eaPay™ Production Guide

eaPay™ is a member of the eaSuite™ product line

V3.0
Document ID: PAGN-05-3.0-01
Date Published: 03.01.2002

© 1997–2002 edocs® Inc. All rights reserved.

edocs, Inc., Two Apple Hill, 598 Worcester Road, Natick, MA 01760

The information contained in this document is the confidential and proprietary information of edocs, Inc. and is subject to change without notice.

This material is protected by U.S. and international copyright laws. edocs and eaPost are registered in the U.S. Patent and Trademark Office.

No part of this publication may be reproduced or transmitted in any form or by any means without the prior written permission of edocs, Inc.

eaSuite, eaDirect, eaPay, eaProcess, eaService, eaMarket, and eaXchange are trademarks of edocs, Inc.

All other trademark, company, and product names used herein are trademarks of their respective companies.

Printed in the USA.

Table of Contents

Preface **7**
About Online Account Management and the eaSuite™ 7
About This Guide 9
Related Documentation 10
If you need help 