agreement. When setting up the account with VeriSign, the merchant must specify the level of AVS check: full, medium or light (see the VeriSign documentation for further information). When ePayment Manager passes the address information, VeriSign accepts or rejects the transaction based on the AVS check level. Note that the AVS check level is specified once during merchant account setup and applies to all transactions for that merchant. The customer (merchant) also needs to specify to VeriSign during setup and that they will be using Payflow Pro (through ePayment Manager) for transactions.

6 Recurring Payments

Overview

ePayment Manager provides two types of recurring payments for check and credit card:

- A **recurring payment** allows a customer to schedule a payment amount that is fixed, for the entire amount due from a bill, or for the minimum amount due from a bill. The payment can be scheduled to be paid on a certain date of the week, month or quarter.
- An **automatic payment** allows a customer to schedule a payment of a fixed amount, for the entire amount due from a bill, or for the minimum amount due from a bill, to be made a certain number of days before due date. Automatic payments of the entire amount due can also be made, if the amount due is less than a specified amount.

Both recurring and automatic payments are designated as recurring payments by the NACHA 2001 specification. NACHA 2001 defines a payment as recurring when the account manager (ePayment Manager) keeps the account information (in a database).

Recurring payments can be modified or cancelled at any time before the payment is scheduled.

Recurring payment allows a customer to make payments automatically, based on the amount and pay date. There are several kinds of recurring payments:

- (Minimum) amount due and before due date. For example, pay the entire amount due two days before the due date.
- (Minimum) amount due and fixed pay date. For example, pay minimal amount due on day 31 of each month.
- Fixed amount and before the due date. For example, pay $100 one day before the due date.
- Fixed amount and fixed pay date. For example, pay $100 on the first day of each month.
- (Minimum) amount due up to a fixed amount and send email if over that fixed amount.

Amount defines how much the recurring payment is going to pay for each payment. The amount can be fixed, amount due or minimum amount due. If the amount is (minimum) amount due, then it must be indexed by eStatement Manager. You must specify the name and format of the (minimum) amount due must be specified on the Payment Settings page in the Command Center.

Pay date defines when each payment is going to be cleared (money will be transferred). Pay date can be fixed or before due. If it is before due, then the due date must be indexed by eStatement Manager. You must specify the name and format of the due date on the Payment Settings page in the Command Center.

For monthly payments, if day 29, 30, or 31 is selected, and that day does not exist for a particular month, the pay date defaults to the last day of that month. For example, specifying day 31 of each month ensures that payments are made on the last day of each month.

For weekly payments, the week starts on Sunday. For example, day 1 of each week means Sunday.

The **effective period** defines when a recurring payment starts and ends. A payment is made if its pay date is within the effective period (inclusive). If the pay date is after the end date of the effective period, the system deactivates the recurring payment. By default, a recurring payment only starts tomorrow. This is done so that all bills that arrive up to and including today are considered paid, so recurring payment should not pay these bills a second time.

After an end-customer creates a recurring payment, that customer is not permitted to change the payment amount from fixed to (minimum) amount due, or to change the pay date from *fixed to before due date*, or vice versa. When a recurring payment starts (which is when the first recurring payment has been made), the start date of the recurring payment cannot be modified.

CAUTION: Before deleting an Indexer DDN, make sure there are no recurring payments associated with the DDN.

Multiple Recurring Payment Feature

The multiple recurrent payment feature lets a customer (payee) select multiple biller accounts to pay with a single recurring payment setup.

It is also possible to set up one recurring payment for a collection of bills from different accounts that come from different indexer DDNs.

Payment entities like recurring payments, bank accounts, and payments made are visible to and can be modified by all authorized persons.

To use the multiple recurring payment feature in the paymentComplex application, for example, you would select multiple billing accounts when configuring the recurring payment.

Recurring Payment Transaction Cycle

Recurring payment information is saved into the *recurring_payments* table.

pmtRecurringPayment retrieves bills from eStatement, makes payments (check or credit card) and sends email notifications for recurring payments. The job performs two actions:

1. pmtRecurringPayment contacts eStatement to get the latest bill for a recurring payment that a customer set up through the UI. This process is called **synchronization**. A recurring payment can only be synchronized with eStatement if it's associated with a bill and the amount to pay is the minimum (amount) due or the pay date is before the due date. A recurring payment with fixed amount and fixed date won't be synchronized with eStatement, which means there is no bill information associated with this recurring payment.

2. pmtRecurringPayment schedules payments (inserts a payment with status of "scheduled "in the *check_payments* or *creditcard_payments* table so that the payments are processed. This process is called **scheduling**. A payment is scheduled three days before the pay date (by default). The number of days can be changed by changing the Number of days before pay date to schedule the payment field in the job configuration. This delay allows the customer to modify or cancel this payment before the payment is processed by the pmtCheckSubmit or pmtCreditCardSubmit jobs.

The following table shows the columns that are updated in the *recurring_payments* table by the pmtRecurringPayment job:

recurring_payments Column Name	**Description**
bill_scheduled	Y/N: determines whether the current bill associated with the recurring payment has been scheduled (inserted) into *check_payments* or *creditcard_payments*. It's always "N" for a fixed amount and fixed pay date.
Status	Active/Inactive: This status is calculated internally. It indicates whether the recurring payment has ended, because either the pay date is after the end date, or the number of payments has reached the maximum allowed.
last_process_time	The last synchronization time. To improve performance, only bills whose doc date falls between *last_process_time* and the current job running time (inclusive) are synchronized. By default, *last_process_time* is set to the *start_date* of the effective period when the recurring payment is created, which means all bills whose doc dates are before *start_date* won't be synchronized.
last_pay_date	The pay date of last payment made. It is set to 01/01/1970 if the recurring payment has not started yet.
next_pay_date	The pay date of next payment. It is calculated based on *start_date*, *last_pay_date* and *pay_interval*.
bill_id	A foreign key reference to a row in the *payment_bill_summaries* table. Use *bill_id* to retrieve the latest bill information paid by the recurring payment. It may be null if there is no such bill.
curr_num_payments	Current number of payments made.

TIP: There is no payment inserted into *check_payments* or *creditcard_payments* table when a recurring payment is created by the user. Payments are inserted by the pmtRecurringPayment job. See "pmtRecurringPayment Job" on page 82 for more information.

Tables Affected by Recurring Payments

The *recurring_payments* table only contains the setup information for the recurring payment, which is the data entered from Web interface by end users. It is not used to save bill summary or actual payment information. The *amount* field in the *recurring_payments* table records the amount when you:

- Specify the recurring payment to pay fixed amount, or
- Pay if less than this amount, or
- Pay up to this amount

Bill summary information is pulled from the eStatement tables and saved into the *payment_bill_summaries* table. After the pmtRecurringPayment job runs, the *payment_bill_summaries* table is populated, and the *bill_id* of the *recurring_payments* table is also populated.

Actual payment information is scheduled into the *check_payments* (for check) or *creditcard_payments* (for credit card) tables.

Recurring Payment Examples

Four examples of recurring payment are described here, with additional details about the relevant database interactions.

Case 1: Amount Due and Before Due Date

1 On date 04/09/2001, a customer with account number acct1111 creates a recurring payment. The amount is amount due, the pay date is one day before due date, the start date is 04/10/2001, and the end date is 06/10/2001.

recurring_payments Column Name	Value
payer_account_number	acct1111
bill_scheduled	Y
status	active
last_process_time	04/10/2001, same as start date
last_pay_date	01/01/1970, not paid yet
next_pay_date	01/01/3000; this future date ensures there is no due date available yet
bill_id	null
max_num_payments	2147483647; This large number means the recurring payment is only deactivated when the pay date is after the end date

2 The eStatement Indexer job runs and indexes one bill (the doc ID is bill1, in this example) on 03/10/2001. On 04/10/2001, the indexer job runs again and indexes two more bills: bill2 and bill3.

Z_PRIMARY	Z_DOC_ID	Z_DOC_DATE	AmountDue	DueDate
acct1111	bill1	03/10/2001	100.01	04/15/2001
acct1111	bill2	04/10/2001	50.00	04/25/2001
acct1111	bill3	04/10/2001	100.00	05/15/2001

3 The pmtRecurringPayment job runs on 04/10/2001 23:59:00PM, after the Indexer job. The job searches the recurring_payments table to find all recurring payments whose bill_scheduled is "Y" and status is "active". It finds the example recurring payment and then asks eStatement to

return all bills whose account number is acct1111 and whose Z_DOC_DATE is between 04/10/2001 (last_process_time) and 04/10/2001 23:59:00PM (job run time). Two bills, bill2 and bill3 will be returned. pmtRecurringPayment then finds the bill with latest due date bill3. bill2 is ignored because only the latest bill is paid.

4 After finding the latest bill from eStatement, pmtRecurringPayment checks whether the due date of this bill is after the due date of the bill used in the last payment (last bill info can be retrieved from payment_bill_summaries using the bill_id). If not, that means this is an old bill and should not be paid. In this case, since there is no last payment, the bill bill3 is paid.

5 bill3 is inserted into the payment_bill_summaries table and the recurring_payment table is recalculated as follows:

Column Name	**Value**
payer_account_number	acct1111
bill_scheduled	N, means this bill has not been paid or scheduled
status	active, because *next_pay_date* is within the effective period
last_process_time	04/10/2001 23:59:00PM, changes to job run time
last_pay_date	01/01/1970, unchanged
next_pay_date	05/14/2001, one day before the due date, 05/15/2001
bill_id	bill3

6 If pmtRecurringPayment runs between 04/11/2001 and 05/10/2001, nothing happens to this recurring payment because synchronization and scheduling does not happen. The table remains unchanged.

7 On 05/11/2001 11:59:00PM, three days before *next_pay_date*, pmtRecurringPayment runs again. The recurring payment mentioned previously won't be synchronized, because its *bill_scheduled* is "N". However, it will be scheduled. pmtRecurringPayment finds all recurring payments whose *bill_scheduled* is N, *status* is "active "and *next_pay_date* is equal to or before 05/14/2001 (05/11/2001 + 3 days). The previously mentioned recurring payment is picked up and a payment is inserted into the *check_payments* or *creditcard_payments* table. The amount of the payment is $100.00, and the pay date is 05/14/2001. After this, the recurring payment table is changed to:

Column Name	**Value**
payer_account_number	acct1111
bill_scheduled	Y, means this bill has been paid
status	"active" because *next_pay_date* is within the effective period
last_process_time	04/10/2001 23:59:00PM, unchanged since there was no synchronization
last_pay_date	05/14/2001, change to check's pay date
next_pay_date	05/14/2001, unchanged

Column Name	**Value**
bill_id	bill3
payment_id	points to the new *payment_id* inserted into the *check_payments* or *creditcard_payments* table

The customer can now view the payment from Future Payments in the example interface. They can update or cancel the scheduled payment if desired.

8 On 05/12/2001 23:59:00PM, pmtRecurringPayment runs again and finds bills whose doc date is between 04/10/2001 11:59:00PM and 05/12/2001 23:59:00PM. No bills exist, and the last process time is updated to 05/12/2001 23:59:00PM. Everything else remains the same.

9 On 05/13/2001, the indexer job runs again and inserts a new bill, bill4:

Z_PRIMARY	**Z_DOC_ID**	**Z_DOC_DATE**	**AmountDue**	**DueDate**
acct1111	bill1	03/10/2001	100.01	04/15/2001
acct1111	bill2	04/10/2001	50.00	04/25/2001
acct1111	bill3	04/10/2001	100.00	05/15/2001
acct1111	bill4	05/13/2001	80.00	06/15/2001

10 On 05/13/2001 23:59:00PM, the pmtRecurringPayment job runs again. It contacts eStatement and retrieves bills whose doc date are between 05/12/2001 23:59:00PM and 05/13/2001 23:59:00PM. bill4 is retrieved and the *recurring_payments* table is updated like this:

Column Name	**Value**
payer_account_number	acct1111
bill_scheduled	N, means this bill has not been paid
status	"inactive", because *next_pay_date* is beyond the effective period
last_process_time	05/15/2001 23:59:00PM, changes to job run time
last_pay_date	05/14/2001, unchanged
next_pay_date	06/14/2001, one day before due date, 06/15/2001
bill_id	bill4

After synchronization, the recurring payment is deactivated, and it will never be synchronized or scheduled again.

Case 2: Amount Due And Fixed Pay Date

Case 2 is similar to case 1. Refer to case 1 for additional information.

1 On 04/09/2001, a customer with account number acct1111 creates a recurring payment. The amount is amount due, the pay date is day 31 of each month, the start date is 04/10/2001, and the recurring payment stops after 10 payments.

Column Name	Value
payer_account_number	acct1111
bill_scheduled	Y
status	"active"
last_process_time	04/10/2001
last_pay_date	01/01/1970
next_pay_date	4/30/2001; the first available pay date after 04/10/2001 (because there is no April 31).
bill_id	null
end_date	01/01/3000; The end date is so far in the future that the recurring payment will only be deactivated when the number of payments reaches maximum allowed.
curr_num_payments	0; no payments yet.

The index table has the following values:

Z_PRIMARY	Z_DOC_ID	Z_DOC_DATE	AmountDue	DueDate
acct1111	bill1	03/10/2001	100.01	04/15/2001
acct1111	bill2	04/10/2001	50.00	04/25/2001
acct1111	bill3	04/10/2001	100.00	05/15/2001

Even though the pay date is not related to the due date, *DueDate* must still be indexed because it is used to decide which bill is the latest.

2 pmtRecurringPayment runs on 04/10/2001 23:59:00PM, after the indexer job. bill3 is found in the index table and inserted into the *payment_bill_summaries* table. The *recurring_payments* table is recalculated as follows:

Column Name	Value
payer_account_number	acct1111
bill_scheduled	N; this bill has not been paid.
status	"active"; *curr_num_payments* is less than max_num_payments.
last_process_time	04/10/2001 23:59:00PM; changes to job run time.
last_pay_date	01/01/1970; unchanged.

Column Name	Value
next_pay_date	04/30/2001; there is no April 31.
bill_id	bill3
curr_num_payments	0

3 On 04/27/2001, three days before *next_pay_date*, pmtRecurringPayment runs again. There is no synchronization (*bill_scheduled* is "N"), but a payment is inserted into the *check_payments* or *creditcard_payments* table. The amount of the check is $100.00 and its pay date is 04/30/2001. The recurring payment table is changed as follows:

Column Name	Value
payer_account_number	acct1111
bill_scheduled	Y; means this bill has been paid.
status	"active"; *curr_num_payments* is less than *max_num_payments*.
last_process_time	04/10/2001 23:59:00PM: not changed since there has been no synchronization.
last_pay_date	04/30/2001; changed to *next_pay_date*.
next_pay_date	05/31/2001; changed to next available pay date.
bill_id	bill3
payment_id	Points to the new *payment_id* inserted into the *check_payments* or creditcard_payments table.
curr_num_payments	1

4 Repeat steps 2, 3 and 4 until *curr_num_payments* reaches 10. At step 4 of the tenth payment, the status changes to "inactive."

If no bills arrive for a month, then *next_pay_date* is automatically moved to next month. For example, if there is no bill for April, then the *next_pay_date* is automatically moved from 04/30/2001 to 05/31/2001 when the current job run time is May 1.

Case 3: Fixed Amount and Before Due Date

Case 3 is similar to case 1. Refer to case 1 for additional information.

1 On 04/09/2001, a customer with account number as acct1111 creates a recurring payment from the UI. The amount is $50, the pay date is one day before the due date, the start date is 04/10/2001 and the recurring payment stops after 10 payments.

Column Name	Value
payer_account_number	acct1111
bill_scheduled	Y

Column Name	Value
status	"active"
last_process_time	04/10/2001
last_pay_date	01/01/1970
next_pay_date	01/01/300
bill_id	null
end_date	01/01/3000; the end date is so far in the future that the recurring payment will only be deactivated when the number of payments reaches the maximum allowed.
curr_num_payments	0; no payment yet.

Index table entries are as follows:

Z_PRIMARY	Z_DOC_ID	Z_DOC_DATE	DueDate
acct1111	bill1	03/10/2001	04/15/2001
acct1111	bill2	04/10/2001	04/25/2001
acct1111	bill3	04/10/2001	05/15/2001

Amount due is not required for this case.

2 The pmtRecurringPayment job runs on 04/10/2001 23:59:00PM, after the indexer job. In this case, bill3 is found in the index table and inserted into the *payment_bill_summaries* table. The *recurring_payments* table is recalculated as follows:

Column Name	Value
payer_account_number	acct1111
bill_scheduled	N; this bill has not been paid.
status	"active"; *curr_num_payments* is less than *max_num_payments*.
last_process_time	04/10/2001 23:59:00PM; changes to job run time.
last_pay_date	01/01/1970; unchanged.
next_pay_date	05/14/2001 (One day before the due date, 05/15/2001.)
bill_id	bill3
curr_num_payments	0

3 On 05/11/2001, three days before *next_pay_date*, pmtRecurringPayment runs again. There is no synchronization (because *bill_scheduled* is "N"), but a payment is inserted into the *check_payments* or *creditcard_payments* table. The amount of the payment is $50.00 and its pay date is 05/14/2001. The *recurring_payments* table is changed as follows:

Column Name	**Value**
payer_account_number	acct1111
bill_scheduled	Y; means this bill has been paid.
status	"active"; *next_pay_date* is not after *end_date*.
last_process_time	04/10/2001 23:59:00PM; unchanged, since there was no synchronization.
last_pay_date	05/11/2001; changed to *next_pay_date*.
next_pay_date	05/11/2001; unchanged, the next bill is not known.
bill_id	bill3
payment_id	Points to the new `payment_id` inserted into the *check_payments* or *creditcard_payments* table.
curr_num_payments	1

Repeat steps 2, 3 and 4 until *next_pay_date* is after *end_date*, when status changes to inactive.

Case 4: Fixed Amount and Fixed Pay Date

Case 4 is similar to case 1. Refer to case 1 for additional information.

1 On 04/09/2001, a customer with account number "acct1111" creates a recurring payment. The amount is $50 and the pay date is day 1 of each month. The recurring payment starts at 04/10/2001 and ends at 06/10/2001. The columns in the *recurring_payments* table are updated as follows:

Column Name	**Value**
payer_account_number	acct1111
bill_scheduled	N
status	"active"
last_process_time	04/10/2001
last_pay_date	01/01/1970
next_pay_date	05/01/2001
bill_id	null
end_date	06/10/2001
curr_num_payments	0; no payment yet.

2 On 04/28/2001, three days before *next_pay_date*, pmtRecurringPayment runs again. There is no synchronization (*bill_scheduled* is always "N") but a payment is inserted into the

check_payments or *creditcard_payments* table. The amount of the check is $50.00 and its pay date is 05/01/2001. The columns in the *recurring_payments* table are updated as follows:

Column Name	**Value**
payer_account_number	acct1111
bill_scheduled	N; this bill has been paid.
status	"active"; *next_pay_date* is not after *end_date*.
last_process_time	04/10/2001; unchanged, since there was no synchronization.
last_pay_date	05/01/2001; changed to *next_pay_date*.
next_pay_date	06/01/2001; changed to the next available pay date.
bill_id	null
payment_id	Points to the new *payment_id* inserted into the *check_payments* or *creditcard_payments* table.
curr_num_payments	1

Repeat step 2 until *next_pay_date* is after *end_date*. Then the status changes to "inactive".

Scheduling ePayment Manager Jobs

We recommend that you schedule payment jobs to run during periods of low customer activity, for example, at midnight.

If two jobs access the same table, schedule them to run at different times. For example, you should run pmtCheckSubmit, pmtCheckUpdate and pmtPaymentReminder at different times, allowing enough time between jobs so that two jobs won't try to access the same database table at the same time (which could lead to a database access error). pmtSubmitEnroll, pmtConfirmEnroll and pmtNotifyEnroll should also run at different times. The best solution is to use the pmtAllCheckTasks job or the job chain feature from eStatement to create a chain of jobs, both of which ensure that no two jobs run at the same time.

There is no strict ordering of ePayment jobs. For check jobs, we recommend that you run pmtRecurringPayment, pmtCheckSubmit, pmtCheckUpdate and pmtPaymentReminder in that order. For enroll jobs, we recommend that you run pmtSubmitEnroll, pmtConfirmEnroll and pmtNotifyEnroll in that order. There is only one credit card job, pmtCreditCardSubmit, and we suggest that you run it and then the pmtPaymentReminder job.

Many recurring payments depend on the bill data in eStatement being indexed. In order to ensure that recurring jobs run successfully for all users, the Indexer job must run on the same calendar day as the recurring payment job. For example, run the indexer at 12:30 AM and pmtRecurringPayment at 5AM.

ePayment Manager Job Status Monitoring

When a payment job is done, an email can be sent to the administrator about the status of the mail. This feature is enabled in the ePayment Manager Settings.

ePayment Manager Job Plug-In

Some payment jobs support plug-ins to extend core payment functionality. See the *Customizing and Extending Oracle Siebel ePayment Manager* guide for details.

To Configure Recurring Payments

Recurring payments are configured by the pmtRecurringPayment job.

Recurring payments can be configured to provide email notification when a payment is scheduled by Payment, when the effective period has ended, and when a recurring payment is cancelled because the check or credit card account information in the recurring payment does not match the account in the user profile.

See "pmtRecurringPayment Job" on page 82 for details on configuring recurring payments.

Testing Recurring Payment

Testing recurring payments is somewhat more complex than testing other payment functionality because not all steps in the recurring payment cycle can occur on the same calendar date. As a result, unless one is willing to conduct "real-time" testing that takes place over more than one day, the test process involves changing the system date of the machine(s) on which the application server and database reside. This is certainly not recommended in a production environment. Even in a test environment, recurring payment testing should not be conducted while testing other functionality, because the system date changes are likely to interfere with the results of the other tests.

If operating in a distributed environment (with the application server and database server on different machines), moving the system date forward must be done on both machines. It should not matter if the **time** on each machine differs by a few minutes. Once testing is completed and the environment is to be returned to the actual current date, **both** the database and the application server must be shut down (not the machine itself), the date should be reset, and then the database and application server may be restarted. The Oracle *logger* and *scheduler* should also be shut down and restarted in this manner. Billing data and transactions that were indexed or entered while the system date was moved ahead should not be used for additional testing after the environment has been reset to the current date because the associated "future" dates recorded in the database causes incorrect behavior.

The following example test sequences validate the operation of the recurring payment feature. The numbering corresponds to the first four cases listed in the Recurring Payments document. All examples assume a beginning "true" date of April 1. For testing purposes, it may be necessary to manipulate the due date of certain bills. This is done by altering the source data prior to indexing, or by updating the due date field directly in the database after indexing.

Case 1: Pay Amount Due *X* days Before Due Date

1. Some billing data is already indexed, and a user is enrolled with a valid payment account.
2. On April 1, a user sets up a recurring payment, to pay the amount due 1 day before the due date. The amount due and the date due must both be indexed fields. The start date is April 2, 2002 (the earliest possible start date).
3. Move system date ahead to April 2, 2002.
4. Index another bill, with a due date of (for example) April 10, 2002.
5. Run the recurring payment task. Note that on the pmtRecurringPayment job configuration page, the number of days before pay date to schedule the payment defaults to three. This does not affect when a payment is **made**; it affects **how long** before the pay date the customer sees the payment on the future payments page. This gives the customer time to cancel or modify the payment, if so desired, if this functionality has been allowed by your application.
6. The bill in question should now be recorded in the *payment_bill_summaries* table; it has been synchronized, but not yet scheduled. The *recurring_payments* table should show this bill's *bill_id*, *bill_scheduled* should equal "N," and the *next_pay_date* should be 04/09/2002 (one day before the due date).
7. Move the system date ahead to April 9, 2002.
8. Run the pmtRecurringPayment task. The payment should be shown as "scheduled", because it is now one day before the due date. In the *recurring_payments* table, *bill_scheduled* should equal "Y," and both *next_pay_date* and *last_pay_date* should be 04/09/2002. These values do not change again until another bill is synchronized. The payment itself should now appear in the *check_payments* or *creditcard_payments* table, whichever applies.

   Note that the pmtRecurringPayment job had not been run on April 2, but only after the system date was set to April 9, then the bill would be synchronized and the payment would be scheduled at the same time.
9. Run the applicable submission job (CreditCard or Check), and the payment should be submitted as expected.

Case 2: Pay Amount Due on a Fixed Date

1. Some billing data is already indexed, and a user is enrolled with a valid payment account.
2. On April 1, a user sets up a recurring payment, to pay the amount due on the 10th day of the month. Only the amount due must be an indexed field. The start date is April 2, 2002 (the earliest possible start date).
3. Move system date ahead to April 2, 2002.
4. Index another bill.
5. Run the recurring payment task.
6. The bill in question should now be recorded in the *payment_bill_summaries* table; it has been synchronized, but not yet scheduled. The *recurring_payments* record should show this bill's *bill_id*, *bill_scheduled* should equal "N," and the *next_pay_date* should be 04/10/2002 (the customer's desired pay date).

7. Move the system date ahead to April 7, 2002.
8. Run the pmtRecurringPayment job. The payment should be shown as "scheduled", if the “number of days before pay date to schedule the payment” on the pmtRecurringPayment configuration page is set to 3. In the *recurring_payments* table, *bill_scheduled* should equal “Y,” *next_pay_date* should be 05/10/2002, and *last_pay_date* should be 04/10/2002.
9. Move the system date ahead to April 10, 2002.
10. Run the applicable submission job (CreditCard or Check), and the payment should be submitted as expected.

Case 3: Pay Fixed Amount *X* Days before Due Date

1. Some billing data is already indexed, and a user is enrolled with a valid payment account.
2. On April 1, a user sets up a recurring payment, to pay the specific amount of $19.95 one day before the due date. In this case, only date due must be indexed, although it is unlikely a biller would not index the amount due. The start date is April 2, 2002 (the earliest possible start date).
3. Move the system date ahead to April 2, 2002.
4. Index another bill, with a due date of (for example) April 10, 2002.
5. Run the pmtRecurringPayment task.
6. The bill in question should now be recorded in the *payment_bill_summaries* table; it has been synchronized, but not yet scheduled. The *recurring_payments* record should show this bill’s *bill_id*, *bill_scheduled* should equal “N,” and the *next_pay_date* should be 04/09/2002 (one day before the due date).
7. Move the system date ahead to April 9, 2002.
8. Run the pmtRecurringPayment job. The payment should be shown as "scheduled", because it is now one day before the due date. In the *recurring_payments* table, *bill_scheduled* should equal “Y,” and both *next_pay_date* and *last_pay_date* should be 04/09/2002. These values do not change again until another bill is synchronized. The payment of $19.95 should now appear in the *check_payments* or *creditcard_payments* table, whichever applies. Note that if the pmtRecurringPayment task had not been run on April 2, but only after the system date was set to April 9, then the bill would be synchronized and the payment would be scheduled at the same time.
9. Run the applicable submission job (pmtCreditCardSubmit or pmtCheckSubmit), and the payment should be submitted as expected.

Case 4: Pay Fixed Amount on a Fixed Date

1. Some billing data is already indexed, and a user is enrolled with a valid payment account.
2. On April 1, a user sets up a recurring payment, to pay the specific amount of $19.95 on April 10. This payment does not rely on any indexed fields. The start date is April 2, 2002 (the earliest possible start date).
3. Move the system date ahead to April 7, 2002.

4 Run the pmtRecurringPayment task. The payment should be shown as "scheduled", if the number of days before pay date to schedule the payment on the pmtRecurringPayment configuration page is set to "3". In the *recurring_payments* table, *bill_scheduled* should equal “Y,” *next_pay_date* should be 05/10/2002, and *last_pay_date* should be 04/10/2002. No record is inserted into the *payment_bill_summaries* table, since this recurring payment does not depend on any indexed fields.

5 Move the system date ahead to April 10, 2002.

6 Run the applicable submission job (CreditCard or Check), and the payment should be submitted as expected.

Rebill and Recurring Payment

Description

A *rebill* is a bill that is reissued during the current billing cycle. This occurs when a biller makes frequent adjustments to a bill before the due date.

After a user sets up a recurring payment, the pmtRecurringPayment job runs to schedule payments. pmtRecurringPayment gets the latest bill from eStatement (which is called synchronization), and then determines whether to schedule it for payment.

In previous versions of Payment, the bill amount was synchronized when the bill first arrived. If the bill amount was adjusted after synchronization, then the amount of payment scheduled would not be correct, until the payment was actually scheduled, and the bill was synchronized again.

By default, ePayment Manager now synchronizes a rebill each time the pmtRecurringPayment job runs. You can also configure ePayment Manager to synchronize only once, to reduce processing overhead for sites that do not adjust bills during a billing cycle.

Payment Settings

The following additional parameter has been added to the pmtRecurringPayment job: "When to synchronize recurring payment with eStatement".

By default, ePayment Manager uses the latest available bill when submitting the payment to the payment gateway. You can configure each payment gateway to only synchronize once, which reduces processing. The setting "whenever job runs" can be changed to "Only after the current bill is scheduled", which causes ePayment Manager to synchronize only once; when the bill is scheduled.

The following additional parameter has been added to Payment Settings:

For the parameter Implementation of com.edocs.payment.imported.BillDepot:

- com.edocs.payment.imported.eadirect.BillDepot - Default
- com.edocs.payment.imported.eadirect.SampleBillDepot - Adds the following features:
  - If the rebill has a zero amount, the due date is set to the previous bill's due date.
  - If the due date is "ON RECPT", the due date is changed to the current date.

Payment History

Bills that were adjusted are marked as "scheduled". Only the latest bill is marked as "active".

Email

If pmtRecurringPayment is configured to send email when a payment is scheduled, then email is sent for each rebill.

Logic

ePayment Manager recognizes a rebill using doc_id and INV number. It does not check the date of the bill, or the amount. Because of this, a rebill must be indexed after the original bill.

The following list describes the steps ePayment Manager takes to synchronize a bill for the default payment setting:

1. If the setting of when to synchronize recurring payment with eStatement is Whenever job runs**,** the pmtRecurringPayment job synchronizes the bill with eStatement every time pmtRecurringPayment runs. If it is Only after the current bill is scheduled**,** a rebill is not synchronized unless it is time to schedule the payment.
2. The pmtRecurringPayment job runs again. If the setting of Synchronize with eStatement Every Time the Job Runs is Yes, and a rebill arrives, the pmtRecurringPayment job synchronizes the bill with eStatement.

   If the payment for this bill has already been scheduled, the job cancels the scheduled payment, and schedules a payment for the updated amount.

   If the status of the bill is "processed", then the rebill is ignored.

Each time the pmtRecurringPayment job runs, it looks at the bills indexed by eStatement since the last time the pmtRecurringPayment job ran. To determine whether a bill is newer, it checks the due date. If the due date is the same as the previous bill, then the bill is considered newer if the *doc_date* database field or the *IVN* number is newer, and the bill's payment status is "processed". The last process time of that recurring payment is updated.

7 Email Notifications

Email Notifications

Email can be sent to customers to notify them of an action regarding a payment and about enrollment status. The job where customer email is configured and the type of email available is listed in the following table:

Job	**Email Notification Type**
pmtRecurringPayment	Sends email notification when a payment is scheduled by a recurring payment, and when the recurring payment expires.
pmtPaymentReminder	Reminds the customer when a payment is about to be made, when a payment has failed, and/or when a payment has been returned.
pmtNotifyEnroll	Notifies the customer about enrollment status.
pmtCreditCardExpNotify	Notifies customer that their credit card is about to expire.
pmtAllCheckTasks	Email can also be sent to the payment administrator indicating whether payment jobs have finished successfully. Job status email is optional. It can be disabled on a per payment gateway basis during payment gateway configuration.

Oracle Siebel ePayment Manager uses Velocity templates. The templates can be found in the *EDX_HOME*\template directory (*EDX_HOME*/template on Windows). For more information about using Velocity templates, see *Customizing and Extending Oracle Siebel ePayment Manager* guide.

NOTE: The templates in the $PAYMENT_HOME$/lib/payment_resources directory (the %PAYMENT_HOME%\lib\payment_resources directory on Windows) are obsolete; ignore these files.

8 Configuring Payment Gateways

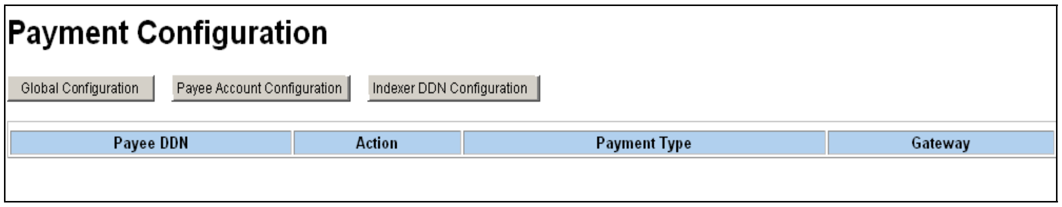
Configuring a Payment Gateway

You must configure at least one payment gateway for ACH, credit card, or CheckFree payments after installing ePayment Manager and creating an application. You can also update a payment gateway at any time to add or remove information about a particular payee.

Configuration information specific to a payment gateway must be provided by the payment gateway or by the biller's bank. You should have this information in-hand before configuring.

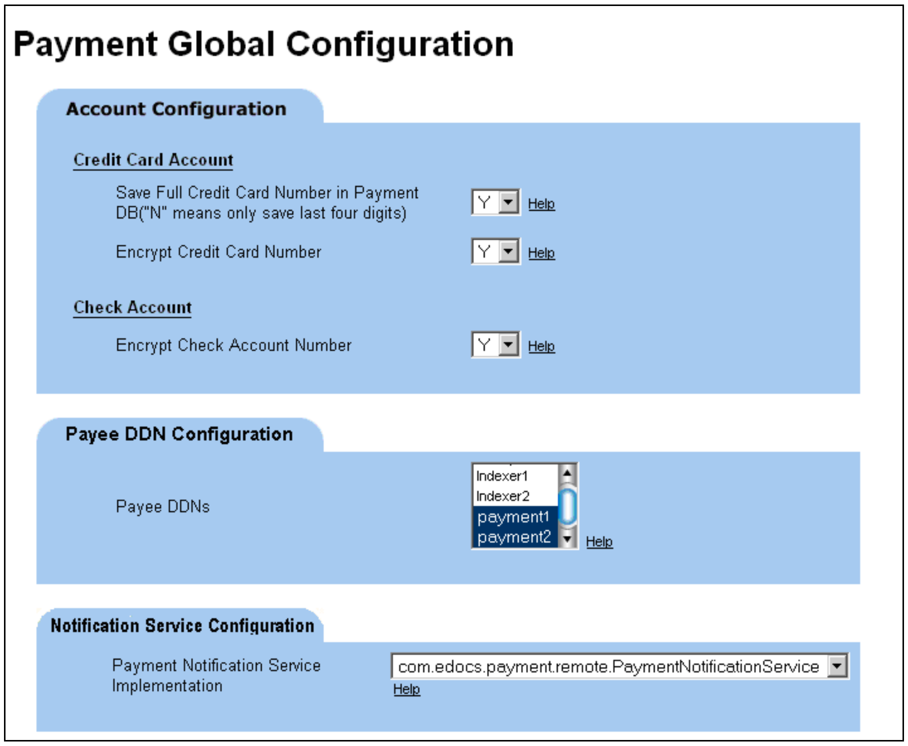
To configure a payment gateway and specify global configuration settings:

1. Access the Command Center by pointing your Web browser to the virtual directory on the Web Server (*http://<Server_Name>:<Port>/eBilling/*). Log on using the **admin** username and the default password, **oracle**.
2. If you have not created payment DDNs already, create two DDNs: one for regular payment and one for external payments (if needed). (See the *Administration Guide for Oracle Siebel eStatement* for details on creating a DDN.) These DDNs are for running payment jobs, as payee DDNs. The DDN used for indexing the bills can be same as payment DDNs or can be entirely different. When creating these DDNs specify the data source as: **edx/paymentComplex/ejb/EdocsDataSource**.
3. From the Command Center menu, click **Settings**. The Settings page appears.
4. Click the **Payment Settings** tab. The Payment Configuration page appears.


Payment Configuration

Payee DDN	Action	Payment Type	Gateway

5. Click **Global Configuration**. The Payment Global Configuration page appears showing the payee DDNs in the Payee DDN Configuration section.



Payment Global Configuration

Account Configuration

Credit Card Account

Save Full Credit Card Number in Payment DB("N" means only save last four digits)	Y	Help
Encrypt Credit Card Number	Y	Help

Check Account

Encrypt Check Account Number	Y	Help

Payee DDN Configuration

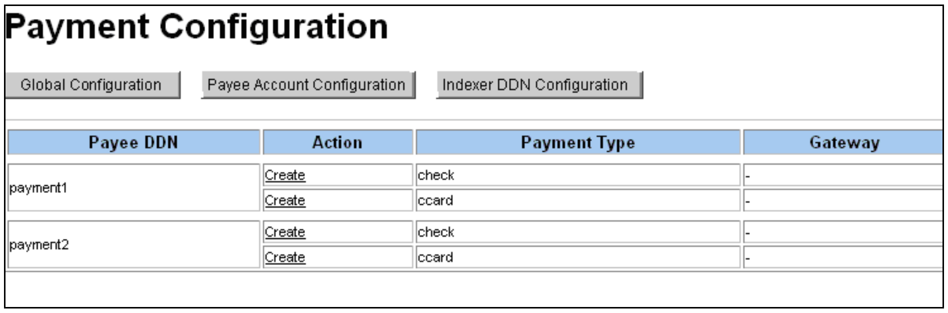
Payee DDNs: Indexer1, Indexer2, payment1 (selected), payment2 (selected) — Help

Notification Service Configuration

Payment Notification Service Implementation: com.edocs.payment.remote.PaymentNotificationService — Help

6 Select both of the payment DDNs as Payee DDNs. (Hold down the Ctrl key to select multiple items).

7 Click **Update**. The Payment Configuration screen appears showing the selected DDNs.



Payment Configuration

Global Configuration | Payee Account Configuration | Indexer DDN Configuration

Payee DDN	Action	Payment Type	Gateway
payment1	Create	check	-
	Create	ccard	-
payment2	Create	check	-
	Create	ccard	-

8 Double-click **Create** next to the **regular payee DDN** for **checks**.

9 Select a payment gateway from the drop-down list and click **Add**. The ACH check cassette configuration screen appears.

10 Specify the payment gateway settings for checks.

For details on check payment settings, see “Shared Check and Credit Card Gateway Configuration Parameters” on page 62 and “Check Payment Gateway Configuration (Regular Payee DDN Only)” on page 63. Note that fields with an asterisk (*) are required.

(If you are migrating, use the same settings as the old payment configuration noted before deleting the old payment configurations.)

11 Click **Add**. The Payment Configuration screen appears, showing the type of gateways configured.

12 For the same DDN, double-click **Create** next to the **regular payee DDN** for **credit cards** (ccard).

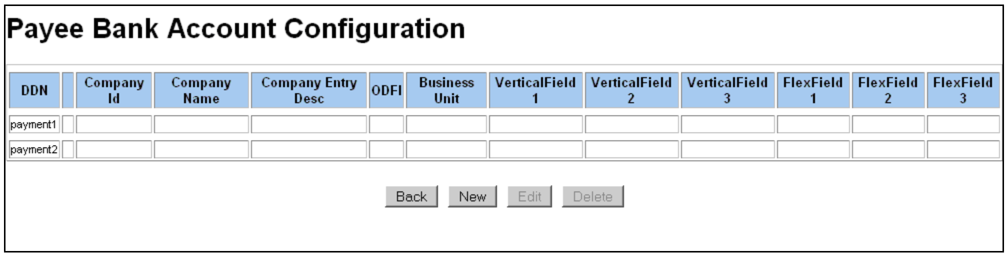
13 Select a credit card gateway from the drop-down list of available credit card processors and click **Add**. The Payment Configuration screen appears.

14 Specify the credit card gateway settings for credit cards.

For details on credit card gateway settings, see "Shared Check and Credit Card Gateway Configuration Parameters" on page 62 and "Credit Card Payment Gateway Configuration (Regular Payee DDN Only)" on page 68. Note that fields with an asterisk (*) are required.

15 Click **Add.** The Payment Configuration screen appears, showing the type of gateways configured.

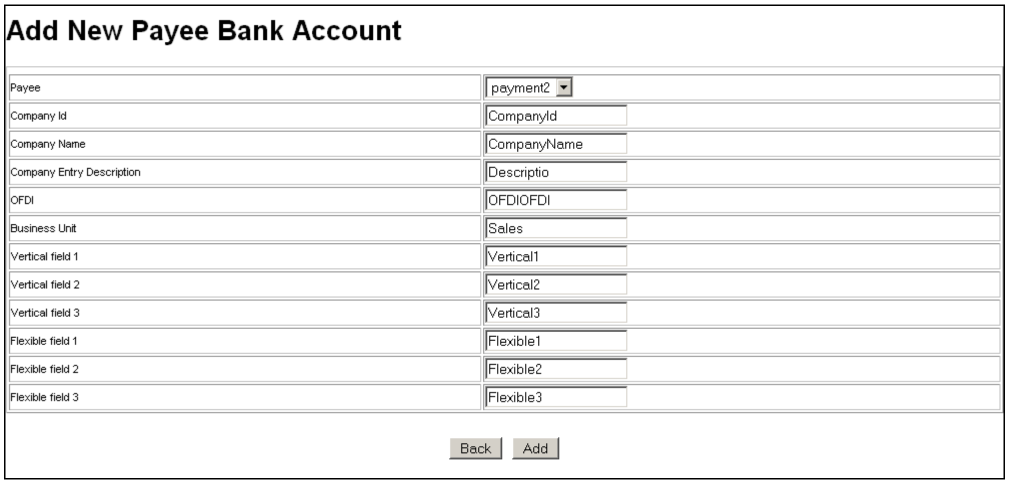
16 Click **Payee Account Configuration.** The Payee Bank Account Configuration screen appears.



Payee Bank Account Configuration

DDN		Company Id	Company Name	Company Entry Desc	ODFI	Business Unit	VerticalField 1	VerticalField 2	VerticalField 3	FlexField 1	FlexField 2	FlexField 3
payment1												
payment2												

17 Click **New.** The Add New Payee Bank Account screen appears.



Add New Payee Bank Account

Payee	payment2
Company Id	CompanyId
Company Name	CompanyName
Company Entry Description	Descriptio
OFDI	OFDIOFDI
Business Unit	Sales
Vertical field 1	Vertical1
Vertical field 2	Vertical2
Vertical field 3	Vertical3
Flexible field 1	Flexible1
Flexible field 2	Flexible2
Flexible field 3	Flexible3

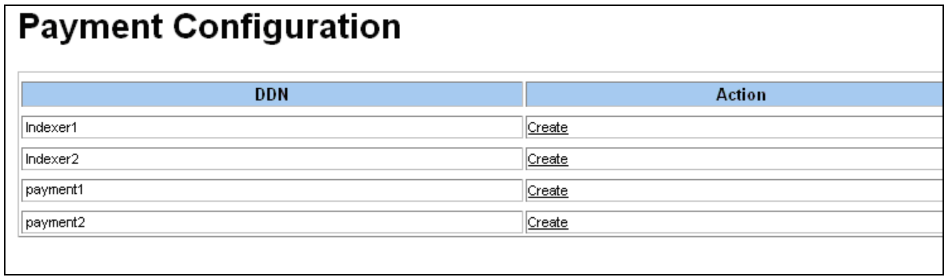
18 Select a payee DDN from the drop-down list. Specify bank account details for the payee DDN. (The values of Company ID, Company Name, Company Entry Desc, and ODFI can be same as the values of the corresponding fields of the old (deleted) ACH check payment configurations).

You must configure at least one payee bank account for each Payee DDN.

See "Payee Bank Account Configuration" on page 61 for details.

19 Click **Add** to save the settings. The Payment Configuration screen reappears.

20 Click **Indexer DDN Configuration**. You can define an indexing job using the payment DDNs or a separate DDN created for indexing. In either case, you must configure the indexer DDN for use with ePayment. The Payment Configuration screen reappears:



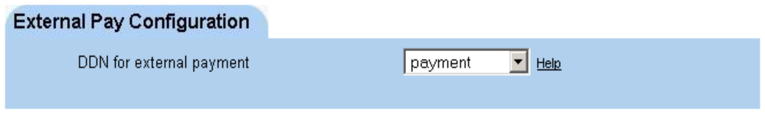
Payment Configuration

DDN	Action
Indexer1	Create
Indexer2	Create
payment1	Create
payment2	Create

21 Click **Create** to add indexer DDN settings for the indexer DDN. (These values can be the same as the values of the corresponding shared fields of the old payment configurations.) See "Indexer DDN Configuration Parameters" on page 70 for parameter descriptions.

22 Click **Add** to save the settings. Configure each indexing DDN, if necessary.

23 Click **Global Configuration**. The Payment Global Configuration screen appears.



External Pay Configuration

DDN for external payment: payment

24 Under the Payee DDN Configuration section, select the DDN you defined for external payments from the drop-down list. Then also select the external DDN from the External Pay Configuration section, and click **Update**. The Payment Configuration screen appears.

25 Click **Create** next to the **external payee DDN** for checks. The ACH check cassette configuration screen appears. Add the payment gateway settings for checks.

For details on check payment settings, see "Shared Check and Credit Card Gateway Configuration Parameters" on page 62 and "Check Payment Gateway Configuration (External Payee DDN Only)" on page 67.

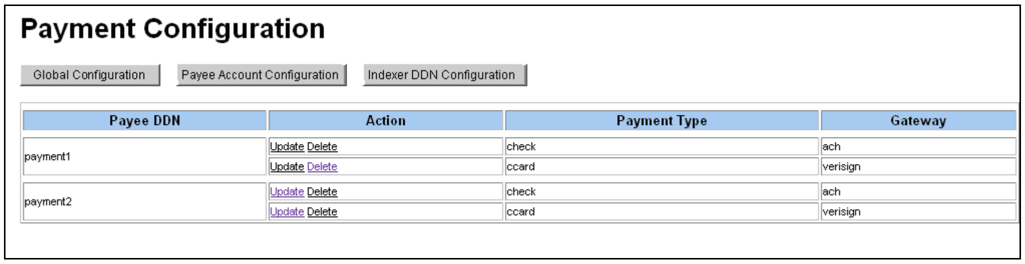
(If you are migrating, use the same settings as the old payment configuration noted before deleting the old payment configurations.)

26 Click **Add**. Repeat Step 25, clicking **Create** next to the external payee DDN for **credit cards (ccard)**.

For details on credit card payment settings, see "Shared Check and Credit Card Gateway

Configuration Parameters" on page 62 and "Credit Card Payment Gateway Configuration (External Payee DDN Only)" on page 69.

27 Once you have configured the gateways for the regular payment DDN, the Payment Configuration screen appears, showing the type of gateways configured:



Payee DDN	Action	Payment Type	Gateway
payment1	Update Delete	check	ach
	Update Delete	ccard	verisign
payment2	Update Delete	check	ach
	Update Delete	ccard	verisign

28 Click **Add**. The payment Configuration Screen appears. (To update any payment gateway settings in the future, use the Update option on this screen.)

Note: To delete a payee bank account, select the account from the Payee Account Configuration screen, and click Delete.

Payee Bank Account Configuration

The configuration parameters for the Add New Payee Bank Account configuration screen are:

Payee - The name of the payee DDN. The payee bank account will be assigned to this payment DDN.

Company ID - This information is assigned to you by your bank, and follows a format specific to the needs of the ODFI. The pmtCheckSubmit job inserts this information into the ACH Batch Header record's Company ID field. The pmtCheckUpdate job uses this value to identify the biller in the ACH return file.

Company Name - The name given to the Biller by the Biller's bank (ODFI). This information is assigned to you by your bank, and follows a format specific to the needs of the ODFI. The pmtCheckSubmit job inserts this information into the ACH File Header record's Company Name field. The pmtCheckUpdate job uses this value to identify the biller in the ACH return file.

Company Entry Description - Describes the purpose of the detail records, and is dependent on the type of details records. For example, this could be "Gas Bill" if the ACH Detail records following this Batch Header record are of type PPD. This value provided by your bank (the payment gateway). The pmtCheckSubmit job inserts this information into the ACH Batch Header record's Company Entry Description field.

ODFI - The routing number of the Biller's bank (ODFI). This value provided by the payment gateway. The pmtCheckSubmit job inserts this information into the ACH Batch Header record's Originating DFI ID field.

Business Unit – The name of the business unit associated with this account.

Vertical Field 1, Vertical Field 2, Vertical Field 3 - Fields that can be customized for use with your application.

Flexible Field 1, Flexible Field 2, Flexible Field 3 – Fields that can be customized for use with your application.

Shared Check and Credit Card Gateway Configuration Parameters

The following parameters are shared by both the credit card cartridge (if any) and check cartridges for both regular and external payee DDNs:

Batch Size for Payment Reminder Table - Specifies the number of payment reminders to be read into memory from the ePayment Manager database for the pmtPaymentReminder job. Note that specifying a batch size that is too small increases the number of times the database is accessed, and specifying a batch size value that is too large could result in an excessive amount of memory being used.

A batch size of 100 is suggested for a medium-sized database, and a batch size of 1000 is suggested for a large database.

You can enter a batch size of 0 (zero) to disable batched table reads, but this is not recommended because it requires a lot of system memory. Entering zero means that one partition ID is created for all payments, and that all payments are processed at once instead of in batches. This also means that the resulting batch file does not have multiple batch records, which some banks prefer.

Send Email Notification when Payment Jobs are Done (with or without error) - **Y** enables and **N** disables sending of email about the status of the ePayment jobs that support job status notification. Additional email information is specified in the following fields.

Mail-to addresses (separated by ";", semicolon) for job status notification - One or more email addresses that should be sent job status notification, separated by semicolons.

JNDI name of IAccount - Implementation of IAccount, which must match the enrollment model (single or multiple DDNs per user account) used by applications that use this payment gateway (DDN).

Implementation of IUserAccountAccessor - The name of the class that handles getting eStatement user information, which is determined by the type of enrollment supported. The options are:

> *com.edocs.payment.payenroll.usracct.JNDISingleDDNUserAccountAccessor*, for when single DDN eStatement user information is stored in CDA.
>
> Or:
>
> *com.edocs.payment.payenroll.usracct.JNDIMultipleDDNUserAccountAccessor*, for when multiple DDN eStatement user information is stored in CDA.

Select the class you want to use from the drop-down list on the right, and it appears in the left box.

Implementation of IPaymentAccountAccessor - The name of the class that handles getting ePayment user information. The options are:

> *com.edocs.payment.payenroll.payacct.SSOPaymentAccountAccessor*, for when ePayment user information is stored in CDA.

Select the class you want to use from the drop-down list on the right, and it appears in the left box.

Make Authorize Reversal Payments - **Y** - Yes, **N** - No

Check Payment Gateway Configuration (Regular Payee DDN Only)

This section describes the parameters for configuring the ACH and CheckFree payment gateways. These parameters are shown when configuring a regular payee DDN.

ACH Gateway Parameters

This section describes the configurable parameters for an ACH payment gateway.

Payment Type – The type of payment: Check.

Gateway - The type of payment gateway: ACH.

(See "Shared Check and Credit Card Gateway Configuration Parameters" on page 62 for descriptions of configuration parameters shared by both check and credit card cartridges.)

The following parameters are specific to check payment gateways for a regular payee DDN:

Batch Size for Check Payment Table - Specifies the number of scheduled checks to read into memory from the payment database as a batch job. Note that specifying a batch size that is too small increases the number of database accesses, and specifying a batch size value that is too large might result in an excessive amount of memory being used.

A batch size of 100 is suggested for a medium-sized database, and a batch size of 1000 is suggested for a large database.

A batch size of 0 (zero) may be entered to disable batched table reads, but it is not recommended because it requires a lot of system memory. Entering zero means that one partition ID is created for all payments, and that all payments are processed at once, instead of in batches. This also means that the resulting ACH file will not have multiple batch records, which some banks prefer.

Number of Business Days After Pay Date to Clear the Check - The number of business days for a check to be marked as paid after submitting to the payment gateway without returns or failures. The default is five business days.

Days to Activate Pending Subscribers - The number of business days to wait before approving a customer for online payment. If a prenote file is returned before this time, then the customer is rejected for the causes stated in the prenote file. The default is three days.

Batch File Name Prefix - The text used as a prefix to each batch file name.

Batch File Name Time Stamp - The format used for each batch file time stamp.

Batch File Name Suffix - The text used as a suffix to each batch file name.

ACH Prenote File Name Prefix - The text used as a prefix to each ACH prenote file name.

ACH Prenote File Name Time Stamp - The format used for each ACH prenote file name.

ACH Prenote File Name Suffix - The text used as a suffix to each ACH prenote file name.

Update Payment enrollment in Case of NOC - "Y" causes the pmtCheckUpdate job to update enrollment status when a NOC is returned. "N" means that this value is updated by the customer through the user interface.

Send Email Notification in Case of NOC - "Y" causes the pmtNotifyEnroll job to send emails to customers whose payment returns a NOC. "N" means the customer does not receive email when a NOC is returned.

Skip non-business days for batch entry effective dates - Determines whether batch effective dates skip U.S. federal holidays and weekends.

Generate an empty ACH file if no checks to submit - Determines whether the pmtCheckSubmit job generates an empty ACH file if there a no checks to submit.

Immediate Destination - The routing number of the Biller (DDN). This information is assigned to you by your bank, and follows a format specified by the Biller's bank (ODFI). The pmtCheckSubmit job inserts this information into the ACH File Header record's Company Name field. The pmtCheckUpdate job uses this value to validate entries in the ACH return file. Must be exactly 10 characters in length and must start with a blank; the leading blank is counted as one of the 10 characters.

Immediate Origin - The Routing Number of the Biller's bank (ODFI). This number is assigned to you by your bank. The pmtCheckSubmit job writes this information to the ACH File Header record's Immediate Destination field. The pmtCheckUpdate job uses this value to validate entries in the ACH return file. Must be exactly 10 characters in length. The value assigned to you by your bank may or may not have a leading blank. If the value is less than 10 characters in length (including the leading blank, if there is one), you must pad the entry with trailing blanks to reach a total length of 10 characters.

Immediate Destination Name - The name of the Biller. This information is assigned to you by your bank, and follows a format specific to the needs of the ODFI. The pmtCheckSubmit job inserts this information into the ACH File Header record's Immediate Destination Name field. The pmtCheckUpdate job uses this value to validate entries in the ACH return file.

Immediate Origin Name - The name of the Biller's bank (ODFI). This information is assigned to you by your bank, and follows a format specific to the needs of the ODFI. The pmtCheckSubmit job inserts this information into the ACH File Header record's Immediate Destination Name field. The pmtCheckUpdate job uses this value to validate entries in the ACH return file.

ACH File Output Directory - Directory where ACH files are created to be sent to the originating bank. ePayment does not create this directory.

ACH File Input Directory - Directory where the originating bank sends ACH return files. ePayment does not create this directory.

ACH Template File Directory - Directory where ACH XML files are stored. The default for UNIX is: *PAYMENT_HOME/lib/payment_resources/ach/template*.

Implementation of IAchCheckSubmitPlugIn - The plugin allows modification of whether a check payment is submitted, plus other actions based on a check selected for payment. For example, to generate a remittance file with a format different from the standard ACH file specification.

For information about implementing this class, contact Oracle Professional Services. The default is *com.edocs.payment.cassette.ach.AchCheckSubmitPlugIn*.

Flexible Field 1 and 2 - *IAchCheckSubmitPlugin* can take up to two parameters, which can be specified here.

ACH Federal Holidays

ACH check payment gateways have the field *Skip non-business days for batch effective entry date* to determine when a payment should be made. Non-business days in ePayment Manager include the following U.S. federal holidays:

Holiday	**Date**
New Years Day	January 1
Martin Luther King's Birthday	Third Monday in January
Presidents' Day	Third Monday in February
Memorial Day	Last Monday in May
Independence Day	July 4
Labor Day	First Monday in September
Columbus Day	Second Monday in October
Veterans' Day	November 11
Thanksgiving	Fourth Thursday in November
Christmas Day	December 25

CAUTION: If a U.S. federal holiday falls on a Saturday, then the previous Friday is a holiday for Federal employees, but it is not a holiday for most businesses and employees.

CheckFree Gateway Parameters

This section describes the configurable parameters for a (CheckFree) CDP payment gateway.

(See “Shared Check and Credit Card Gateway Configuration Parameters” on page 62 for descriptions of configuration parameters shared by both check and credit card cartridges.)

The following parameters are specific to checkfree payment gateways for a regular payee DDN:

Batch Size for Check Payment Table - Specifies the number of scheduled checks to be read into memory from the ePayment database for the pmtCheckSubmit and pmtCheckUpdate jobs. Note that specifying a batch size that is too small increases the number of times the database is accessed, and specifying a batch size value that is too large might result in an excessive amount of memory being used.

A batch size of 100 is suggested for a medium-sized database, and a batch size of 1000 is suggested for a large database.

A batch size of 0 (zero) may be entered to disable batched table reads, but it is not recommended because it requires a lot of system memory. Entering zero means that one partition ID is created for all payments, and that all payments are processed at once, instead of in batches. This also means that the resulting ACH file will not have multiple batch records, which some banks prefer.

Number of business days after pay date to clear the check - The number of business days for a check to be marked as paid after submitting to the payment gateway without returns or failures. The default is five business days.

Days to Activate Pending Subscribers - The number of business days to wait before approving a customer for online payment. If a prenote file is returned before this time, then the customer is rejected for the causes stated in the prenote file.

Batch File Name Prefix - The text used as a prefix to each batch file name.

Batch File Name Time Stamp - The format used for each batch file time stamp.

Batch File Name Suffix - The text used as a suffix to each batch file name.

Client ID - Supplied by CheckFree. It is unique for each customer and must be set up in advance. CheckFree requires four weeks to set up a new account for a customer.

Sender ID - Supplied by CheckFree.

FTP User Name - Supplied by CheckFree. Files are transmitted to and from CheckFree using FTP via the Internet. CheckFree only allows transfers from specific FTP user names that originate from specific IP addresses.

File Transmission Mode - Select either Test or Production. This value is included in the name of the debit file that eStatement creates, and is handled differently by CheckFree when the file is received Test files are run through the CheckFree processes but no debits are actually made.

Generate Empty File - **Y** causes pmtCheckSubmit to generate an empty CDP when there are no checks. This file will contain 0000, 3000, or 9999 records (but no 4000 records). **N** causes no empty CDP file to be created (**default**).

Payee Number - Supplied by CheckFree.

Payee Short Name-This value must be present, but is not always validated by CheckFree. Contact CheckFree to see if they have a value that must be used. If not, make one up.

Payee Name - Matches Payee Short Name.

Payee Address - Street address for the customer, which should match the information CheckFree has on file.

Payee City - This information should match CheckFree's information.

Payee State - This information should match CheckFree's information.

Payee Zip - This information should match CheckFree's information.

CDP File Output Directory - Directory where pmtCheckSubmit creates debit files.

CDP File Input Directory - Directory where pmtCheckUpdate should look for files to process.

Template File Directory - For a standard installation, should be:

<PAYMENT_HOME>/lib/payment_resources/checkfree/cdp/ (UNIX)

C:\<PAYMENT_HOME>lib\payment_resources\checkfree\cdp\template (Windows).

Check Payment Gateway Configuration (External Payee DDN Only)

This section describes the parameters for configuring the ACH and CheckFree payment gateways. These parameters are shown when configuring a regular payee DDN.

(See "Shared Check and Credit Card Gateway Configuration Parameters" on page 62 for descriptions of configuration parameters shared by both check and credit card cartridges.)

The following parameters are specific to check payment gateways for an external payee DDN:

Batch Size for Check Payment Table - Specifies the number of scheduled checks to read into memory from the payment database as a batch job. Note that specifying a batch size that is too small increases the number of database accesses, and specifying a batch size value that is too large might result in an excessive amount of memory being used.

A batch size of 100 is suggested for a medium-sized database, and a batch size of 1000 is suggested for a large database.

A batch size of 0 (zero) may be entered to disable batched table reads, but it is not recommended because it requires a lot of system memory. Entering zero means that one partition ID is created for all payments, and that all payments are processed at once, instead of in batches. This also means that the resulting ACH file will not have multiple batch records, which some banks prefer.

Number of Business Days After Pay Date to Clear the Check - The number of business days for a check to be marked as paid after submitting to the payment gateway without returns or failures. The default is five business days.

Days to Activate Pending Subscribers - The number of business days to wait before approving a customer for online payment. If a prenote file is returned before this time, then the customer is rejected for the causes stated in the prenote file. The default is three days.

Batch File Name Prefix - The text used as a prefix to each batch file name.

Batch File Name Time Stamp - The format used for each batch file time stamp.

Batch File Name Suffix - The text used as a suffix to each batch file name.

Batch File Output Directory - The name of the directory to place the batch output file.

Credit Card Payment Gateway Configuration (Regular Payee DDN Only)

This section describes the configurable parameters for a credit card payment gateway using the VeriSign payment processor. These parameters are shown when configuring a regular payee DDN.

(See "Shared Check and Credit Card Gateway Configuration Parameters" on page 62 for descriptions of configuration parameters shared by both check and credit card cartridges.)

The following parameter is listed, but cannot be changed from this screen:

Gateway - The name of the credit card cassette to be used for this payment gateway. The same cassette can be used for more than one credit card payment gateway.

The following parameters are specific to a VeriSign gateway for regular payee DDN:

Batch Size for Credit Card Payment Table - Specifies the number of scheduled credit card payments to be read into memory from the ePayment database for the pmtCreditCardSubmit job. Note that specifying a batch size that is too small increases the number of times the database is accessed, and specifying a batch size value that is too large might result in an excessive amount of memory being used.

A batch size of 100 is suggested for a medium-sized database, and a batch size of 1000 is suggested for a large database.

A batch size of 0 (zero) may be entered to disable batched table reads, but it is not recommended because it requires a lot of system memory. Entering zero means that one partition ID is created for all payments, and that all payments are processed at once, instead of in batches. This also means that the resulting batch file will not have multiple batch records, which some banks prefer.

VeriSign Host Name - The URL to the VeriSign host that will process credit card transactions for this payment gateway. The options are: "test-payflow.verisign.com" (for testing) and "payflow.verisign.com" (for production).

VeriSign Host Port - The TCP port number to be used when contacting the VeriSign host. The default is 443.

VeriSign Timeout Period for Transaction - The number of seconds the pmtCreditCardSubmit job will wait for a transaction to complete with the VeriSign Host before timing out.

VeriSign User - The case-sensitive vendor name from VeriSign. This should match VeriSign Vendor.

VeriSign Vendor - The case-sensitive vendor name assigned by registering with VeriSign.

VeriSign Partner - If PayFlow service is provided by a VeriSign Reseller, the reseller ID, otherwise, "VeriSign".

VeriSign Password - The case-sensitive password assigned by VeriSign.

VeriSign Certificate Path - The path to the SSL server certificate purchased from VeriSign and installed on the application server.

Number of Threads - Specifies the number of connections to open with the VeriSign payment gateway at one time. More threads consume more system and network resources, but decrease the time it takes the pmtCreditCardSubmit job to complete processing credit card payments. **The maximum allowed is 10; the default is 1**.

Enable VeriSign address verification service - **Y** (default) enables address verification for credit card payments. AVS support must also be set up with Verisign. See "AVS (Address Verification Service)" on page 36 for more information.

Implementation of IVeriSignCreditCardSubmitPlugIn - The plugin allows modification of whether a credit card payment is submitted, plus other actions based on the payments selected for settlement. For example, to deny a credit card payment based on additional business rules.

For information about implementing this class, contact Oracle Professional Services. The default is: *com.edocs.payment.cassette.verisign.VeriSignCreditCardSubmitPlugIn*.

Flexible Field 1 and 2 - The plugin can take up to two parameters, which are specified here.

Credit Card Payment Gateway Configuration (External Payee DDN Only)

This section describes the configurable parameters for a credit card payment gateway using the VeriSign payment processor. These parameters are shown when configuring an external payee DDN.

(See "Shared Check and Credit Card Gateway Configuration Parameters" on page 62 for descriptions of configuration parameters shared by both check and credit card cartridges.)

The following parameters are specific configuring a credit card gateway for an external payee DDN:

Batch Size for Credit Card Payment Table - Specifies the number of scheduled credit card payments to be read into memory from the ePayment database for the pmtCreditCardSubmit job. Note that specifying a batch size that is too small increases the number of times the database is accessed, and specifying a batch size value that is too large might result in an excessive amount of memory being used.

A batch size of 100 is suggested for a medium-sized database, and a batch size of 1000 is suggested for a large database.

A batch size of 0 (zero) may be entered to disable batched table reads, but it is not recommended because it requires a lot of system memory. Entering zero means that one partition ID is created for all payments, and that all payments are processed at once, instead of in batches. This also means that the resulting batch file will not have multiple batch records, which some banks prefer.

Batch File Name Prefix - The text used as a prefix to each batch file name.

Batch File Name Time Stamp - The format used for each batch file time stamp.

Batch File Name Suffix - The text used as a suffix to each batch file name.

Batch File Output Directory - The name of the directory to place the batch output file.

Indexer DDN Configuration Parameters

DDN

Implementation of com.edocs.payment.imported.IBillDepot - The default *com.edocs.payment.imported.eadirect.BillDepot* retains the default handling for bill processing. Choosing *com.edocs.payment.imported.eadirect.SampleBillDepot* adds the following features:

- If the rebill has a zero amount, the due date is set to the previous bill's due date.
- If the due date is "ON RECPT", the due date is changed to the current date.

Name of Due Date in Statement Index Table (for recurring payment) - The name of the field you have defined to extract the Due Date from each statement. This Field **must** be routinely indexed in the eStatement database, during data processing, for use in ePayment Manager. See the *Administration Guide for Oracle Siebel eStatement Manager* for details.

TIP: This field only applies to recurring payments. If you are not implementing recurring payments, you can leave this field empty.

Due Date Format - Selected from the dropdown list, the format of the data extracted as the Due Date from each statement. This parameter depends on the format of the legacy data source.

Name of Amount Due in Statement Index Table (for recurring payment) - The name of the field you have defined to extract the Amount Due from each statement. This field **must** be routinely indexed in the eStatement database, during data processing, for use in ePayment Manager. See the *Administration Guide for Oracle Siebel eStatement Manager* for details.

TIP: This field only applies to recurring payments. If you are not implementing recurring payments, you can leave this field empty.

Amount Due Format - Selected from the dropdown list, the format of the data extracted as the Amount Due from each statement. This setting depends on the format of the legacy data source.

Name of Minimum Amount Due in Statement Index Table (if applies) - The name of the variable defined by the DefTool for minimum amount due, which must be indexed by eStatement. If scheduling options that require minimum amount due are not going to be offered in the customer interface, then this value should be left blank.

Minimum Amount Due Format - Selected from the dropdown list, the format of the data extracted from each statement, as the field that you have designated as the Minimum Amount Due. This setting depends on the format of the legacy data source.

Global Configuration Parameters

The configuration parameters on the Payment Global Configuration screen are described here.

Credit Card Account

Save Full Credit Card Number in Payment DB ("N" means only save last four digits): **Y** saves the full credit card number in the ePayment Manager database. **N** saves only last four digits of the credit card number.

Encrypt Credit Card Number: Specifies whether or not to encrypt credit card information in the ePayment database. This is highly recommended if you choose to save the entire credit card number, which is specified in the previous configuration parameter.

Check Account

Encrypt Check Account Number: Y causes ePayment Manager to encrypt check account numbers in the Payment database.

Payee DDN Configuration

Payee DDNs: Choose one or more regular payment DDNs to configure.

Notification Service Configuration

Payment Notification Service Implementation: The default payment notification service implementation appears. Specify a different one for the selected applications only if you have created a custom implementation.

External Pay Configuration

DDN for external payment: Choose the DDN you created for external payments when you want to configure this DDN (you configure the external DDN separately).

Audit Configuration

Enable Payment Audit: Y causes any actions performed that affect the ePayment tables *check_payments* or *creditcard_payments* to be audited. These actions can be from the Web application or from the ePayment jobs. **N** disables auditing. The default is **Y**.

Enable Recurring Payment Audit: Y causes any actions that affect the *recurring_payments* tables to be audited. These actions can be from the Web application or from the ePayment jobs. **N** disables auditing. The default is **Y**.

Enable Payment Account Audit: Y causes any actions that affect the *payment_accounts* table to be audited. These actions can be from the Web application or from the ePayment jobs. **N** disables auditing. The default is **N**.

Enable Payment Reminder Audit: Y causes any actions that affect the *payment_reminders* table to be audited. These actions can be from the Web application or from the ePayment jobs. **N** disables auditing. The default is **N**.

Enable Bill Summary Audit: Y causes any actions that affect the *payment_bill_summaries* table to be audited. These actions can be from Web application or from the ePayment jobs. **N** disables auditing. The default is **Y**.

Email Notification Audit

Email Content Audit Length: Specifies how many characters of an email's content is audited. The default is 100. The audit length must be between 0 and 2048.

Email Content AuditOffset: Specifies where to start the audit in the email content.

Enable Recurring Payment Notification Audit: Y causes emails sent out by the recurring payment job to be audited. **N** disables email auditing. The default is **N**.

Enable Payment Reminder Notification Audit: Payment reminders send out two kinds of emails: remind-pay-bill emails and payment status notification emails. **Y** causes the remind-pay-bill emails (fixed-date payment reminder, pre-due-date payment reminder and post-due-date reminder) to be audited. The default is **N**.

Enable Payment Status Notification Audit: Y causes payment status notification emails to be audited for both check and credit card payments. **N** disables payment status notification email auditing. The default is **N**.

Enable Payment Account Enrollment Notification Audit: Y causes enrollment notification emails sent by the pmtNotifyEnroll job to be audited. **N** disables enrollment notification emails auditing. The default is **N**.

Enable Credit Card Expiration Notification Audit: Y causes credit card expiration emails sent by the pmtCreditCardExpNotify job to be audited. **N** disables credit card expiration emails auditing. The default is **N**.

9 Configuring ePayment Manager Jobs

Regular Payment Jobs

You must configure the following payment jobs to enable payment processing.

pmtAllCheckTasks Job

Configuring the pmtAllChecksTasks job runs the following ePayment check payment jobs sequentially:

- pmtRecurringPayment
- pmtCheckSubmit
- pmtCheckUpdate
- pmtPaymentReminder
- pmtSubmitEnroll
- pmtConfirmEnroll
- pmtNotifyEnroll
- pmtARIntegrator

This job presents all the configuration menus from all the other jobs, so you can configure all listed jobs from this job.

You can also edit pmtAllCheckTasks to not run specific tasks if you want to tailor your environment.

pmtARIntegrator Job

pmtARIntegrator uses an XLST template to translate data extracted from ePayment tables into a different file format. This job runs queries against the *check_payments* and *creditcard_payments* tables to populate a file formatted according to an XML template. Then it uses XLST to transform that data into another file in the format specified by the XLST template.

Configuration

The configurable parameters for this job are:

Full Path Name of XML Query File - Enter the path to the general query template file. The default *PAYMENT_HOME/lib/payment_resources/ar/arQuery.xml*, which you can modify to fit your needs. See the sample *arQuery.xml*, where a check query and a credit card query are shown.

Check Query Name in XML Query File- Enter the value of the “name” attribute of the “query” element in *arQuery.xml*, which is used for the query against the *check_payments* table. This query only works for the *check_payments* and *check_payments_history* tables. You can modify the values, or add new query elements.

Credit Card Query Name in XML Query File - Enter the value of the “name” attribute of the “query” element in *arQuery.xml*. This query currently only works for the *creditcard_payments* and *creditcard_payments_history* tables. You can modify the values, or add new query elements.

Implementation of IARPaymentIntegrator - Specifies the implementation class for IPaymentIntegrator. The default parameter is *com.edocs.payment.tasks.ar.SampleARPaymentIntegrator*. The IPaymentIntegrator interface defines a method to use ePayment Manager to generate A/R files in a specific format. See the *Customizing and Extending Oracle Siebel ePayment Manager* guide for information about modifying and implementing a custom class for a plugin.

Full Path of AR Template File - The complete path to the payment source template file. The default parameter is *PAYMENT_HOME/lib/payment_resources/ar/arFlat_template.txt*. The default file is a sample flat text template file that shows how to format output, which you can edit to meet your requirements. The other template file provided with the ARIntegrator job is arXML_template.xml.

Directory for Output AR File – Specify a directory to put the output file (*PAYMENT_HOME/Output/AR or other location*).

Transform Output AR File to Another Format? - This flag is ignored unless an XML template is chosen. If it is “Y”, the job takes the generated XML file in the output directory as XML input for the XSLT processor, reads the XSLT template, and transforms the data into a different file format.

CAUTION: Do not set this field to "Y" if the XLST file used to transform the AR file to a different format is in TXT format. Only enter Y if the XLST template file format is well formed XML.

Full Pathname of XLST File Used for Transform - The XSLT template file. You can create your own XSLT template file in a different directory with a different name. The default *PAYMENT_HOME/lib/payment_resources/ar/arTransform.xsl* is a sample flat text template file that shows how to format output, which you can edit to meet your requirements.

After specifying the preceding parameters and scheduling the job, the A/RIntegrator job generates an output file in output directory based on your check and credit card query criteria as specified in *arQuery.xml* and the Java class *ARPaymentIntegrator.java*. You can modify the sample templates files, and re-implement IPaymentIntegrator interface to add features, if desired. For more information about re-implementing the IPaymentIntegrator interface, see the *Customizing and Extending Oracle Siebel ePayment Manager* guide.

CAUTION: The XLST template file must be well-formed XML or the pmtARIntegrator job fails with the error: java.lang.exception: javax.xml.transform.TransformerException.

Directory for Transformed File - The directory for the final transformed file. The default is *PAYMENT_HOME/Output/ar*.

File Name Extension for Transformed File - The file extension of the final transformed file. The default is TXT.

Flexible Parameter 1 (for customization) - Optional parameter used in the AR query.

Flexible Parameter 2 (for customization) - Optional parameter used in the AR query.

pmtCheckUpdate Job

pmtCheckUpdate updates a check's status according to the response from the payment gateway. All files that match the necessary criteria are processed and moved to a history directory under the input directory. The *check_payments* table is examined, and any checks whose status is "processed", and whose pay date is five or more days ago, and has not been returned will have their status changed to "paid".

After the job completes processing, it moves all the processed files to a *history* directory under file input directory.

pmtCheckUpdate Configuration

None required.

ACH Logic

pmtCheckUpdate processes return files from ACH, which can include checks from multiple DDNs. pmtCheckUpdate compares the immediate origin (routing number of the ODFI), immediate destination, immediate origin name, and immediate destination name in the ACH File Header to the same fields that are configured for the ACH payment gateway. If they are the same, pmtCheckUpdate continues processing the ACH return file. If they are not the same, check is ignored.

The order of Immediate Destination and Immediate Origin can be swapped in the ACH return file from what the ACH specification recommends and the pmtCheckUpdate job will still process the file correctly.

For each Batch Header record, pmtCheckUpdate matches the *Company Name* and *Company ID* against the payment gateway definition. If they match, then the current DDN's payment setting is used to process this batch. If either one does not match, pmtCheckUpdate searches the Payment Settings of all DDNs. If it finds a DDN setting with the same immediate information and company information, it uses that setting to process the checks in that batch. If it cannot find a match, it raises an exception and the job fails.

For each successful Batch Header record match, pmtCheckUpdate updates the status according to the following table:

Addenda Type	Amount field in the first detail record	Return	Status change
99	0	prenote error	prenote_returned
98	0	NOC	noc_returned
99	not 0	check error	processed

pmtCheckUpdate also updates the status to "paid" if there is no return from ACH after the configured number of days. The number of days to wait after the pay date can be changed in the configuration settings for an ACH payment gateway.

TIP: If pmtCheckUpdate will be processing ACH return files containing multiple DDNs, then the Payment Settings for each payment gateway must have the same **immediate origin** and **immediate**

destination. Also, each payment gateway used by those DDNs must use the same ACH template files (ACH Template Directory parameter).

ACH Returns

There may be three kinds of returns in one return file:

- check returns
- NOC returns
- prenote returns

For **check returns**, the corresponding check in the *check_payments* table is identified by either payment ID or gateway payment ID. The check status is updated to "returned", and the **txn_err_msg** field is set to the return code.

The following columns are updated in the *check_payments* table after a check return is processed:

Column updated	Value
last_modify_time	current time
status	-4
reminded	“N”
log_id	id of the exception report in the *payment_log* table
txn_err_msg	ACH return code

For **prenote returns**, the corresponding prenote check in the *check_payments* table is identified by either payment ID or gateway payment ID. The prenote's status is then updated to "prenote_returned", and the *txn_msg_err* column is set to the return code. From the prenote check, the payer_id (which is the *user_id* column in the *payment_accounts* table) is used to update payment enrollment information. The payment account with the same user ID and payment account number will be updated to "bad_active" (*account_status* column) and *txn_message* is set to the ACH return code.

The following columns are updated in the *payment_accounts* table after a prenote is processed:

Column updated	Value
txn_message	return ACH code
account_status	bad_active
notify_status	“N”

NOC Returns

For NOC returns, the corresponding check (which can be either regular or prenote) is identified by either payment ID or gateway payment ID. The NOC's payer_id identifies the corresponding account (ePayment Manager matches it with the *user_id* column in the *payment_accounts* table).

If the auto update flag Update Payment enrollment in Case of NOC field was set to true ("Y") in the Payment Settings, then the corresponding payment account information (routing, acct number or account type) is updated based on the NOC code. If the flag is false ("N"), then the current payment account information is not changed. In either case, the *txn_message* column is populated with the following format:

```
NOC_CODE::NEW_ADDENDA_INFO::OLD_ADDENDA_INFO
NOC_CODE::NEW_ADDENDA_INFO::OLD_ADDENDA_INFO
```

Where:

NOC_CODE is the returned NOC code,

NEW_ADDENDA_INFO is the correct NOC information returned from ACH (content from position 36 to 64 of addenda record, with white spaces trimmed),

OLD_ADDENDA_INFO is the existing or incorrect addenda information with the same format as *new_addenda_info* and is calculated on current payment account information.

notify_status is set to “N” if Notify NOC flag is set to “Y” in Payment Settings. If Send Email Notification in Case of NOC is set to "Y" in Payment Settings, then *notify_status* is set to N". ePayment Manager keeps both the old and new payment account addenda information, which allows pmtCheckUpdate to send an email containing both the old and new account information.

The following columns are updated in the *payment_accounts* table after a NOC is processed:

Column updated	**Value**
txn_message	NOC_CODE::NEW_ADDENDA_INFO::OLD_ADDENDA_INFO
account_number	Changed for C01, C03, C06, C07
routing_transit	Changed for C02, C03, C07
account_type	Changed for C05, C06, C07
txn_date	current time
notify_status	“N”

ACH Return File Format

The ACH return file should not include new lines at the end of each record. To ensure that the ACH return files are processed correctly, the following information must be returned correctly:

Name	**Record**	**Description**
immediateDest	fileHeader	Immediate destination must be the same as the one from original ACH file.
immediateOrigin	fileHeader	Immediate origination must be the same as the one from original ACH file.
immediateDestName	fileHeader	Immediate destination name must be the same as the one from original ACH file.

Name	Record	Description
immediateOriginName	fileHeader	Immediate origination name must be the same as the one from original ACH file.
companyName	batchHeader	Company name must be the same as the one from original ACH file.
companyId	batchHeader	Company ID must be the same as the one from original ACH file.
transactionCode	entryDetail	See the ACH specification.
dfiAcctNumber	entryDetail	Check account number; see the ACH specification.
amount	entryDetail	Amount of original check.
individualId	entryDetail	Must be the same as the one from original ACH file.
individualName	entryDetail	Must be the same as the one from original ACH file.
addendaTypeCode	addenda	98: NOC return; 99: check return. See the ACH specification for details
returnReasonCode	addenda	Return code.
originalEntryTraceNumber	addenda	The trace number of the check from the original file sent to ACH.
originalRDFI	addenda	Copy data from positions 04-11 of the original Entry Detail Record. See the ACH specification.
entryAddendaCount	fileTrailer	Total number of addenda returned in the file.
totalDebitAmount	fileTrailer	Total debit amount in the file.
totalCreditAmount	fileTrailer	Total credit amount in the file.

CheckFree Logic

pmtCheckUpdate only processes the files belonging to the payee or DDN of this job.

pmtCheckUpdate checks the third field of the file to determine if the file type is confirm, trnsumm, trnjrnl, setlmnt, or returns. It then verifies that the ClientID field in the file matches the Client ID setting for this gateway configuration. pmtCheckUpdate updates the status of the checks in each file as follows:

File Type	Action
confirm, trnjrnl	Changes the status from processed to paid for all the checks whose pay date is five days before the current day and have no error. An error message results in the status of the checks in that file being changed to “failed.”
returns	Status of checks in this file will be updated to “Returned.”
transumm, setlmnt	These files are not processed, but are still moved to the history directory.

pmtConfirmEnroll Job

This job only applies to the ACH payment gateway.

The pmtConfirmEnroll job checks the *txn_date* field in the *payment_accounts* table to find each new account (the *account_status* field is "pnd_wait") If the specified number of days have passed since the user signed up for enrollment (*txn_date*), pmtConfirmEnroll updates the *account_status* field to "active".

The number of days to wait is specified by the Days to Activate Pending Subscribers parameter in the Payment Settings for the ACH payment gateway.

pmtConfirmEnroll Configuration

No configuration is required when creating this job.

pmtCreditCardExpNotify

For users that have configured a credit card account for payments, this job sends an email to users whose credit card will expire soon.

The configurable parameters for this job are:

Number of days before card expiration date to send email - Specifies the number of days before their credit card expires to send the expiration notice. The default is 7.

Implementation of Interface ICCExpNotificationPlugin - This is the name of the java class that is called before the pmtCreditCardExpNotify job emails notifications. It currently does nothing, but you can replace this class with one of your own to process additional business logic to and possibly modify credit card expiration email, or cancel it completely.

For information about implementing this class, contact Oracle Professional Services. The default value is: *com.edocs.payment.tasks.ccexpnotify.CCExpNotificationPlugIn*, which takes no action.

pmtCustom

The pmtCustom job can be customized to perform tasks that are not part of other ePayment Manager jobs. See the *Customizing and Extending Oracle Siebel ePayment Manager* guide for information about using the pmtCustom job.

pmtNotifyEnroll Job

Sends email notification to customers about the status of their payment account activation, plus account information changes as a result of ACH NOC returns.

No configuration is required for the pmtNotifyEnroll job.

pmtNotifyEnroll Job Email Format

pmtNotifyEnroll sends out email notifications to customers who have successfully enrolled, and to customers who have had problems with their enrollment.

For ACH, pmtNotifyEnroll finds payment accounts in the *payment_accounts* table whose *notify_status* field is "N". It then checks the *account_status* field, and if it is "active" or "bad_active", it sends an email to the customer about the status of their account. After sending email, pmtNotifyEnroll changes the *notify_status* field to "Y". pmtNotifyEnroll does not send emails if *account_status* is "pnd_active" even though the *notify_status* is “N”.

For Checkfree, pmtNotifyEnroll finds payment accounts in the *payment_accounts* table whose *notify_status* is "N", sends email, and changes the *notify_status* to "Y".

The format of the email message that is sent is generated based on the contents of an email template text file. The Full Path of Message Template File configuration parameter specifies the location of this template file. The default is *<PAYMENT_HOME>\lib\payment_resources\notifyEnroll.txt* (Windows) or *<PAYMENT_HOME>/lib/payment_resources/notifyEnroll.txt* (UNIX). This job also uses the payment_account_status.xml.vm template.

pmtPaymentReminder Job

The pmtPaymentReminder job sends payment email notifications to customers:

- Who have configured payment reminders
- When a check's status changes to processed, failed, canceled, or returned.

pmtPaymentReminder Job Configuration

The configurable parameters for this job are:

Implementation of Interface IPaymentReminderPlugin - The plugin allows modification of whether a payment reminder is sent, plus other actions based on the type of reminder. For information about implementing this class, contact Oracle Professional Services. The default is *com.edocs.payment.tasks.reminder.PaymentReminderPlugin*.

Notify if a check is sent for processing? - Indicates whether notification is to be sent for checks that were sent for processing.

Notify if a check is cleared? - Indicates whether notification is to be sent for checks that have been paid (cleared).

Notify if an ACH transaction fails, is returned or gets canceled? - Indicates whether notification is to be sent for checks that were returned by the payment gateway as canceled, returned or failed settlement.

Notify if a credit card is settled - Indicates whether email should be sent for credit card payments that settled successfully.

Notify if a credit card transaction does not get authorized or is canceled - Indicates whether email should be sent for credit card payments that failed to authorize or were canceled by ePayment Manager.

pmtPaymentReminder Operation

Payment Reminders

Email notifications are sent out for customers that have set up payment reminders. pmtPaymentReminder sends out reminder email for reminders in the *payment_reminders* table whose *next_reminder_date* is today or later, and whose *Reminded* field is "N". After sending the email, the *Reminded* field is updated to "Y", and the *next_reminder_date* field is updated as specified by the *reminder_interval* field.

Check Payment Notification

pmtPaymentReminder sends out email for checks as configured in the job. Email notifications can be sent for rows in the *check_payments* table where the *status* field is "Returned", "Failed", or "Processed" (sent for processing). After sending the email, the *Reminded* field is updated to "Y".

The valid values for the *Status* field are:

Status	Value
Returned	-4
Failed	-1
Scheduled	6
Processed	7
Paid	8
Canceled	9

Credit Card Payments

pmtPaymentReminder sends email for the scheduled credit cards in the *creditcard_payments* table whose *status* field is "settled" or "failed to authorize", and whose *reminded* field is "N". After sending email, pmtPaymentReminder sets the *reminded* field to "Y". pmtCreditCardSubmit sets the reminded field back to "N" when it makes a payment for that scheduled credit card.

Normally, no emails are sent for instant credit card payments. You can enable email notification for instant credit card payments by editing the *instantPayment.jsp* file to set the *reminded* field to "N". The instantPayment.jsp file is in web\payment\user\jsp when you un-jar the EAR file. See the *Customizing and Extending Oracle Siebel ePayment Manager* guide for more information about updating and re-deploying ePayment Manager EAR files.

Email Template

The email address is retrieved from the *payer_email_addr* field in the *payment_reminders* table. Email content is created using the email template file configured for this job. The default template file is *paymentReminder.txt*, which is in the *<PAYMENT_HOME>/lib/payment_resources/* directory.

In addition, the template used for the email sent for “who have configured payment reminders” is payment_reminder_fixed.xml.vm.

The template used for the email sent “When a check's status changes to processed, failed, canceled, or returned” is payment_reminder_predue.xml.vm.

(The pmtPaymentReminder job also makes use of the notifyEnroll.txt and payment_account_status.xml.vm templates associated with the pmtNotifyEnroll job.)

See “Using Email Templates” on page 133 for information about editing the email template to modify email content.

pmtRecurringPayment Job

The pmtRecurringPayment job checks for recurring payments that are due for payment and schedules them to be paid. It also optionally sends email to the customer when it schedules a payment, so that the customer can modify or cancel the scheduled payment before the payment is made.

pmtRecurringPayment Configuration

The configurable parameters for this job are:

Task 1: RecurPaymentSynchronizerTask

Implementation of interface IRecurringPaymentPlugIn - The name of the java class that is called before the pmtRecurringPayment job schedules a payment. It currently does nothing, but you can replace this class with one of your own to process additional business logic to and possibly modify how the payment is scheduled, or cancel it completely. For example, you could use this to copy selected fields from an index table into a payment table, or to deny a recurring payment.

For information about implementing this class, contact Oracle Professional Services. The default value is: *com.edocs.payment.tasks.recur_payment.RecurringPaymentPlugIn*, which takes no action.

When to synchronize recurring payment with eStatement - By default, ePayment Manager uses the latest available bill when submitting the payment to the payment gateway. You can configure each payment gateway to only synchronize once, which reduces processing. The setting "whenever job runs" can be changed to "Only after the current bill is scheduled", which causes ePayment Manager to synchronize only once; when the bill is scheduled.

Skip Synchronization: N (default) enables synchronization. If you wish to ignore synchronization and start scheduling immediately, then change this to **Y**.

Task 2: RecurPaymentSchedulerTask

Number of days before pay date to schedule the payment – The check payment will be scheduled *N* days before the pay date by pmtRecurringPayment. *N* days before the due date, email notification will be sent, if Send email notification when the payment is scheduled is set to "Y". That gives the customer *N* days (less one, the day it is paid) to modify or cancel the scheduled payment.

CAUTION: Modifying this field may require modifications to the JSP that checks for valid entries when a customer schedules a check.

Implementation of interface IRecurringPaymentPlugIn - This is the name of the java class that is called before the pmtRecurringPayment job schedules a payment. It currently does nothing, but you can replace this class with one of your own to process additional business logic to and possibly modify how the payment is scheduled, or cancel it completely. For example, you could use this to copy selected fields from an index table into a payment table, or to deny a recurring payment.

For information about implementing this class, contact Oracle Professional Services. The default value is: *com.edocs.payment.tasks.recur_payment.IRecurringPaymentPlugIn*, which takes no action.

Send email when payment is scheduled? - Determines whether email notification is active for recurring payments.

Send email if scheduled payment amount is zero? - Determines whether email notification occurs when the scheduled payment amount is zero.

Send email when recurring payment is expired? - Determines whether email notification is sent when a recurring payment effective period has ended.

Cancel recurring payment if payment account is canceled? - Specify whether the recurring payment should be canceled if the account has been cancelled. This will be considered recurring payment expiration, so an email will be sent to the user.

Normally, this parameter will be "Y". Use "N" to have the pmtCheckSubmit job handle this condition, or if the plugin is going to take actions based on this condition.

Cancel recurring payment if payment account is invalid? - The account information (contained in a prenote for ACH or in a payment for CheckFree CDP) sent to the ODFI by the pmtConfirmEnroll job was returned to ePayment Manager as having incorrect account information. The user is enrolled, but the account is not valid.

If a credit card account was used for enrollment, the account information is not checked until a payment is made. If a credit card payment is sent to the payment processor with invalid account information, the account will be marked invalid.

Since the customer's enrollment failed, they will be sent an email when the pmtNotifyEnroll job runs. The customer must resubmit the information for that account, which must be verified before this account can be used to make a payment.

Send email when recurring payment is canceled? - Specify whether to send email to notify the user that their recurring payment was cancelled.

pmtRecurringPayment Operation

pmtRecurringPayment examines all the customer accounts to see which recurring payments need to be scheduled. It looks for recurring payments where the amount due is greater than zero, and the date to be scheduled is equal to or greater than *n* days before the current date, where *n* is configured on Number of days before pay date to schedule the payment .

If the number of payments specified in the **effective period** on the customer interface has been met, this job sets the recurring payment to "inactive".

If the customer selects a payment to be made based on the number of days before the due date, or selects the amount to be based on the due amount, pmtRecurringPayment must query eStatement to determine when to schedule the payment and/or how much to pay. For that reason, the pmtRecurringPayment job should be run after the eStatement Indexer job runs.

If the customer selects a payment for a fixed amount on a fixed date, then the pmtRecurringPayment job does not need to query eStatement to schedule the payment.

For additional information about how recurring payments work and the different types, see Chapter 6, Recurring Payments on page 39.

pmtSubmitEnroll Job

ACH optionally accepts enrollment information to verify the customer's check routing number and check account number. ACH calls this enrollment information a **prenote** which is the same as a regular check payment, except its dollar amount is zero. The name of the generated ACH file is *ppd_yyyyMMddHHmmssSSS.ach*.

pmtSubmitEnroll submits enrollment information to a payment gateway (for ACH only). It finds all accounts in the *payment_accounts* table whose *account_status* field is "pnd_active" and writes them into an ACH file. The *txn_date* field is set to the current date, and *account_status* field is changed to "pnd_wait".

pmtSubmitEnroll Configuration

The configurable parameters for this job are:

Skip Holidays - Determines whether to send the ACH payment batch file to the ACH payment gateway even when the bank is closed because of a holiday. The default is "N", which means send the file even if it is a holiday. The federal holidays are listed on page 65.

pmtPaymentRefund Job

A CSR user or an administrator can authorize refunds and the pmtPaymentRefund job processes the refund payment.

There are three categories of refunds:

- **Cancellations** – A payment can be cancelled only when a payment has a status of Scheduled. If a payment is in another state, the payment cannot be cancelled.
- **Voids** – If a payment has a status of Authorized, then it can be voided. An authorized payment should not be cancelled without performing an authorization reversal for the authorized amounts. Currently this applies only to credit cards and not for checks. The authorization reversal procedure is determined through the cassette implementation.
- **Refunds** – A payment can be refunded if it has Settled/Paid status. It is necessary to create a new payment transaction to make the refund.

Refund XML Format and Configuration

Process the batch refunds XML in the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--  The root element can be either doc or refunds both elements are
supported. doc is supported for the ddf conversion.   -->
<doc><!-- refunds -->
   <refund>
      <payeeId>BCBS</payeeId>
      <userId>John</userId>
      <paymentAccountNumber>4317345689007461</paymentAccountNumber>
      <payerAccoutNumber>ACC1234567890</payerAccoutNumber>
      <amount>200.00</amount>
      <paymentType>ccard</paymentType>
      <refundType>R</refundType><!-- R for Refunds D for Deposits -->
      <billingSystemTransactionNumber></billingSystemTransactionNumber>
      <originalTransactionNumber>12132312</originalTransactionNumber>
      <sourceSystemPaymentInitiator>CSR2</sourceSystemPaymentInitiator>
      <transactionDescription>Refund Desc</transactionDescription>
      <notifyRequired>Y</notifyRequired>
      <flexFieldOne>xxxxxx</flexFieldOne>
      <flexFieldTwo>xxxxxx</flexFieldTwo>
      <flexFieldThree>xxxxxx</flexFieldThree>
      <flexFieldDate>mm-dd-yyyy</flexFieldDate>
   </refund>
</doc><!-- refunds-->
```

Field Name	**Field Description**
Payee Id	DDN Name of the Biller. (Optional)
User Id	The user’s registration ID used to identify the user. (Optional field for Refunds.)
Payment Account Number	The user’s credit card or checking account number where the refund should be made. (Optional field for a refund.)
Payer Account Number	The account number that the user is paying for. (Mandatory field for refunds.)
Amount	The amount to be refunded. (Mandatory field.)
Payment Type	The refund payment mode Check or Ccard.
Refund Type	Refund type: R–Refund
Billing System Transaction Number	The transaction number provided by the system (the payment ID). Optional field, but this can reduce the effort of finding the correct payment for the refund.
Original Transaction Number	The transaction number provided to ePayment Manager by the authorizing gateway or clearing house. Optional field, but populating this field helps the gateway to process the refund more efficiently.
Source System Payment Initiator	The ID of the refund payment initiator. This is used for security.
Transaction Description	The description on why a refund has to be made.
Notify Required	Whether a notification is required for the refund. Notification can be sent to the biller as well as the payer.

Flex Field One	Flexible field one - Can be used by PS to store custom data.
Flex Field Two	Flexible field two - Can be used by PS to store custom data.
Flex Field Three	Flexible field three - Can be used by PS to store custom data.
Flex Flied Date	Flexible field four - Can be used by PS to store custom data.

LoadsRefundsTask

The primary purpose of the LoadsRefundsTask is to obtain the refund records through the IRefundDepot interface and schedule the eligible refunds for processing by the refund submit task.

The default implementation uses an XML file to retrieve the refund records. Specify the implementation class in the job configuration.

If the refund type is R (Refund) then the task will check whether the mandatory fields are set i.e. payer account number and amount.

To restrict querying of all the payment transactions (to balance the load) a job parameter is set so that querying is done for payments made for the last 'n' number of days/months. This configuration depends on the Billers business logic.

The plug-in validates the refund according to the Biller's business logic and updates the refund "Refund for Payment ID" and "Payment Type" fields with one of the payments "payment ID" and its payment type field with the selected payment transaction for the refund.

The plug-in then accepts or rejects the refund. If the plug-in accepts the refund, the updated refund record is written to the database with the status as scheduled. If the plug-in rejects the refund, the refund record is written to the database with the status as Rejected. If the mandatory parameters are not set then the refund is rejected and an entry is written to the payment_refunds table with the status Rejected.

LoadsRefundsTask Configuration

The configurable parameters for this job are:

Payment selection criteria for refunds – Select the unit of time (days, weeks, or months) to use for payment refund criteria.

Selection criteria value (Greater than 0 value) – Enter a specific number (of days, weeks, or months specified in previous field) to use for payment refund criteria.

Refund batch file location directory – The Input directory of the XML file.

Refund batch file name – The XML file name.

Refund batch file archive location directory – The Output Directory of the XML file.

Implementation of interface IRefundDepot – The implementation class of the IRefundDepot.

Implementation of interface ILoadRefundPlugin – The implementation class of the ILoadRefundPlugIn.

Skip Refund Loader Task – Default is **N**-No. If there are no batch refunds for the Biller, set this parameter to **Y**-Yes to skip the task.

SubmitRefundTask

SubmitRefundTask submits all the scheduled refunds for payment. This task accesses the payment_refunds table and queries all the scheduled refunds and processes them one at a time.

Cancellations - For the SubmitRefundTask cancellations are not scheduled; all cancellations happen online. The system cancels the payment in real time and inserts a refund record into the refund table as "Processed" and sets the refund type to "Cancellation." The pmtPaymentRefund job never gets the opportunity to process any cancellations because a cancellation is done on payments that have been scheduled and it is cancelled immediately.

If a scheduled payment is cancelled, the system processes it and if the status of the payment is "Scheduled," it cancels it immediately. The refund status changes to "Processed." A refund type of "Cancellation" is never scheduled, so the status changes to "Processed" since there is no gateway interaction. If the payment is in another state it is processed as a void or a refund.

Voids - Voids are special type of cancellation. Voids are processed online and inserted as a void to the payment_refunds table. SubmitRefundTask uses the void records, "refund for payment ID" and "payment type" fields to obtain the original payment from the payment tables. The task checks whether the obtained payments status is in the Authorized state.

If it is in the authorized state, the task performs the final validation according to the customization and accepts or rejects the new refund payment for the void. If the plug-in rejects the new payment, the refund status updates to rejected, and is stored in the database.

If the plug-in accepts the refund and the payment id of the new payment is captured through the refund, the old payment which is being voided is cancelled and updated in the database and the new payment is scheduled for processing. The status of the refund will be updated to processed and stored in the database.

If the original payment is not in the "Authorized" state and the status is "Processed" or "Settled," the system processes it as a refund.

Voids are applicable to only credit cards.

Refunds - All refunds with type as "Refund" are processed in this category. The task checks the status of the original payment, and it must be either "Processed" or "Settled."

If the status is "Processed" it implies that the payment has been sent to the gateway and a response is not yet received regarding the status. In this case, the refund process for this record is temporally halted and move to the next record.

If the status of the original payment is "Settled," then the system creates a new payment transaction for the specified refund with a negative amount. The refund will only be made to the original payment account if and only if there is no payment account number specified in the refunds. If the payment account number is specified in the refund along with the payment type, then the refund is made to that account number.

If the plug-in accepts the new payment, then it schedules the new payment transaction in the payment tables and updates the status of the refund to "Processed" and records the new payment ID in the refund using the “refund payment id” field. If the plug-in rejects the payment, the system changes the refund status to “Rejected.”

Note: Verisign gateway does not support refunds because it is not allowed to make negative payments.

SubmitRefundTask Configuration

The configurable parameters for this job are:

Implementation of Interface ISubmitRefundPlugin - The implementation class of the ISubmitRefundPlugin

Allow multiple refunds per payment - **Y**-Yes, or **N**-No. The option to allow multiple refunds for a payment.

External Payment Jobs

Create and configure the following jobs if you enable payment to external billers.

pmtCheckSubmit Job

pmtCheckSubmit selects scheduled check payments that are ready to pay that DDN matches the job's DDN or (optionally) one of the DDNs listed in the Submit Checks for Multiple DDNs field. Checks that are ready to pay are those whose pay dates are scheduled for tomorrow or sooner. It then generates a batch file in the output directory. The output directory is defined in the configuration settings for the payment gateway whose DDN matches the Application.

pmtCheckSubmit uses the check's *pid* to get the latest check account information from the enrollment database, and then uses that to submit the check payment. If the *pid* is null, the check's account information is used for check submission.

A check account may be deactivated, cancelled or physically deleted from the database at the time the scheduled check is submitted. For example, if the check account is deleted, the check will be cancelled instead of submitted. If the check is deactivated to the "pnd_active" or "bad_active" state or is cancelled, you can configure this job to decide whether to cancel the payment or submit it.

A zero dollar amount check (a prenote) won’t be submitted, and this job changes the check's status to "processed".

For ACH, you can put checks from other DDNs into the same ACH file, but each DDN must be in a separate batch. The DDNs must have save immediate origination, immediate destination, immediate origination name, and immediate destination name.

The file naming convention for an ACH file is *ppd_yyyyMMddHHmmssSSS.ach*.

The file naming convention for a CheckFree file is *test.ftp_user_name.debit.dat.CCCCMMDDHHMMSS.pgp*. The configuration setting *test* will be replaced by *prod* after the check has gone into production.

The format of the ACH file can be modified by editing the XML files in *<PAYMENT_HOME>/lib/payment_resources/ach/template/* (*<PAYMENT_HOME>\lib\payment_resources\ach\template\*). See Oracle Professional Services for help modifying ACH batch format.

After a check is submitted, its status in the database changes from "scheduled" to "processed". If an error occurs during the check submission process, the status of the check changes to "failed".

The following fields are updated in the *check_payments* table after pmtCheckSubmit runs:

Column updated	**Value**
last_modify_time	current time
status	7
action_code	For ACH: 27 for checking and 37 for saving. For Checkfree: ADD.
txn_number	For ACH: trace number.
reminded	“N”
log_id	ID of the summary report in the *payment_log* table.
gateway_payment_id	Populated only if you are using the gateway payment ID to match a check returned from ACH to a check in the ePayment database. For more information, see the *Customizing and Extending Oracle Siebel ePayment Manager* guide or contact Oracle Professional Services.

Scheduling and Holidays

By default, ePayment Manager allows a check payment to be scheduled on a bank holiday. The following rules explain when a Federal holiday qualifies as a bank holiday.

If the holiday is on:

- A weekday, then it is a bank holiday.
- Sunday, then the following Monday is a bank holiday.
- Saturday, then even if the previous Friday is a Federal holiday, it is not a bank holiday.

ACH is closed on bank holidays, but it is okay to send a file to ACH which requires transfer of money on holidays: ACH simply processes the checks on the next available bank business day. However, some banks require ACH files to skip bank holidays. By default, ePayment Manager skips bank holidays.

When an ACH file is generated, all checks with the same pay dates are put into the same batch, and the batch entry effective date is set. That date is the suggested date for ACH to process the checks in that batch. The following rules determine how the batch entry effective date is set:

- If the pay date is today or earlier, then the batch entry effective date is set to tomorrow. If not, it is set to the pay date.

- If the batch entry effective date is on a bank holiday, then it is moved to the next availed bank business date.

If you don't want to skip holidays when calculating the batch entry effective date, modify the Payment Settings.

Configuration

The configurable parameters for this job are:

Number of days before a check's pay date for it to be submitted - When a check payment is scheduled, a date must be specified when the check is going to be cleared. By default, a check payment will not be submitted to a payment gateway until one day before the scheduled pay date. The submission date can be changed by specifying a different value.

CAUTION: Modifying this field may require modifications to the JSP that checks for valid entries when a customer schedules a check.

For example, if the value is one and the job runs today, all the checks whose pay dates are tomorrow or earlier will be selected to send to the ACH payment gateway.

CAUTION: Payments made after this job runs will not necessarily be paid on the same day. We recommend running this job at 11:59PM to ensure that payments will be processed on the same day as they were made. If the job runs early in the morning each day (e.g. 2AM), then the job will not process payments scheduled during normal business hours on the same day, since it already ran that day.

Cancel payment if check account is canceled? - Specifies whether the scheduled payment should be canceled if the check account has been cancelled. Normally, this is "Y". Use "N" if the plugin is going to take actions based on this condition.

Cancel payment if account information is invalid? - The account information (contained in a prenote for ACH or in a payment for CheckFree CDP) sent to the ODFI by the pmtConfirmEnroll job was returned to ePayment Manager as having incorrect account information. The user is enrolled, but the account is not valid.

Since the customer's enrollment failed, they will be sent an email when the pmtNotifyEnroll job runs. The customer must resubmit the information for that account, which must be verified before this account can be used to make a payment.

Submit payment if check account is pending? - When the customer adds a new checking account, it is in a pending state until the period specified by Days to Activate Pending Subscribers has expired. "Y" means submit the payment even if the account is pending. "N" means do not submit the payment when the account is pending.

Submit checks for additional DDNs - List any additional DDNs of checks that this job will submit to the payment gateway for processing, separated by semicolons.

Skip Holidays - Determines whether to send the ACH payment batch file to the ACH payment gateway even when the bank is closed because of a holiday. The default is "N", which means send the file even if it is a holiday. The federal holidays are listed on page 65.

pmtCreditCardSubmit Job

This job selects credit card payments that are scheduled to be paid within a configurable number of days before today, and opens a connection to a credit card payment gateway to authorize and settle those transactions. Both authorization and payment are done at the same time.

pmtCreditCardSubmit Configuration

The configurable parameters for this job are:

Number of Days Before a Credit Card's Pay Date for it to be Submitted - When a credit card payment is scheduled, a date must be specified when the credit card payment should be settled. By default, a credit card payment will not be submitted to a payment gateway until one day before the scheduled pay date. The submission date can be changed by specifying a different value.

CAUTION: Modifying this field may require modifications to the JSP that checks for valid entries when a customer schedules a payment.

Cancel Payment if Credit Card Account has Expired - "Y" means if the credit card account used in a payment has expired, then cancel the payment. "N" means try to make the payment with the old account, failure of which causes email to be sent to the customer, if configured by the pmtPaymentReminder job.

pmtCreditCardSubmit Operation

pmtCreditCardSubmit submits credit cards to a credit card gateway to be processed. It searches the *creditcard_payments* table to find all scheduled credit card payments whose *status* field is "scheduled" (6) and whose *pay_date* field has a date the same as or prior to one day after the day the job is running (by default), and sends them the credit card gateway for processing.

Credit card account information is saved by ePayment Manager as part of the payment when the payment is scheduled. Whether this copy of the account information is used for submission depends on the contents of the *pid* field.

- When *pid* is **not null**, the saved account information is used to extract the latest credit card account information from the enrollment database, and the extracted account information is used for submission. This eliminates any potential problems related to changing or deleting a credit card account after the payment is scheduled. The Cancel Payment if Credit Card Account has Expired parameter determines which action to take with scheduled payments when the credit card account is changed.
- When the *pid* is **null**, the saved copy of the credit card account is used. This is useful when the pid cannot be found in enrollment database. For example, when a customer database offers payment account information but doesn't have a unique *pid*.

If pmtCreditCardSubmit is successful submitting the credit card payment, the payment is approved, money is guaranteed to be transferred, and the status of the payment is set to "settled" (8). If there is a problem submitting the payment, its status is set to "failed-authorize" (-4).

ePayment Manager supports the Verisign credit card cartridge and uses HTTP to communicate with it. If there is a network problem, the *status* of the payment stays "scheduled", but the payment's *txn_err_msg* field gets the error message. This ensures that the payment will be picked up by the next run of the pmtCreditCardSubmit job. If the payment is successful, ePayment Manager stores the confirmation number from Verisign in the *txn_number* field of the *creditcard_payments* table.

Verisign Threads

To speed up credit card processing, you can configure the Verisign cartridge to use simultaneous connections (threads) with Verisign. By default, the Number of Threads field in the Payment Settings is 1, but you can enter a larger number to speed processing.

However, there is a bug with the Verisign SDK, which causes a connection failure when the number of threads is too high. This problem is especially noticeable NT, but also occurs on UNIX. On NT, the number of threads should not exceed 10. Connection failures can be significantly reduced by using multiple copies of the Verisign certificates. By default, there is only one certificate.

Connection failures caused by the Verisign bug are not fatal. ePayment Manager recognizes the failure and keeps the payment's status as scheduled, so that the failed payments will be processed the next time pmtCreditCardSubmit runs. If you increase the number of threads and find there are failures, schedule your job run twice, back to back.

The following columns are updated after a credit card is submitted:

Column updated	**Value**
last_modify_time	Current submit time.
reminded	Set to “N” if the status is "settled" or "failed-authorize".
status	Set to "settled" if the payment is accepted by card issuer. Set to "failed-authorize" if the payment is rejected or returned as “referral” by card issuer. Stays "scheduled" if there is a network error.
log_id	Contains the value in *payment_log*, which represents a report ID.

10 ePayment Manager Reports

Overview

ePayment Manager keeps payment history for auditing purposes. Checks go through a list of states before it is cleared. For insert, update/insert and update operations, ePayment Manager keeps a copy of the in the *check_payments_history* table. ePayment Manager records when the check was created, when the check was updated or cancelled, when the check is processed, and other check actions.

ePayment Manager also logs additional important information: warnings and errors. Based on the format of the logging messages, ePayment Manager can hook up with other monitor systems, such as EMC patrol.

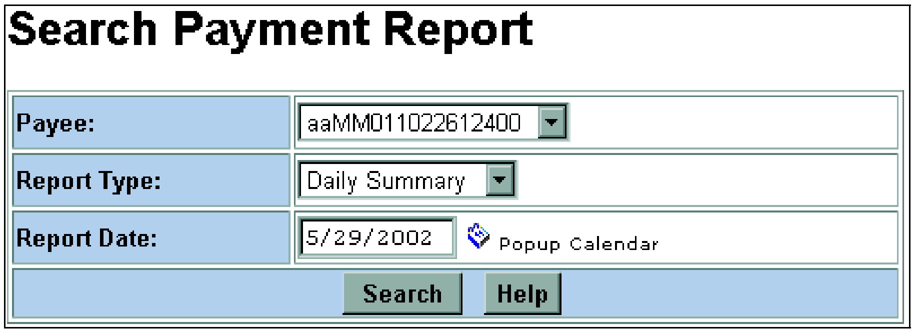
Viewing ePayment Manager Reports

After a payment job runs, you can view payment reports for entries from that job based on two report types:

- Daily Summary
- Daily Exceptions

To view ePayment Manager reports:

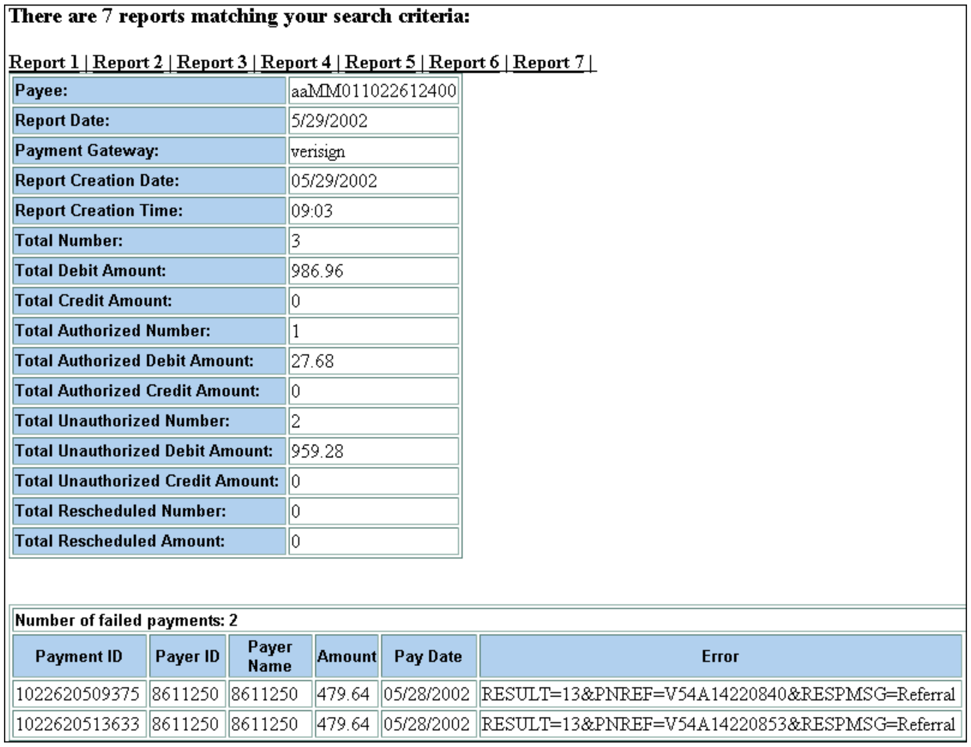
1. In the Command Center, click **Reporting**. The Reporting and Log page appears.
2. Select the **Payment Reports** tab. The Search Payment Report page appears.
3. Select a payee from the drop-down menu and select a report type.
4. Enter a date on which you want to perform the search. You can use the Popup Calendar to help you determine the date.



Search Payment Report

Payee:	aaMM011022612400
Report Type:	Daily Summary
Report Date:	5/29/2002

5. Click **Search**. Depending on the type of payment report selected (Daily Summary or Daily Exception), you see a report similar to the following:



There are 7 reports matching your search criteria:

Report 1 | Report 2 | Report 3 | Report 4 | Report 5 | Report 6 | Report 7 |

Payee:	aaMM011022612400
Report Date:	5/29/2002
Payment Gateway:	verisign
Report Creation Date:	05/29/2002
Report Creation Time:	09:03
Total Number:	3
Total Debit Amount:	986.96
Total Credit Amount:	0
Total Authorized Number:	1
Total Authorized Debit Amount:	27.68
Total Authorized Credit Amount:	0
Total Unauthorized Number:	2
Total Unauthorized Debit Amount:	959.28
Total Unauthorized Credit Amount:	0
Total Rescheduled Number:	0
Total Rescheduled Amount:	0

Number of failed payments: 2

Payment ID	Payer ID	Payer Name	Amount	Pay Date	Error
1022620509375	8611250	8611250	479.64	05/28/2002	RESULT=13&PNREF=V54A14220840&RESPMSG=Referral
1022620513633	8611250	8611250	479.64	05/28/2002	RESULT=13&PNREF=V54A14220853&RESPMSG=Referral

Credit Card Gateways

Credit card gateways will only report credit card transactions.

Viewing ePayment Manager Error Logs

ePayment Manager jobs may generate error messages. ePayment Manager writes all errors to the payment database table, and appends them with the *.log* suffix. Error logs can be viewed through the Reporting menu option in the Command Center.

The following table lists ePayment Manager error messages.

Error Message	Description
NoPaymentAccountException	Indicates an ACH customer cannot be activated (the ACH payment gateway only supports up to three payment accounts for each customer) and can result in the failure of the payment application. This type of error typically indicates a problem with the Enrollment JSP page and should be discovered during initial system setup and testing.
FileIOException	Indicates there is a problem regarding the reading and writing of ACH files. Verify the existence of the ACH file output directory and file input directory, and make sure the permissions on them are correct.

Error Message	**Description**
CassetteException	Indicates there is a general problem regarding an ACH operation and the interpretation of the error depends on the specific error messages themselves. This error message is sometimes used as a wrapper for other errors.
AchFormatException	Indicates there is a problem with the ACH file format. For example, one common source of this error is that the length of supplied data is greater than the length allowed by ACH.
TemplateException	Indicates one or more ACH template files failed during processing by the Template engine. In some cases, this error can be corrected during system setup and configuration. Also check the ACH template formats to make sure they are valid.
SQLException	Indicates a generic exception accessing the ePayment Manager database. This error message can be generated by a variety of errors, but one common source is the failure to establish connection pools in the ePayment database. In this case, check whether the database server is still running, and check the application server configuration.
OperationNotSupported Exception	Indicates the current payment operation is not supported by ACH. This error message typically indicates faulty coding and should be corrected by a developer immediately.
RemoteException	Indicates a generic exception and typically serves as the wrapper for other exceptions. This error message typically indicates a failed payment operation, which cannot be corrected by running the operation again.
DuplicateKeyException	Indicates an exception that occurs when there are two check payments in the database with the same payment ID. ePayment Manager uses a timestamp (in milliseconds) to assign payment IDs. This type of error typically does not require manual intervention. Instead, the customer will receive an error message and then be prompted to submit the check payment again.
NoDataFoundException	Indicates there is no data being extracted from the ePayment Manager database. This type of error is typically corrected by the application during processing.

11 ePayment Manager Administration

ePayment Manager Database

Preventing Multiple Payments

By default, ePayment Manager allows a bill to be paid more than once. To ensure that a bill can only be paid once, you need to add a unique key constraint on the *bill_id* field of the *check_payments* table. Run *<PAYMENT_HOME>/db/<DB_NAME>/set_unique_bill_id.sql* to set the unique constraint. The *bill_id* in ePayment Manager is the same as the doc ID in eStatement.

If a customer tries to pay a bill that has already been paid after the unique key constraint has been added (either from the UI or by a previously scheduled recurring payment), the customer will receive an error message stating that the bill has been already paid. If the bill is paid from the UI and a recurring payment tries to pay it again, the payment will fail and an email notification message will be sent to the customer (if recurring payments are configured for that email notification).

Adding this constraint won't prevent a customer from making a payment using a bill ID. For example, a customer can still make a payment directly from the **Make Check Payment** link, which allows them to make a payment without specifying a bill.

The unique key constraint only informs a customer that the bill has been paid when they try to pay a bill that has already been paid. If you want to provide additional features, such as disabling the payment button when the bill has already been paid, you must query the database to get that information. Use caution when adding extra functions because performing additional database queries can deteriorate ePayment Manager performance. Make sure to create the proper index if you plan to create a new query.

UI Actions and Database Changes

The following table lists user actions available in the sample interface and the ePayment Manager Command Center jobs, and describes their impact on the ePayment Manager database. This example uses the enrollment for ACH where ePayment Manager keeps payment information in a separate database:

UI Action	ePayment Manager Database
Create payment setting (Command Center)	Payment setting information is saved in the *payment_profile* table. The param_name *user_account_accessor* should point to the right *IUserAccountAccessor* implementation and *payment_account_accessor* should point to the right *IPaymentAccountAccessor* implementation.

UI Action	**ePayment Manager Database**
User enrolls	User information is inserted into CDA and payment accounts are inserted into the *payment_accounts* table.
Run pmtSubmitEnroll	Finds all payment accounts whose *account_status* is "pnd_active", *txn_date* is "null", and sends to the ACH payment gateway. After that, it sets *txn_date* to the current date (yyyyMMddHHmm).
Run pmtConfirmEnroll	Changes *account_status* to "bad_active" if returned or to "active" if there is no return after three days. Updates *notify_status* to "N".
Run pmtNotifyEnroll	Finds all payment accounts whose *notify_status* is "N", and send emails. Updates *notify_status* to "Y".
User logs in	The user’s ID and password is checked against the *user_id* and hash in the enrollment table.
User makes a check payment	The payment is saved into the *check_payments* table. The status of the check is "scheduled"(6).
User clicks on **`Future Payments`**	Displays a list of scheduled payments for this user. The user can cancel or update scheduled payments from here.
User cancels or updates a check	When a user cancels a check, the status of that check is set to "canceled"(9). The check is not deleted from the database. When a user updates a check, the same entry in *check_payments* will be updated. A check can only be cancelled or updated when the status of that check is "scheduled"(6).
Run pmtCheckSubmit	Finds all checks due by tomorrow (or before), and sends them to the ACH payment gateway. The status of the check changed to "processed"(7). *txn_number* now holds the trace number of the check, and the *reminded* field is set to "N". A batch report file is written to the *payment_log* table, whose type is *summary*. You can view this report from the Command Center.
User clicks on **`Payment History`**	Processed check payments will show here. Paid, returned, failed and cancelled checks will also show here.
Run pmtCheckUpdate	Changes the check status to "returned" (-4) if there is a return for it, or to "paid" if there is no return after five business days and *reminded* field is set to **`N`**. If the check is returned, then the *txn_error_msg* field is set to the ACH return code, and an exception report is generated into *payment_log* for each return file.
User creates a new payment reminder	The payment reminder is saved into the *payment_reminders* table. The *next_reminder_date* will be set to "start_date".
Run pmtPaymentReminder	1) For regular payment reminders: It finds all the entries in *payment_reminders* whose *next_reminder_date* is before than or equal to today, and sends an email for each email address. After the email is sent, the *next_reminder_date* is updated based on the *remind_interval*. 2) For check payment reminders: It finds all the checks in *check_payments* whose status is one of 7, 8, -1, -4 and whose *reminded* field is N. After the email is sent, the *reminded* field is changed to Y.

Table Sizing

The size to which the tables in the eStatement/ePayment database grow depends on the number of enrolled users.

The following tables describe the ePayment Manager tables, and the eStatement Manager tables that are related to enrollment. Most statements apply to both Oracle and Microsoft SQL Server; differences are noted. The figures in the tables assume that there are 100,000 registered users.

ePayment Manager

ePayment Table	Projected Row Count	Notes
check_payments	1.2 million based on 100K users kept for one year	Customer dependent. Assuming one user makes one check payment per month, and check history is kept in the database for one year, then the total number of rows is approximately 12 times the number of users.
check_payments_ history	Approximately 3 times the size of check_ payments table	Tracks the state of a check payment. Usually a check passes through three states before it's cleared or returned.
check_payments_ status	10	This table is a reference to the meanings of check payment status. It is not used by ePayment Manager.
credit_card_ payments	1.2 million based on 100K users per year	Customer dependent. Assuming one user makes one credit card payment per month, and credit card payments are kept in the database for one year, then the total number of rows will be approximately 12 times the number of users.
payment_accounts	200,000	The estimated row size is approximately 1.4K per user, or 280MB for 100K users.
payment_bill_ summaries	Approximately 33K * 12 = 400K, assuming1/3 of the register with recurring payment	Saves bill information related to recurring payments.
payment_invoices	12 million based on each payment averaging 10 invoices	This is usually used for business customers. Some billers may choose not to use this feature.
payment_log	< 10k per year	Approximately 20 rows will be inserted when a pmtCheckSubmit batch job is run.
payment_profile	< 100	There are approximately 20 rows for each DDN and payment type.
payment_reminders	< 100K based on 100K users	Customer dependent. Each customer can set up one reminder (each reminder creates one row in the table), but not all customers will use reminders.

ePayment Table	Projected Row Count	Notes
recurring_payments	Approximately 33k, assuming 1/3 of the register with recurring payment	If recurring payment is turned off, then this table will be empty.

Table Maintenance

For check payments, there are two tables which may grow quickly: *check_payments* and *check_payments_history*.

The *check_payments* table records the check payments made by users. The *check_payments_history table* records the history (status changes) for each check in *check_payments*. The *check_payments_history table* is approximately three times the size of the *check_payments* table.

Payments that are of a certain age (for example, one year) can be backed up and deleted from the ePayment Manager database. This will ensure proper performance for ePayment Manager for high volume of users. The *create_time* field in the *check_payments* table records when a check is created, and can be used to determine a check's age.

CAUTION: Be careful when deciding how long to keep a check in the ePayment Manager database before it is removed. If you expect the number of users to be low and the database size is an acceptable size, there is no need to downsize the tables.

Backup and Recovery

All eStatement/ePayment database transactions operate in their own transaction context. If a single operation fails (for example, failure to enroll or submit a payment), the eStatement/ePayment database will be automatically rolled back to its original state.

To recover all transactions for a certain period, the database administrator should back up the database regularly so that the database can be restored to the previous day. It is best to back up the database before running the Payment Submit and Update jobs, so there will be no question about whether the jobs were still running during the backup. The frequency of backup depends on how long the period is for payment processing.

What to back up

All tables should be backed up, but the *check_payments*, *check_payments_history*, *creditcard_payments* and *creditcard_payments_history* tables in particular should be backed up on a regular basis.

Stored procedures should be backed up, especially procedures modified by Oracle Professional Services.

Do not backup the master database. The master database is used by the database itself for internal purposes.

Schema

The eStatement/ePayment database schema is defined in the following files:

For UNIX:

```
<PAYMENT_HOME>/db/<DB_NAME>/create_payment_schema.sql
<PAYMENT_HOME>/db/<DB_NAME>/alter_payment_schema.sql
```

For Microsoft Windows:

```
<PAYMENT_HOME>\db\<DB_NAME>\create_payment_schema.sql
<PAYMENT_HOME>\db\<DB_NAME>\alter_payment_schema.sql
```

Table Column Definitions

ePayment Manager Tables

CHECK_PAYMENTS

This table saves check payment information.

Name	Null?	Type	Description
PAYMENT_ID	NOT NULL	NUMBER(28)	Unique for each check. It's time stamp value.
LAST_MODIFY_TIME	NOT NULL	DATE	The last time this check was updated.
CREATE_TIME	NOT NULL	DATE	The time when this check is created.
NEEDS_BACKUP	NOT NULL	CHAR(1)	Not used.
BILL_ID	NULL	VARCHAR2(255)	The ID of the bill paid by this check.
PAYEE_ID	NOT NULL	NUMBER(10)	DDN reference number.
PAYER_ID	NOT NULL	VARCHAR2(40)	User/Login ID.
PAY_DATE	NOT NULL	DATE	Check's pay date (the date the user wishes the check to be cleared).
STATUS	NOT NULL	NUMBER(2)	Check's status: scheduled(6), processed(7), paid(8), returned(-4) or failed(-1).
ROUTING_TRANSIT	NOT NULL	VARCHAR2(9)	Check's routing transit number.
ACCOUNT_NAME	NOT NULL	VARCHAR2(40)	Check account name.
CHECK_ACCOUNT_ NUMBER	NOT NULL	VARCHAR2(255)	Check account number.
AMOUNT	NOT NULL	NUMBER(28,2)	Check amount.

Name	**Null?**	**Type**	**Description**
TXN_TIMESTAMP_1	NULL	DATE	Flexible date field. Can be used for customization.
TXN_TIMESTAMP_2	NULL	DATE	Flexible date field. Can be used for customization.
PARTITION_ID	NOT NULL	NUMBER(10)	This number is used to partition the table into fixed-size buckets for performance tuning. It is configured in Payment Settings.
TXN_STATUS	NULL	VARCHAR2(20)	The transaction status returned from the payment gateway. May not be available for some gateways.
TXN_FEE	NULL	NUMBER(28,2)	The transaction fee charged by payment gateway. May not be always available.
REMINDED	NOT NULL	CHAR(1)	"Y" or "N"; indicates whether an email notification has been sent for the current status.
TXN_ERR_MSG	NULL	VARCHAR2(255)	The transaction error message returned from payment gateway. For ACH, this is the ACH return error code.
TXN_NUMBER	NULL	VARCHAR2(40)	Transaction number sent to or assigned by the payment gateway. For ACH, this is the ACH trace number.
PAYER_ACCOUNT_ NUMBER	NULL	VARCHAR2(40)	User’s account number with the biller.
MEMO	NULL	VARCHAR2(255)	Check memo, which can be used for customization.
ACCOUNT_TYPE	NOT NULL	VARCHAR2(10)	Account type, payment gateway dependent. For ACH: either “checking” or “saving”. For Checkfree: “DDA”.
CHECK_USAGE	NULL	VARCHAR2(10)	“personal” or “business”.
CHECK_NUMBER	NULL	NUMBER(10)	Check number. Not used.
ACTION_CODE	NULL	VARCHAR2(20)	Payment gateway action code. For ACH: 27(checking debit) or 37(saving debit). For Checkfree: “ADD”
LOG_ID	NULL	NUMBER(28)	This login ID points to a log ID in the *payment_log* table. It associates the check with an ePayment Manager report.

Name	**Null?**	**Type**	**Description**
GATEWAY_ PAYMENT_ID	NOT NULL	VARCHAR2(255)	For Checkfree, this is the payment ID assigned by Checkfree. For ACH, it matches a returned check from the ACH file to the database, when it is not possible to populate a check payment ID into the ACH file.
TXN_START_DATE	NULL	DATE	For ACH, this is the effective batch entry date for the check.
TXN_END_DATE	NULL	DATE	Reserved.
PAYMENT_SOURCE	NOT NULL	CHAR(1)	R/S: “S” means paid from UI, “R” means paid from recurring payment
FLEXIBILE_FIELD_1	NULL	VARCHAR2(255)	Flexible field for customization.
FLEXIBILE_FIELD_2	NULL	VARCHAR2(255)	Flexible field for customization.
FLEXIBILE_FIELD_3	NULL	VARCHAR2(255)	Flexible field for customization.
PID	NULL	VARCHAR2(255)	The unique ID used to identify this payment account.
LINE_ITEM_ID	NULL	VARCHAR2(255)	Keeps track of data for line-item disputes.

CHECK_PAYMENTS_HISTORY

This table records the status changes that a check goes through. Whenever a check changes status, a new record is inserted into this table. A check usually goes through three statuses: "scheduled", processed" and then "paid". This means there are usually three records in this table for that check. Use this table to keep track of a check: when it is processed, when it gets paid, returned, cancelled, etc.

This table schema is exactly the same as the *check_payments* table, except that the *payment_id* is no longer a primary key.

CHECK_PAYMENTS_STATUS

This table describes legal check payment status values:

Name	**Null?**	**Type**	**Description**
STATUS	NOT NULL	NUMBER(2)	Numeric value of check status.
STRING_VALUE	NOT NULL	VARCHAR(20)	String value of check status.
DESCRIPTION	NOT NULL	VARCHAR2(255)	Description of each value.

CREDITCARD_PAYMENTS

This table contains credit card payment information. ePayment Manager does not save credit card numbers, so the payment gateway must have a real-time connection to a credit card processor, such as Verisign.

Name	Null?	Type	Description
PAYMENT_ID	NOT NULL	NUMBER(28)	Unique for each check. It's time stamp value.
LAST_MODIFY_TIME	NOT NULL	DATE	The last time this payment is updated
CREATE_TIME	NOT NULL	DATE	The time when this payment is created
BILL_ID	NULL	VARCHAR2(255)	The ID of the bill paid by this payment
PAYEE_ID	NOT NULL	NUMBER(10)	DDN reference number
PAYER_ID	NOT NULL	VARCHAR2(40)	User/Login ID
PAYER_ACCOUNT_ NUMBER	NULL	VARCHAR2(40)	User's account number with the biller.
CARD_HOLDER_ NAME	NULL	VARCHAR2(40)	Name on the card.
CARD_NUMBER	NULL	VARCHAR2(255)	The account number on the card. The contents depend on the payment settings.
CARD_TYPE	NULL	VARCHAR2(40)	Type of card. For example, VISA, MC, AMEX.
PID	NULL	VARCHAR2(255)	The unique ID used to identify this payment account.
CARD_EXPIRE_DATE	NULL	DATE	Card expiration date.
CARD_EXPIRE_ DATE_FORMAT	NULL	VARCHAR2(40)	Format of the expiration date.
CARD_STREET	NULL	VARCHAR2(255)	Street address of cardholder.
CARD_CITY	NULL	VARCHAR2(40)	City of cardholder.
CARD_STATE	NULL	VARCHAR2(40)	State of cardholder.
CARD_ZIP	NULL	VARCHAR2(40)	Zip code of cardholder.
CARD_COUNTRY	NULL	VARCHAR2(40)	Country of cardholder.
PAY_DATE	NOT NULL	DATE	Payment date.
AMOUNT	NOT NULL	NUMBER(28,2)	Payment amount.
STATUS	NOT NULL	NUMBER(2)	Payment status.

Name	**Null?**	**Type**	**Description**
TXN_STATUS	NULL	VARCHAR2(20)	The transaction status returned from payment gateway. May not be available for some gateways.
TXN_ERR_MSG	NULL	VARCHAR2(255)	The transaction error message returned from payment gateway. For ACH, this is the ACH return error code.
TXN_NUMBER	NULL	VARCHAR2(40)	Transaction number sent to or assigned by payment gateway. For ACH, this is the ACH trace number.
TXN_FEE	NULL	NUMBER(28,2)	The transaction fee charged by payment gateway. May not be always available.
TXN_AUTH_CODE	NULL	VARCHAR2(40)	Transaction authentication code.
TXN_AVS_CODE	NULL	VARCHAR2(40)	Address verification code
REMINDED	NOT NULL	CHAR(1)	Determines whether a user should be sent reminder email.
PARTITION_ID	NOT NULL	NUMBER(10)	This number partitions the table into fixed-size buckets for performance tuning. The size is configurable through Payment Settings.
LOG_ID	NULL	NUMBER(28)	ID of the summary report in the *payment_log* table.
TXN_START_DATE	NULL	DATE	Transaction start date.
TXN_END_DATE	NULL	DATE	Transaction end date.
PAYMENT_SOURCE	NULL	CHAR(1)	Describes which ePayment Manager function scheduled this payment. R" for recurring and "S" for single payment.
TXN_TIMESTAMP_1	NULL	DATE	For customization.
TXN_TIMESTAMP_2	NULL	DATE	For customization.
MEMO	NULL	VARCHAR2(255)	Check memo, which can be used for customization.
FLEXIBLE_FIELD_1	NULL	VARCHAR2(255)	Flexible field for customization.
FLEXIBILE_FIELD_2	NULL	VARCHAR2(255)	Flexible field for customization.
FLEXIBILE_FIELD_3	NULL	VARCHAR2(255)	Flexible field for customization.
LINE_ITEM_ID	NULL	VARCHAR2(255)	Keeps track of data for line-item disputes

CREDITCARD_PAYMENTS_HISTORY

This table records the status changes a credit card payment goes through. Whenever the payment status changes, a new record is inserted into this table. Use this table to keep track of a credit card payment: when it is processed, when it settled, returned, cancelled, etc.

This table schema is the same as the *creditcard_payments* table, except the *payment_id* is no longer a primary key.

CHECK_PAYMENTS_STATUS

This table describes the possible status values a check payment (stored in the *check_payments* table) can have.

Name	**Null?**	**Type**	**Description**
STATUS	NOT	NULL NUMBER(2)	Check status as a digit.
STRING_VALUE	NOT NULL	VARCHAR2(20)	Check status name.
DESCRIPTION	NOT NULL	VARCHAR2(255)	Description of check status.

CREDITCARD_PAYMENTS_STATUS

Describes the possible status that a credit card payment can have, which is stored in the *creditcard_payments* table.

Name	**Null?**	**Type**	**Description**
STATUS	NOT	NULL NUMBER(2)	Credit card status as a digit.
STRING_VALUE	NOT NULL	VARCHAR2(20)	Credit card status name.
DESCRIPTION	NOT NULL	VARCHAR2(255)	Description of credit card status.

PAYMENT_ACCOUNTS

This table saves information about all payment accounts.

Name	**Null?**	**Type**	**Description**
PID	NOT NULL	VARCHAR2(40)	Identifies this payment account.
USER_ID	NOT NULL	VARCHAR2(40)	The user who owns this payment account.
DDN	NULL	VARCHAR2(18)	The DDN name, used for ACH pre-note
PAYMENT_TYPE	NOT NULL	VARCHAR2(10)	Either “check” or “ccard”
ACCOUNT_HOLDER_ NAME	NOT NULL	VARCHAR2(40)	The customer’s name for the payment account.

Name	**Null?**	**Type**	**Description**
ACCOUNT_NUMBER	NOT NULL	VARCHAR2(255)	The customer’s payment account number.
ACCOUNT_TYPE	NOT NULL	VARCHAR2(40)	For check: “checking” or “saving”. For credit card: the card type, such as “visa”, "AMEX", etc.
ACCOUNT_USAGE	NULL	VARCHAR2(40)	“personal” or “business”.
ACCOUNT_STATUS	NULL	VARCHAR2(40)	Can be “active”, “inactive”, "bad_active", "pnd_active", "pnd_wait".
ROUTING_TRANSIT	NULL	VARCHAR2(9)	For check: the check routing number.
EXPIRATION_DATE_FORMAT	NULL	VARCHAR2(20)	The date format of the credit card account expiration date.
EXPIRATION_DATE	NULL	DATE	The date when the payment account expires.
STREET	NULL	VARCHAR2(255)	Billing address.
CITY	NULL	VARCHAR2(40)	Billing address.
STATE	NULL	VARCHAR2(40)	Billing address.
ZIPCODE	NULL	VARCHAR2(40)	Billing address.
COUNTRY	NULL	VARCHAR2(40)	Billing address.
NOTIFY_SOURCE	NULL	CHAR(1)	Used for ACH prenote notification.
NOTIFY_STATUS	NULL	CHAR(1)	For ACH prenote notification. Indicates whether the payment account has been notified.
TXN_MESSAGE	NULL	VARCHAR2(255)	Contains the error message for ACH prenote or NOC.
TXN_DATE	NULL	DATE	For ACH prenote. Date of the transaction happens.
FLEX_FIELD_1	NULL	VARCHAR2(255)	Used for customization.
FLEX_FIELD_2	NULL	VARCHAR2(255)	Used for customization.
FLEX_DATE_1	NULL	DATE	Used for customization.
NOTIFY_EXPIRE_STATUS	NOT NULL	CHAR(1)	For ACH prenote notification, indicates whether a notify expire message has been sent.

Name	Null?	Type	Description
GROUP_ID	NULL	VARCHAR2(255)	In a B2B application, a recurring payment set up by one person can be viewed/modified by another, depending on their authorization, and is managed outside ePayment Manager. ePayment Manager provides this flexible field for use in maintaining this relationship.
GROUP_ID_REF	NULL	VARCHAR2(255)	In a B2B application, a recurring payment set up by one person can be viewed/modified by another, depending on their authorization, and is managed outside ePayment Manager. ePayment Manager provides this flexible field for use in maintaining this relationship.
CSC_CODE	NULL	VARCHAR2(5)	Used to store the Card Security Code of the credit card.
ACCOUNT_NICK_NAME	NULL	VARCHAR2(40)	User-defined account nickname.
ACCOUNT_DESCRIPTION	NULL	VARCHAR2(255)	User-defined account description.

PAYMENT_BILL_SUMMARIES

This table saves all the bill summaries paid by recurring payments.

Name	Null?	Type	Description
BILL_ID	NOT NULL	VARCHAR2(255)	DOC ID of a bill.
PAYER_ID	NOT NULL	VARCHAR2(40)	User login name.
PAYEE_ID	NOT NULL	NUMBER(10)	DDN reference
PAYER_ACCT_NUM	NOT NULL	VARCHAR2(40)	User account number with biller.
DOC_DATE	NOT NULL	DATE	Doc date of index table, when the bill was indexed.
BILL_DUE_DATE	NULL	DATE	Bill due date.
BILL_AMOUNT_DUE	NULL	NUMBER(28,2)	Bill amount due.
MIN_AMOUNT_DUE	NULL	NUMBER(28,2)	Minimal amount due.
PAYMENT_ID	NOT NULL	NUMBER(28)	The payment ID of the payment made against this bill
FLEX_FIELD_1	NULL	VARCHAR2(255)	Available for customization.
FLEX_FIELD_2	NULL	VARCHAR2(255)	Available for customization.

PAYMENT_COUNTERS

This table generates counters. ePayment Manager uses this table to generate the ACH trace number, File ID Modifier, etc.

Name	**Null?**	**Type**	**Description**
COUNTER_NAME	NOT NULL	VARCHAR2(40)	Counter name.
COUNTER_VALUE	NOT NULL	NUMBER(28)	Counter value.
SEED	NOT NULL	NUMBER(28)	Counter start value.
INCREMENTAL	NOT NULL	NUMBER(28)	Counter incremental value.
MIN_VALUE	NOT NULL	NUMBER(28)	Counter minimal value.
MAX_VALUE	NOT NULL	NUMBER(28)	Counter maximal value.

PAYMENT_INVOICES

This table contains customer invoice information, usually obtained from eStatement.

Name	**Null?**	**Type**	**Description**
INV_ID	NOT NULL	NUMBER(28)	Unique invoice ID generated by ePayment Manager.
PAYER_ID	NOT NULL	VARCHAR2(40)	User login ID.
PAYEE_ID	NOT NULL	NUMBER(10)	DDN reference number.
PAYER_ACCOUNT_ NUMBER	NOT NULL	VARCHAR2(40)	User account number with biller.
INV_DATE	NULL	DATE	Date when the invoice is issued. Actually not used by ePayment Manager and it can be customized.
INV_NUMBER	NULL	VARCHAR2(40)	A string assigned by the biller to identify this invoice. Not used by ePayment Manager, so it can be used for customization.
INV_AMOUNT	NOT NULL	NUMBER(28,2)	Invoice amount.
INV_DUE_DATE	NULL	DATE	Invoice due date. Not used by ePayment Manager, so it can be used for customization.
INV_ISSUER	NULL	VARCHAR2(40)	The entity that issued the invoice. Not used by ePayment Manager, so it can be used for customization.

Name	**Null?**	**Type**	**Description**
INV_MEMO	NULL	VARCHAR2(250)	Invoice memo. Not used by ePayment Manager, so it can be used for customization.
INV_ISSUER	NULL	VARCHAR2(40)	The entity issues the invoice. Not used by ePayment Manager, so it can be used for customization.
AMT_TO_BE_PAID	NOT NULL	NUMBER(28,2)	The actual amount being paid for this invoice.
PROCESS_FLAG	NOT NULL	VARCHAR2(10)	This flag can be used by custom written jobs. Not used by ePayment Manager, so it can be used for customization.
PAYMENT_ID	NOT NULL	NUMBER(28)	The payment ID of the associated check payment.
TRACKING_NO	NULL	VARCHAR2(40)	Invoice tracking number. Not used by ePayment Manager, so it can be used for customization.
TRANSACTION_ DATE	NULL	DATE	Invoice transaction date. Not used by ePayment Manager, so it can be used for customization.
FLEXIBLE_FIELD_1	NULL	VARCHAR2(255)	Flexible field. Not used by ePayment Manager, so it can be used for customization.
FLEXIBLE_FIELD_2	NULL	VARCHAR2(255)	Flexible field. Not used by ePayment Manager, so it can be used for customization.
FLEXIBLE_FIELD_3	NULL	VARCHAR2(255)	Flexible field. Not used by ePayment Manager, so it can be used for customization.
FLEXIBLE_FIELD_4	NULL	VARCHAR2(255)	Flexible field. Not used by ePayment Manager and is customizable.
FLEXIBLE_FIELD_5	NULL	VARCHAR2 (1000)	Flexible field. Not used by ePayment Manager, so it can be used for customization.
BILL_ID	NULL	VARCHAR2(255)	The bill ID associated with the invoice.

PAYMENT_PROFILE

This table saves the Payment Settings information entered through the Command Center.

Name	**Null?**	**Type**	**Description**
PAYEE_ID	NOT NULL	NUMBER(10)	DDN reference number.
PAYMENT_TYPE	NOT NULL	VARCHAR2 (40)	“check”, “ccard” or “reminder”.
PARAM_NAME	NOT NULL	VARCHAR2(40)	Name of Payment Setting parameter.
PARAM_VALUE	NOT NULL	VARCHAR2(255)	Value of Payment Setting parameter.
CLOSE_DATE	NULL	DATE	Not used by ePayment Manager.

PAYMENT_LOG

This table saves payment reports. Whenever checks are submitted to gateway, a summary report is generated. Whenever there is a return file from gateway, an exception report is generated. Each report contains a list of name-value pairs.

Name	**Null?**	**Type**	**Description**
LOG_ID	NOT NULL	NUMBER(28)	Unique ID for this report.
PARAM_NAME	NOT NULL	VARCHAR2(80)	Report parameter name.
PARAM_VALUE	NULL	VARCHAR2(512)	Report parameter value.
BATCH_INDEX	NOT NULL	NUMBER(38)	ePayment Manager internal use, record the batch indexes in a payment record.

PAYMENT_REMINDERS

This table records the payment reminders set by the users through the ePayment Manager UI.

Name	**Null?**	**Type**	**Description**
PAYER_ID	NOT NULL	VARCHAR2(40)	Login ID.
PAYEE_ID	NOT NULL	NUMBER(10)	DDN reference.
START_DATE	NOT NULL	DATE	Date when this reminder will start.
REMINDER_INTERVAL	NOT NULL	VARCHAR2(20)	The reminder interval: monthly, weekly, etc.
NEXT_REMINDER_DATE	NOT NULL	DATE	The actual date the email will be sent out.
PAYER_EMAIL_ADDR	NOT NULL	VARCHAR2(50)	User’s email address for payment reminders.

Name	**Null?**	**Type**	**Description**
PARTITION_ID	NOT NULL	NUMBER(10)	For performance reasons, this table is partitioned into fixed-size buckets. The bucket size is configured through Payment Settings. This is the number of each bucket.
REMIND_STATUS	NOT NULL	VARCHAR2(20)	"active" or "inactive". Emails are only sent for active reminders.
USE_ENROLLMENT_EMAIL	NOT NULL	CHAR(1)	"Y" means use the email address from the *payment_profile* table. "N" means use the email address from this table.

RECURRING_PAYMENTS

Name	**Null?**	**Type**	**Description**
PAYEE_ID	NOT NULL	NUMBER(10)	DDN reference.
PAYER_ID	NOT NULL	VARCHAR2(40)	User login name.
PAYER_ACCT_NUM	NOT NULL	VARCHAR2(40)	User account number with biller.
PAYMENT_ACCT_NUM	NOT NULL	VARCHAR2(40)	For check, the check account number. For credit card, the card number.
PAYMENT_TYPE	NOT NULL	VARCHAR2(10)	“check”/”ccard”, check or credit card.
AMOUNT	NOT NULL	NUMBER(28,2)	Amount to be paid. For fixed amount, the amount specified from the UI. For pay amount due less than specified amount, the max amount specified. Otherwise, not used.
AMOUNT_TYPE	NOT NULL	VARCHAR2(40)	Can be: “fixed”, “amount due” or “less than”.
DAY_OF_PAY_INTERVAL	NOT NULL	NUMBER(12)	Pay date: the day of the pay interval. For monthly/quarterly: 1-31. For weekly: 1-7
MONTH_OF_PAY_INTERVAL	NOT NULL	NUMBER(12)	Applies to quarterly: 1-3

Name	**Null?**	**Type**	**Description**
PAY_INTERVAL	NOT NULL	VARCHAR2(20)	“monthly”, “weekly”, “quarterly”.
START_DATE	NOT NULL	DATE	When the recurring payment starts.
END_DATE	NOT NULL	DATE	When the recurring payment ends.
MAX_NUM_PAYMENTS	NOT NULL	NUMBER(12)	Maximal number of payments to be paid.
CURR_NUM_PAYMENTS	NOT NULL	NUMBER(12)	Current number of payments that have been paid.
STATUS	NOT NULL	VARCHAR2(10)	Whether the recurring payment has ended: "active" or "inactive"
EMAIL_IND	NOT NULL	CHAR(1)	Whether to send email when amount due is more than the amount specified: "Y" or "N"
BILL_ID	NULL	VARCHAR2(255)	DOC ID of the bill being paid, if applicable.
BILL_SCHEDULED	NOT NULL	CHAR(1)	Whether the current payment has been scheduled: "Y" or "N"
LAST_PROCESS_TIME	NOT NULL	DATE	The last time the job ran.
NEXT_PAY_DATE	NOT NULL	DATE	Pay date of the next payment available.
LAST_PAY_DATE	NOT NULL	DATE	Pay date of the last payment made.
FLEX_FIELD_1	NULL	VARCHAR2(255)	Flexible field. Not used by Payment, so it can be used for customization.
FLEX_FIELD_2	NULL	VARCHAR2(255)	Flexible field. Not used by ePayment Manager, so it can be used for customization.
VERTICAL_FIELD_1	NULL	VARCHAR2(255)	Not used by ePayment Manager, so it can be used for customization.
VERTICAL_FIELD_2	NULL	VARCHAR2(255)	Not used by ePayment Manager, so it can be used for customization.
VERTICAL_FIELD_3	NULL	VARCHAR2(255)	Not used by ePayment Manager, so it can be used for customization.

Name	**Null?**	**Type**	**Description**
VERTICAL_FIELD_4	NULL	VARCHAR2(255)	Not used by ePayment Manager, so it can be used for customization.
PID	NOT NULL	VARCHAR2(255)	The unique ID that identifies this payment account.
NOTIFY_REQUIRED	NOT NULL	VARCHAR2(1)	
GROUP_ID	NULL	VARCHAR2(255)	In a B2B application, a recurring payment set up by one person can be viewed/modified by another, depending on their authorization, and is managed outside ePayment Manager. EPayment Manager provides this flexible field for use in maintaining this relationship.
GROUP_ID_REF	NULL	VARCHAR2(255)	In a B2B application, a recurring payment set up by one person can be viewed/modified by another, depending on their authorization, and is managed outside ePayment Manager. EPayment Manager provides this flexible field for use in maintaining this relationship.
PAYEE_BANK_ACCOUNT_ID	NULL	VARCHAR2(255)	For check payments, this field is used to store the ID of the payee bank account.

Payment Indexes

The following table lists the indexes defined on ePayment Manager and enrollment tables:

Table name	**Index name**	**Indexed columns**
payment_profile	pk_payment_profile	payee_id, payment_type, param_name
check_payments	pk_check_payments	payment_id
check_payments	nuk_check_payments_1	status, pay_date
check_payments	nuk_check_payments_2	status, reminded
check_payments	nuk_check_payments_3	gateway_payment_id
check_payments	nuk_check_payments_4	payer_id

Table name	**Index name**	**Indexed columns**
check_payments	nuk_check_payments_5	partition_id
check_payments_history	nuk_check_payments_history_1	log_id, status
payment_invoices	pk_payment_invoices	inv_id
payment_invoices	nuk_payment_invoices_1	payment_id
payment_invoices	nuk_payment_invoices_2	payee_id, process_flag, inv_amount
credit_card_payments	pk_credit_card_payments	payment_id
credit_card_payments	nuk_credit_card_payments_1	payee_id, partition_id, payer_id, status, pay_date
payment_reminders	pk_payment_reminders	payer_id,payee_id
payment_reminders	nuk_payment_reminders_1	payee_id, partition_id, remind_status, next_reminder_date
payment_log	index_payment_log_1	param_name, param_value
payment_log	index_payment_log_2	log_id
payment_counters	pk_payment_counters	counter_name
recurring_payments	pk_recurring_payments	payer_id, payee_id, payer_acct_num
reccurring_payments	nuk_recur_payment_2	status, bill_scheduled, next_pay_date
payment_bill_summaries	pk_payment_bill_summaries	bill_id
payment_bill_summaries	nuk_pymt_bill_summary_2	payer_id

Database Migration

The ePayment Manager database is designed to migrate from a previous version to new version, whenever ePayment Manager upgrades the database schema. If you have a payment database from an older version of ePayment Manager, and you want to upgrade to a newer version, just run the install script that comes with the newer version. The installation script automatically alters the existing schema while preserving the old data. However, since the ePayment Manager database depends on the *document_definition_name* table from eStatement, it's very important to make sure that this table is migrated, too. If a DDN name/reference is removed or changed during migration, it will cause problems for ePayment Manager.

Job Scheduling

You should schedule payment jobs to run when there is not much user activity, which is typically after midnight.

If two jobs access the same table, schedule them to run at different times. pmtCheckSubmit, pmtCheckUpdate, PmtPaymentReminder pmtSubmitEnroll, pmtConfirmEnroll and pmtNotifyEnroll should run sequentially. Allow enough time between each job so that two jobs won't access the same database table at the same time. In some cases, two jobs trying to access the same table at the same time could cause a database access error.

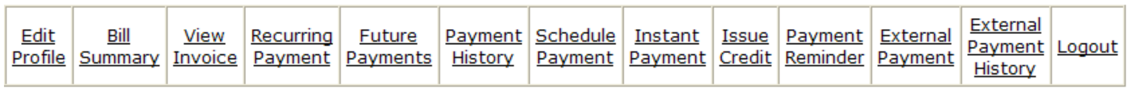
The pmtAllCheckTasks job runs all the ePayment Manager jobs sequentially. You can also edit pmtAllCheckTasks to not run specific jobs, if you wish to tailor your environment.

12 Sample User Interface

Navigating ePayment Manager Sample Interface

ePayment Manager provides a sample interface, paymentComplex, that demonstrates the standard features available for a customer interface. Each customer presents his or her own interface, but the Sample interface is useful for testing the system.

This section describes the ePayment Manager sample interface. At the top of each page is the ePayment navigation menu of payment and account management options:



Menu Option	Description
Edit Profile	Lets a user modify and view enrollment information.
Bill Summary	Shows all payment activity for this account.
View Invoice	Shows a list of invoices for paid bills.
Recurring Payment	Lets a user schedule a recurring payment and view the existing setups.
Future Payments	Shows all future bill payments scheduled for the customer account.
Payment History	Shows a summary of payment history for the customer account.
Schedule Payment	Lets the user schedule a payment.
Instant Payment	Lets the user make an instant payment using a credit card (not scheduled).
Issue Credit	Lets the user schedule a credit on a specified bill.
Payment Reminder	Lets the user configure a payment reminder for any account. ePayment Manager sends an email notification to remind you to pay a bill.
External Payment	Lets the user make an external payment.
External Payment History	Shows a history of external payments.
Logout	Logs out of the paymentComplex interface.

Enrolling for Bill Payment and Account Management

This section describes a sample customer enrollment for online bill payment and personal account management, and demonstrates how ePayment Manager's online payment capabilities can be implemented to create an effective online billing solution.

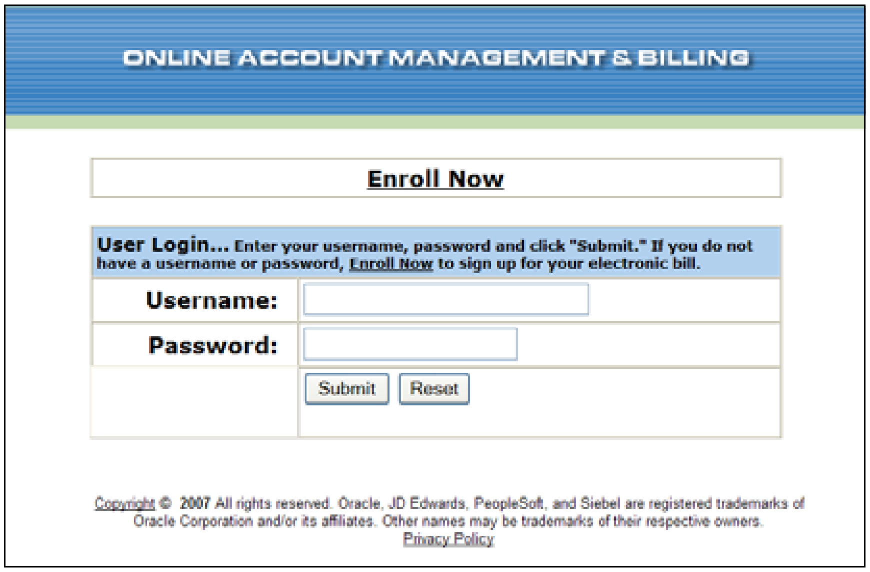
Customers enroll once, after which they can log in and perform a variety of payment and account management tasks, such as viewing the details of a bill, scheduling future payments, checking bill history, and configuring payment reminders.

To enroll customers for online bill payment:

1. In the Command Center, verify that a remote payment gateway has been configured.
2. Access the enrollment User Login page by entering the URL below, substituting the name of your Web Server for <webserver> and existing application:

   `http://<WEBSERVER>:<PORT>/paymentComplex/`

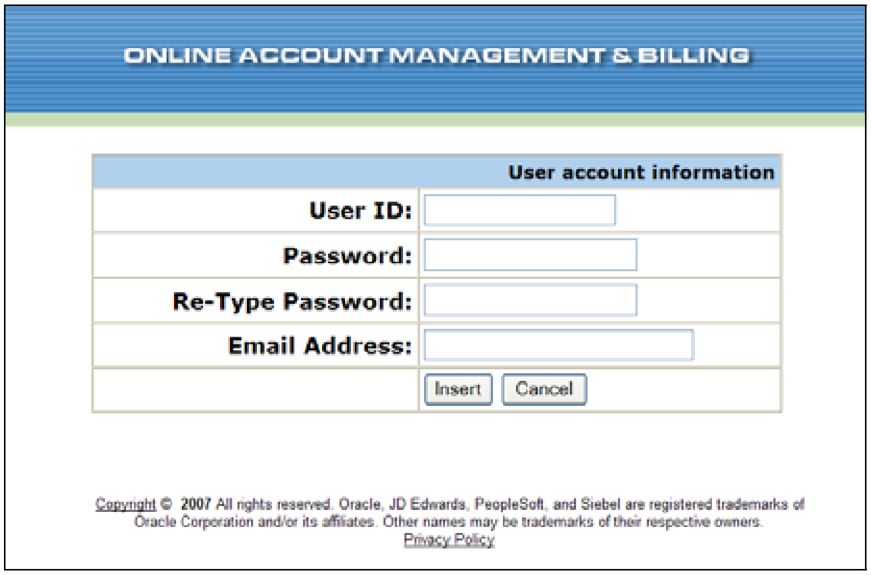
   The Enrollment page appears:



User Login... Enter your username, password and click "Submit." If you do not have a username or password, Enroll Now to sign up for your electronic bill.

Copyright © 2007 All rights reserved. Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
Privacy Policy

3. Log on or click **Enroll Now** to create a new user profile. The following screen appears:



ONLINE ACCOUNT MANAGEMENT & BILLING

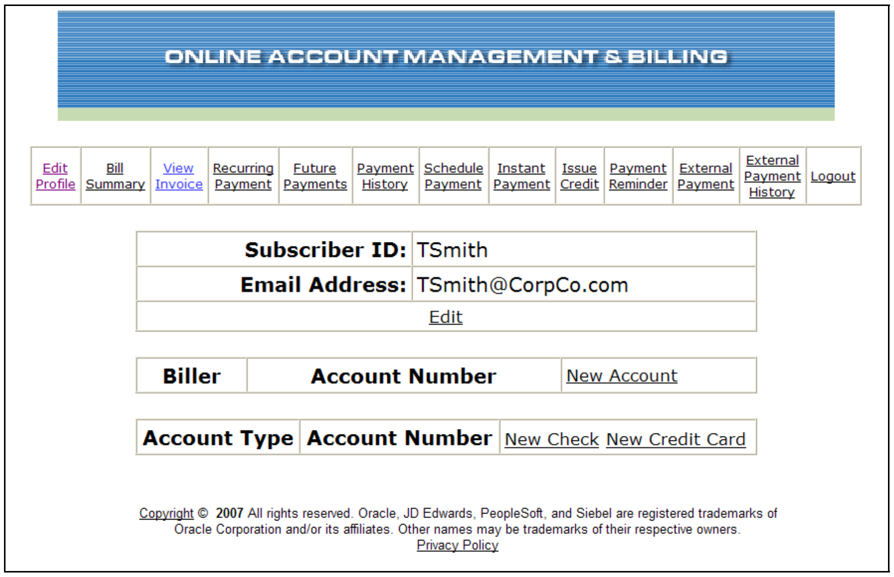
	User account information
User ID:	
Password:	
Re-Type Password:	
Email Address:	
	Insert Cancel

Copyright © 2007 All rights reserved. Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
Privacy Policy

4 Create a User ID for yourself and supply a password and email address. Click **Insert**. A message appears indicating that your subscription was successful.

5 Click **Click Here** to login to your account. A new Login page appears. Enter your **Username** and **Password** and click **Submit**. The Schedule Payment screen appears by default.

6 Before you can use the ePayment Manager features, you must create a profile that includes payment account information. Click **Edit Profile** to display the current profile information:



ONLINE ACCOUNT MANAGEMENT & BILLING

Edit Profile	Bill Summary	View Invoice	Recurring Payment	Future Payments	Payment History	Schedule Payment	Instant Payment	Issue Credit	Payment Reminder	External Payment	External Payment History	Logout

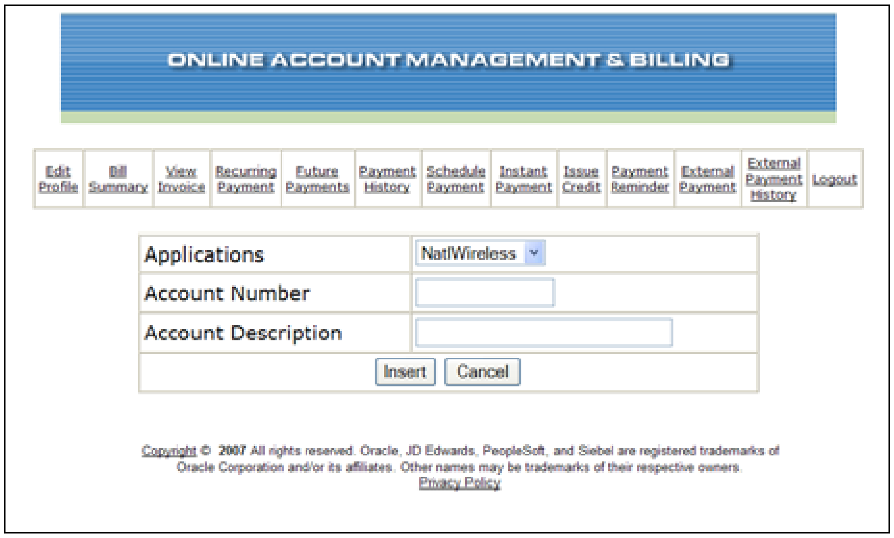
Subscriber ID:	TSmith
Email Address:	TSmith@CorpCo.com
Edit	

Biller	Account Number	New Account

Account Type	Account Number	New Check New Credit Card

Copyright © 2007 All rights reserved. Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
Privacy Policy

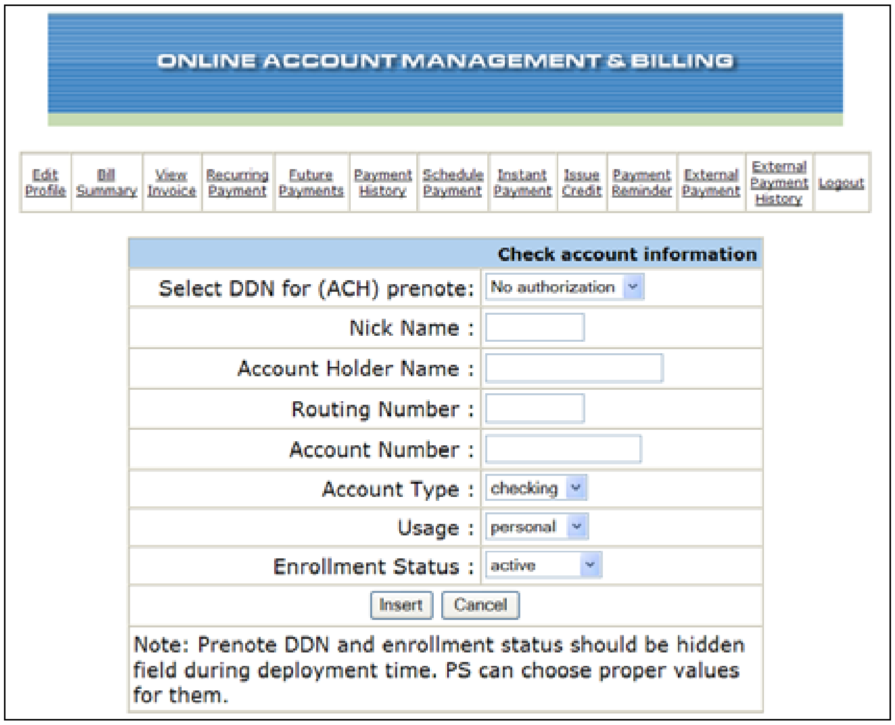
7 Click `New Account`. Enter an account number and description, and click **Insert**.



Applications	NatlWireless
Account Number	
Account Description	

Copyright © 2007 All rights reserved. Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
Privacy Policy

8 Click **Edit Profile** again. (You can enter multiple accounts if needed.) Click **New Check**, for example, to enter check account information:



Check account information	
Select DDN for (ACH) prenote:	No authorization
Nick Name :	
Account Holder Name :	
Routing Number :	
Account Number :	
Account Type :	checking
Usage :	personal
Enrollment Status :	active

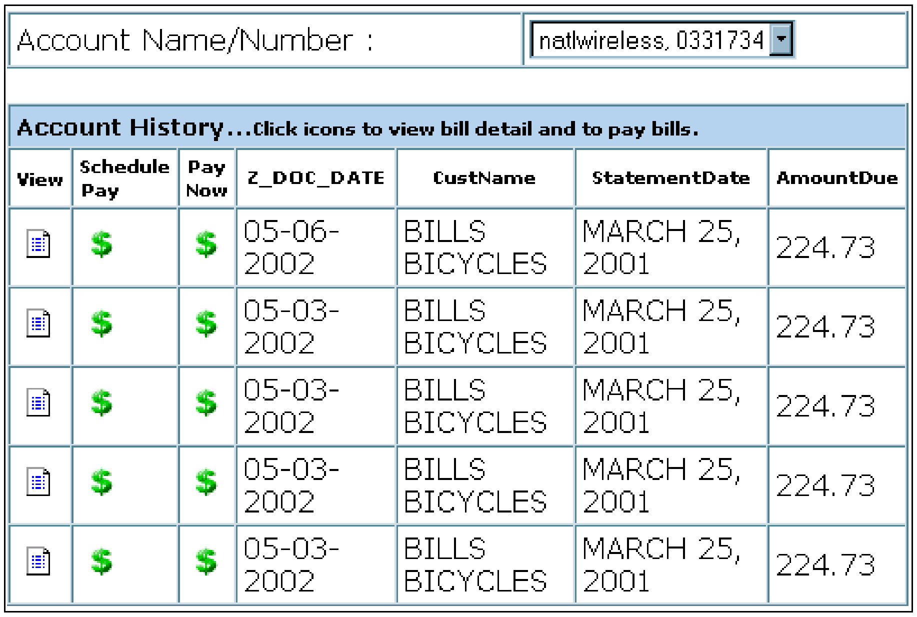
Note: Prenote DDN and enrollment status should be hidden field during deployment time. PS can choose proper values for them.

9 Click **Insert** to add this to the list checking accounts. A message appears indicating that the check account has been added.

The Account Payment Cycle

When customers enroll and make a few payments, they typically view the Bill Summary page to see which bills are coming due, and to make a payment.

1 Click **Bill Summary** on the navigation menu. The system displays details about the bills due. Click the view icon to view a detailed version of the bill.

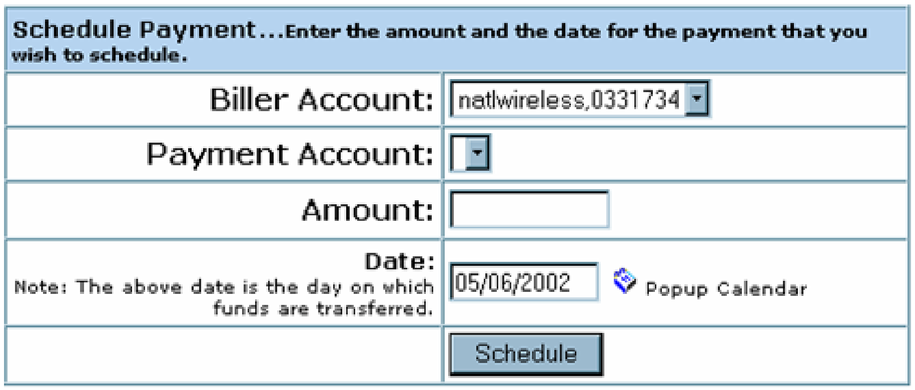


Account Name/Number : natlwireless, 0331734

Account History...Click icons to view bill detail and to pay bills.

View	Schedule Pay	Pay Now	Z_DOC_DATE	CustName	StatementDate	AmountDue
	$	$	05-06-2002	BILLS BICYCLES	MARCH 25, 2001	224.73
	$	$	05-03-2002	BILLS BICYCLES	MARCH 25, 2001	224.73
	$	$	05-03-2002	BILLS BICYCLES	MARCH 25, 2001	224.73
	$	$	05-03-2002	BILLS BICYCLES	MARCH 25, 2001	224.73
	$	$	05-03-2002	BILLS BICYCLES	MARCH 25, 2001	224.73

2 Click the Schedule Pay $ icon to open a Make Payment page similar to the following:



Schedule Payment...Enter the amount and the date for the payment that you wish to schedule.

Biller Account:	natlwireless,0331734
Payment Account:	
Amount:	
Date: Note: The above date is the day on which funds are transferred.	05/06/2002 Popup Calendar
	Schedule

3 To simulate a check payment session, enter:

The Biller Account to make a payment against

The payment account, if more than one account can be paid for this biller

The amount to pay

The date on which you want to transfer funds from the account to pay the bill in the Date field

A Popup Calendar is available to help you schedule a payment date.

4 Click **Schedule**. A confirmation dialog appears.

5 Click **OK** to confirm the scheduled payment. ePayment Manager returns a dialog showing the name of the payee, date on which the payment is scheduled, and the payment amount.

Other ePayment Manager Functionality

Other payment and account management options available in the paymentComplex sample interface are:

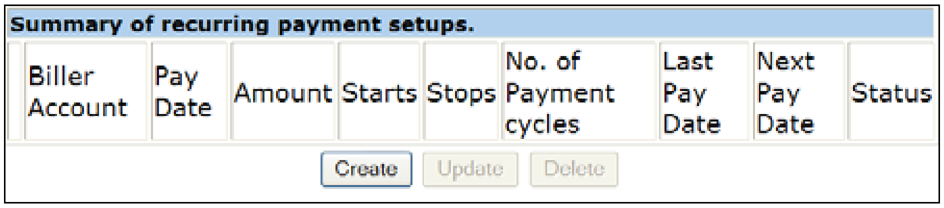
Link from Top ePayment Manager Menu	Description
Recurring Payment	Lets a user schedule a recurring payment and view the existing setups.
Future Payments	Shows all future bill payments scheduled for the customer account.
Payment History	Shows a summary of payment history for the customer account.
Instant Payment	Lets the user make an instant payment using a credit card (not scheduled).
Issue Credit	Lets the user schedule a credit on a specified bill.
Payment Reminder	Lets the user configure a payment reminder for any account. ePayment Manager sends an email notification to remind you to pay a bill.
External Payment	Lets the user make an external payment.
External Payment History	Shows a history of external payments.

Recurring Payments

The Recurring Payments option lets you set up a recurring payment for a biller and view a summary of all your recurring payment setups.

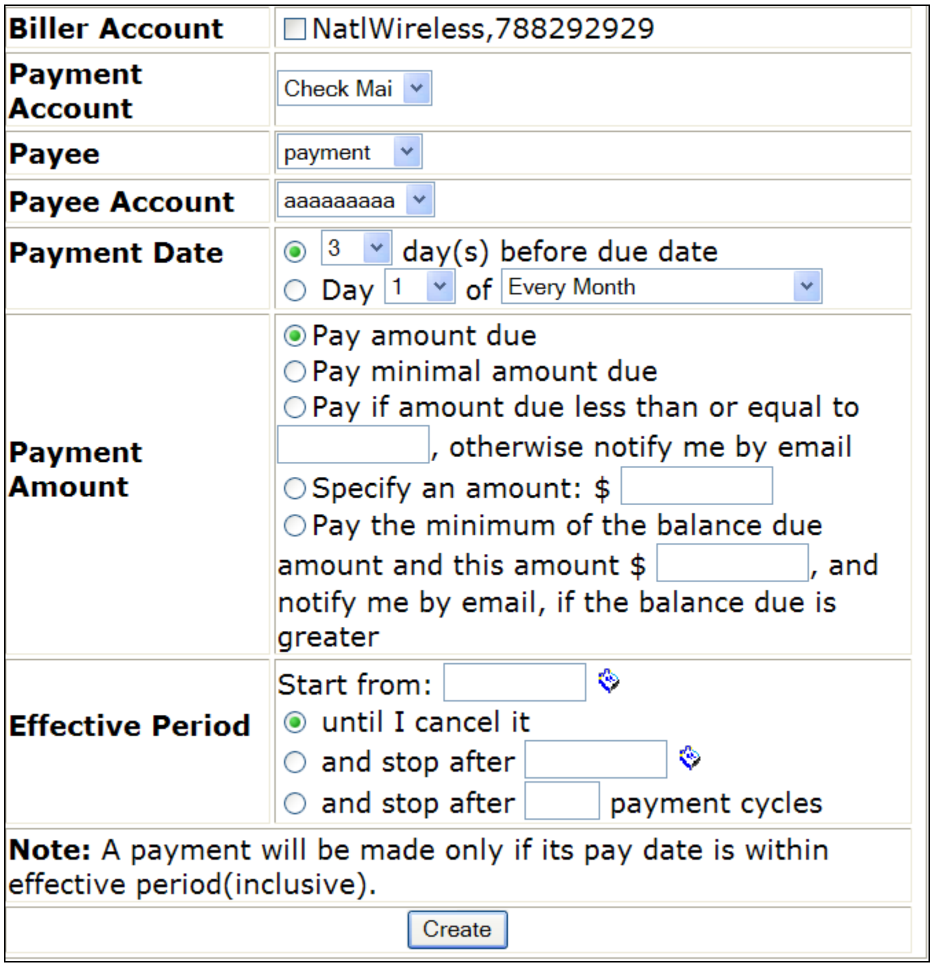
To create a recurring payment (or view a summary):

1 Click **Recurring Payments** on the ePayment Manager navigation menu at the top of the page. The Summary of Recurring Payment Setups page appears. If you have already set up recurring payments, these are shown.


Summary of recurring payment setups.

	Biller Account	Pay Date	Amount	Starts	Stops	No. of Payment cycles	Last Pay Date	Next Pay Date	Status

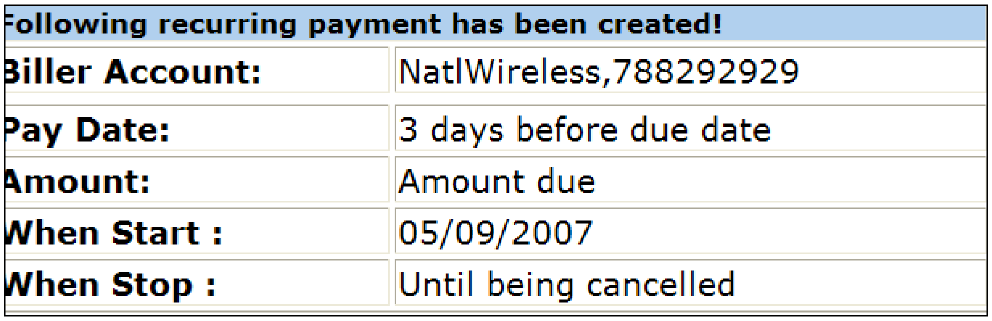
2 To create a recurring payment, click **Create**. The Recurring Payment Setup page appears.



Biller Account	☐ NatlWireless,788292929
Payment Account	Check Mai
Payee	payment
Payee Account	aaaaaaaaa
Payment Date	[●] 3 day(s) before due date ○ Day 1 of Every Month
Payment Amount	[●] Pay amount due ○ Pay minimal amount due ○ Pay if amount due less than or equal to ____, otherwise notify me by email ○ Specify an amount: $ ____ ○ Pay the minimum of the balance due amount and this amount $ ____, and notify me by email, if the balance due is greater
Effective Period	Start from: ____ [●] until I cancel it ○ and stop after ____ ○ and stop after ____ payment cycles
Note: A payment will be made only if its pay date is within effective period(inclusive).	

Create

3 Enter the necessary information. (If the pay date is related to the due date, or the pay amount is related to amount due, then ePayment Manager must query eStatement for information about the bill, which incurs extra processing during the pmtRecurringPayment job.) The system displays a confirmation:



ollowing recurring payment has been created!

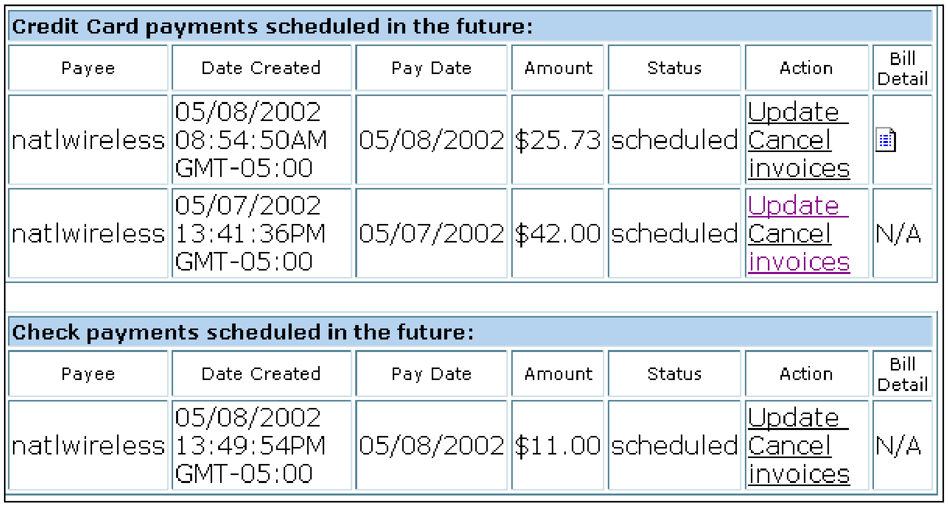
iller Account:	NatlWireless,788292929
ay Date:	3 days before due date
mount:	Amount due
hen Start :	05/09/2007
hen Stop :	Until being cancelled

Future Payments

The Future Payments option shows a summary page of scheduled payments.

To view a summary of future bill payments:

1. Click **Future Payments** on the navigation menu. A future payment page appears showing the name of the biller, the date the payment was created, scheduled payment date, amount of payment, and payment status.



Credit Card payments scheduled in the future:

Payee	Date Created	Pay Date	Amount	Status	Action	Bill Detail
natlwireless	05/08/2002 08:54:50AM GMT-05:00	05/08/2002	$25.73	scheduled	Update Cancel invoices	
natlwireless	05/07/2002 13:41:36PM GMT-05:00	05/07/2002	$42.00	scheduled	Update Cancel invoices	N/A

Check payments scheduled in the future:

Payee	Date Created	Pay Date	Amount	Status	Action	Bill Detail
natlwireless	05/08/2002 13:49:54PM GMT-05:00	05/08/2002	$11.00	scheduled	Update Cancel invoices	N/A

2. Click **Update** in the Action column to modify the payment (such as payment amount, date, or biller), **invoices** to list the invoices that were part of that payment, or **Cancel** to remove the payment from the list. Canceling a payment requires that you confirm the action.

Payment History

The payment history option lets users view a list of payments made.

To view payment history:

1. Click **Payment History** on the navigation bar. The Payment History page appears showing information about the Payee (also known as the biller, DDN or Application), the date the payment was scheduled, the date the payment was made, the payment amount, and the status of payment. The invoices link brings up the invoices (if any) that were included in that payment.

2. If you do not see payments that should have run, check the input directory to see if the transaction file has been processed. pmtCheckUpdate moves processed files into a subdirectory named *history_YYYYMM*, where YYYYMM is the year and month of the day the files in that subdirectory were processed.

Returning Invoice Details

ePayment Manager pages listing payment summary information, such as Payment History and Future Payments, list at least one invoice for each payment. It is possible to create a link for each invoice to the invoice details.

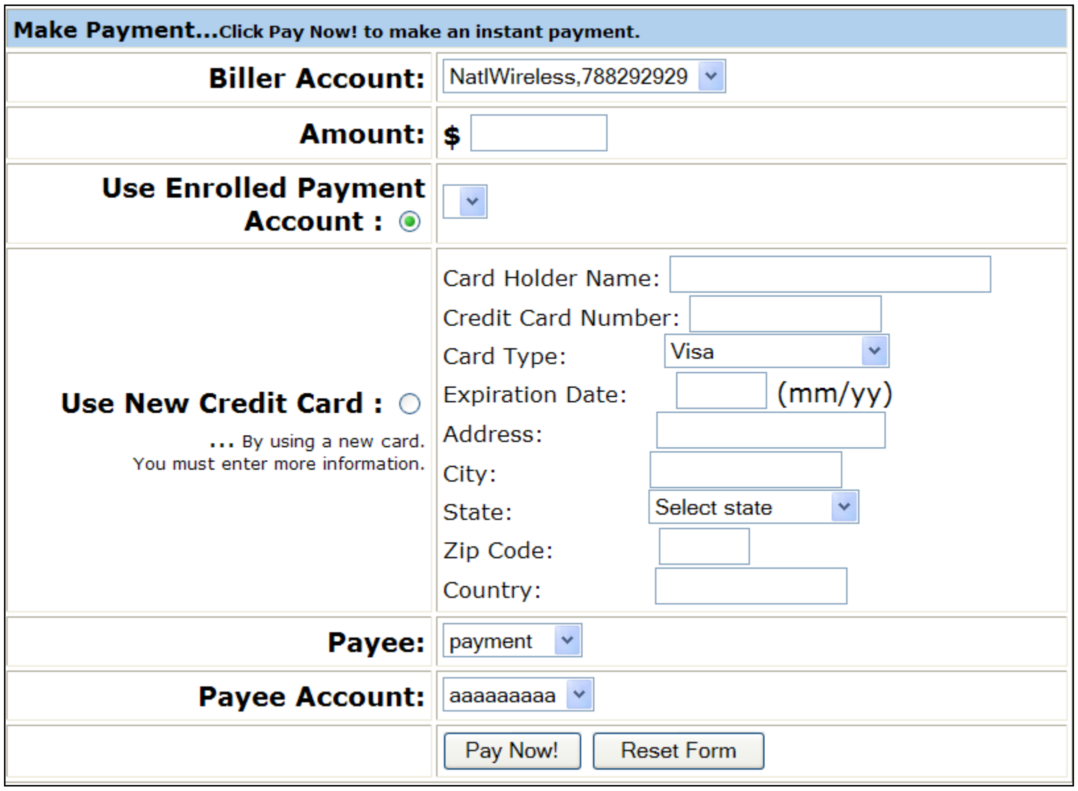
You create these links by customizing JSP files to call *showbill.jsp*. *Showbill.jsp* is shipped as an example of how to retrieve the invoice details.

Instant Payment

The Instant Payments option lets a user make a one-time, unscheduled payment using a credit card. You cannot make an instant payment with a check account.

To make an instant payment:

1. Click **Instant Payment** on the navigation menu. The following screen shows the Instant Payment page:



Make Payment...Click Pay Now! to make an instant payment.

Biller Account:	NatlWireless,788292929
Amount:	$
Use Enrolled Payment Account : ●	
Use New Credit Card : ○ ... By using a new card. You must enter more information.	Card Holder Name: Credit Card Number: Card Type: Visa Expiration Date: (mm/yy) Address: City: State: Select state Zip Code: Country:
Payee:	payment
Payee Account:	aaaaaaaaa
	Pay Now! Reset Form

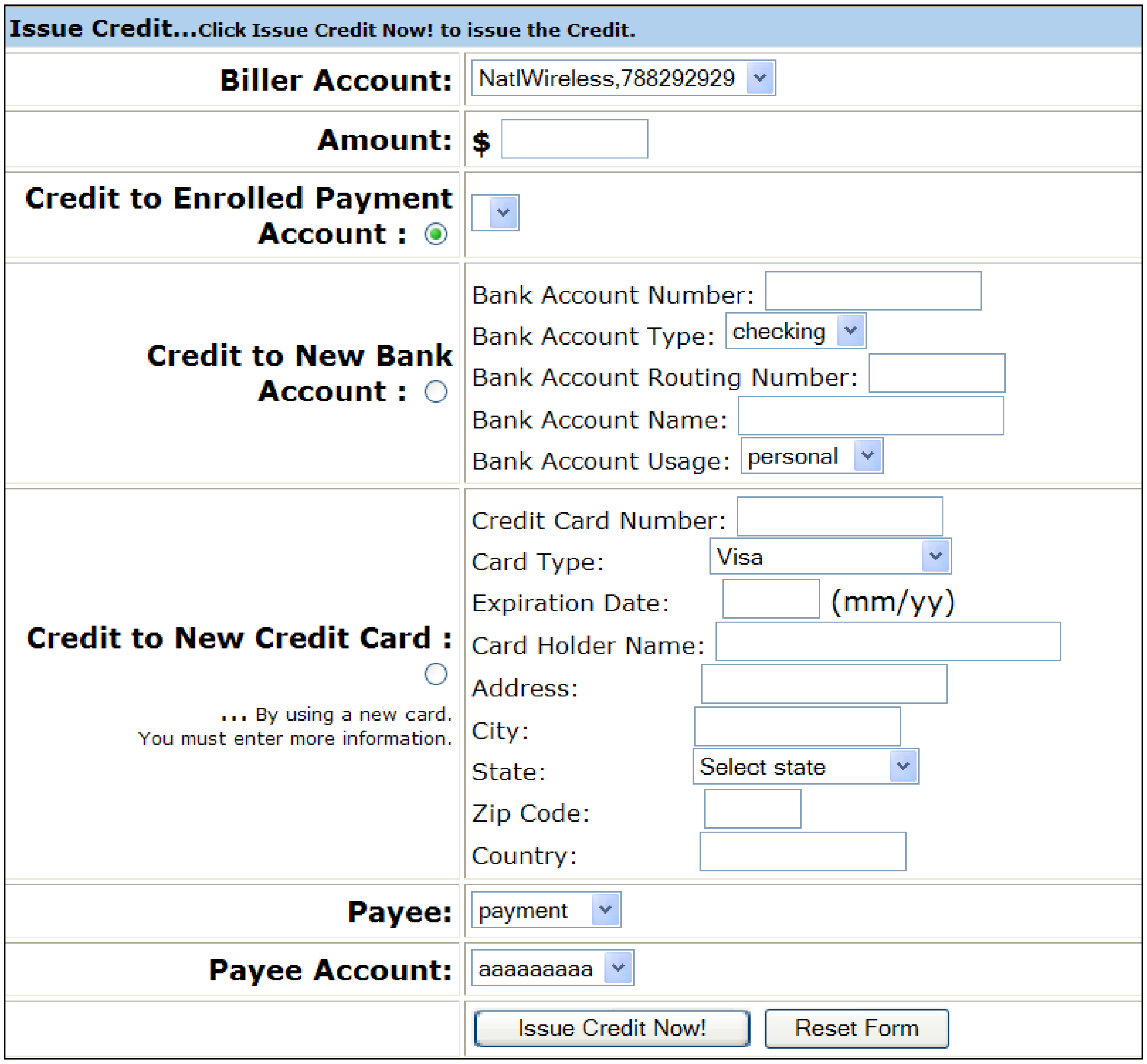
2. Enter the required information and click **Pay Now**.

Issue Credit

You can issue a credit on an account, also known as a payment reversal.

To issue a credit on an account:

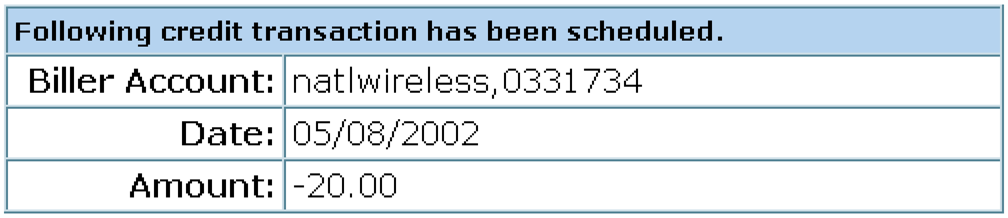
1. Click **Issue Credit** on the navigation bar. The system displays the Issue Credit page:



Issue Credit... Click Issue Credit Now! to issue the Credit.

Biller Account:	NatlWireless,788292929
Amount:	**$**
Credit to Enrolled Payment Account : [✓ ●]	
Credit to New Bank Account : ○	Bank Account Number: Bank Account Type: checking Bank Account Routing Number: Bank Account Name: Bank Account Usage: personal
Credit to New Credit Card : ○ **...** By using a new card. You must enter more information.	Credit Card Number: Card Type: Visa Expiration Date: (mm/yy) Card Holder Name: Address: City: State: Select state Zip Code: Country:
Payee:	payment
Payee Account:	aaaaaaaaa
	Issue Credit Now! Reset Form

2. Enter the required information and click **Issue Credit Now.** The system schedules a payment reversal that will settle when the pmtCheckSubmit or pmtCreditCardSubmit job runs, depending on which type of account the credit was issued against. The system displays a status screen confirming that the credit was scheduled:



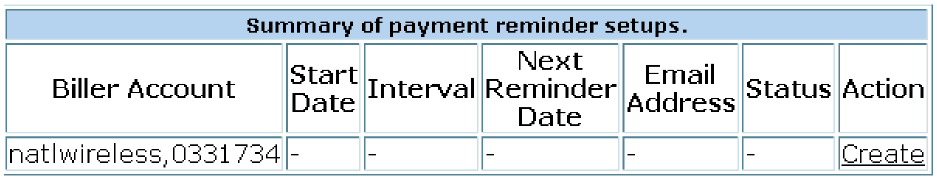
Following credit transaction has been scheduled.	
Biller Account:	natlwireless,0331734
Date:	05/08/2002
Amount:	-20.00

Payment Reminder

By configuring a payment reminder, you can have ePayment Manager automatically send you an email notification prior to the date your bill is due. Payment reminders can be set to a number of intervals, ranging from weekly to quarterly. From this page you an also deactivate previously configured payment reminders.

To configure a payment reminder:

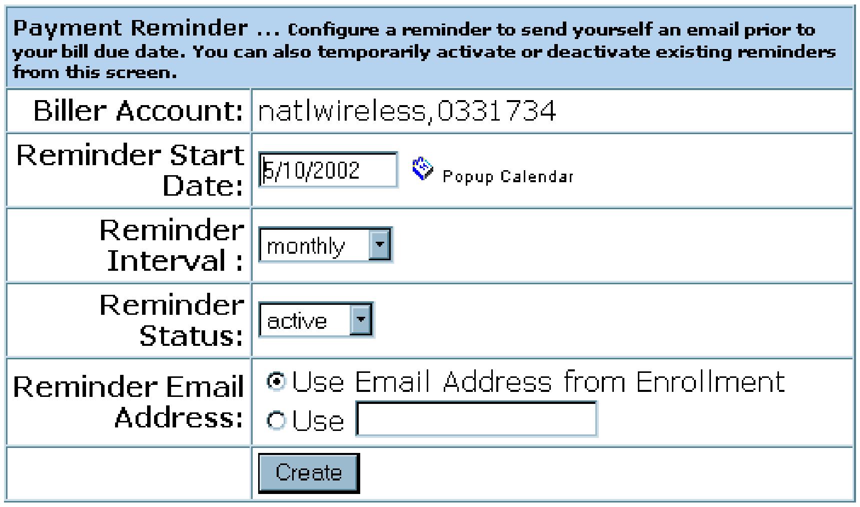
1. Click **Payment Reminder** on the payment options menu. A Payment Reminder page appears where you choose the account you wish to see. After choosing an account, the screen updates to allow you to create a payment reminder for the chosen account.



Summary of payment reminder setups.

Biller Account	Start Date	Interval	Next Reminder Date	Email Address	Status	Action
natlwireless,0331734	-	-	-	-	-	Create

2. Click **Create** to create a new reminder. The system displays the Payment Reminder page:



Payment Reminder ... Configure a reminder to send yourself an email prior to your bill due date. You can also temporarily activate or deactivate existing reminders from this screen.

Biller Account:	natlwireless,0331734
Reminder Start Date:	5/10/2002 Popup Calendar
Reminder Interval :	monthly
Reminder Status:	active
Reminder Email Address:	[✓ Use Email Address from Enrollment] Use
	Create

3. Enter a start date in the Reminder Start Date field (a Popup Calendar is available to help you determine the date). Select a reminder interval from the Reminder Interval drop-down menu. Choices are weekly, bi-weekly, monthly, and quarterly.

   To deactivate a payment reminder, select **inactive** from the Reminder Status drop-down menu. No further payment reminders will be sent until you activate the option again.

In the Reminder Email Address field, enter an email address where you want email notification sent if you wish to use a different one than you enrolled with.

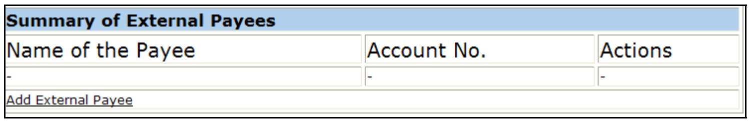
4 Click **Create**.

External Payment

The paymentComplex application lets you make external payments (to an external biller). You must first set up the Payment Settings in the Command Center for the sample application to use external payments. See the instructions in this guide for details.

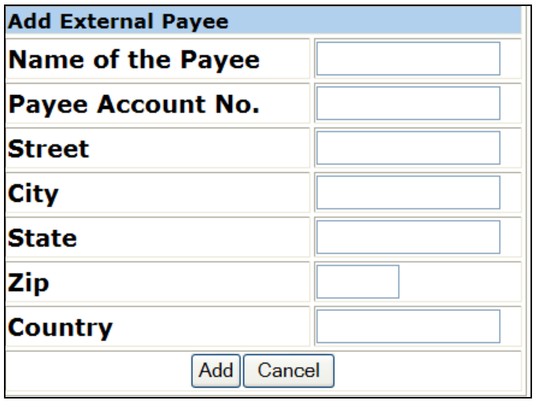
To create an external payee and make a payment:

1 Click **External Payment** on the payment options menu. The Summary of External Payees page appears:



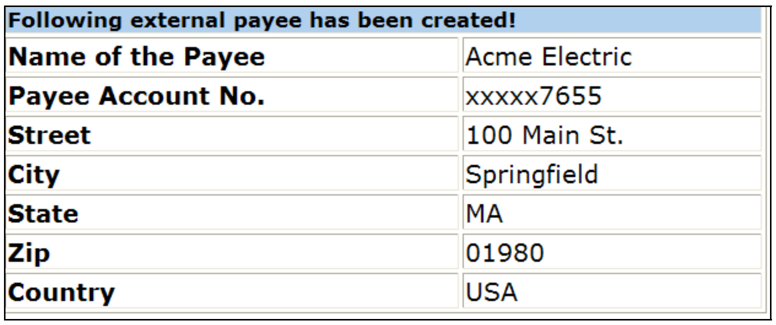
Summary of External Payees		
Name of the Payee	Account No.	Actions
-	-	-
Add External Payee		

2 Click **Add External Payee**. The Add External Payee page appears. Enter the required information for the external payee:



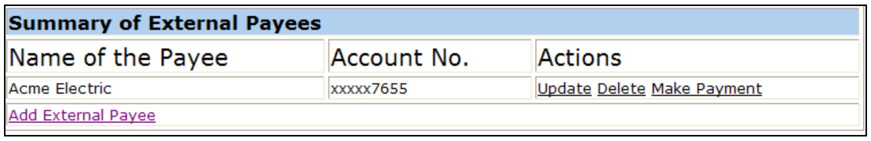
Add External Payee	
Name of the Payee	
Payee Account No.	
Street	
City	
State	
Zip	
Country	

3 Click **Add**. The verification screen appears:



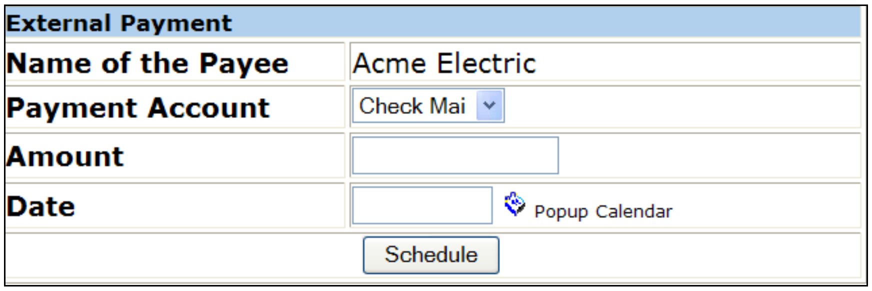
Following external payee has been created!	
Name of the Payee	Acme Electric
Payee Account No.	xxxxx7655
Street	100 Main St.
City	Springfield
State	MA
Zip	01980
Country	USA

4 Click **External Payment** again.



Summary of External Payees		
Name of the Payee	Account No.	Actions
Acme Electric	xxxxx7655	Update Delete Make Payment
Add External Payee		

5 Click **Make Payment**. The External Payment page appears. Select an account to use, amount to pay, and date (a popup calendar is available.)



External Payment	
Name of the Payee	Acme Electric
Payment Account	Check Mai
Amount	
Date	Popup Calendar
	Schedule

6 Click **Schedule**, and click **OK** when prompted to verify the payment.

External Payment History

You can view a history of external payments in the sample application. The External Payment History screen lists the following details for each external payment: External Payee, Date Created, Pay Date, Amount, Status, and Action.

To view external payment history:

1 Click **External Payment History** on the payment options menu. The External Payment History screen appears.

13 Email Template Customization

Using Email Templates

ePayment Manager uses template files to generate customized text that will be sent in a notification email. Professional Services customizes the email templates for you as part of system installation. This chapter describes how email templates can be customized.

Separate email notification templates are used for:

Type of notification	Task that Specifies	Template File
Enrollment status	pmtNotifyEnroll	*NotifyEnroll.txt*
Reminder to pay bills and the status of the checks	pmtPaymentReminder	*paymentReminder.txt*
Recurring payment scheduled	pmtRecurringPayment	*recurringNotify.txt*
ePayment Manager Command Center job status	All Payment jobs	*notifyPayment.txt*
Notify that credit card has expired.	pmtCreditCardExpNotify	*CCExpNotify.txt*

For UNIX, the default path to the email template files is:
<PAYMENT_HOME>/lib/payment_resources/

For Windows, the default path is:
<PAYMENT_HOME>\lib\payment_resources\

You can modify the text in these templates, and use the existing variables that reference the ePayment Manager template engine. To add new variables, see the *Customizing and Extending Oracle Siebel ePayment Manager* guide or contact Oracle Professional Services.

Index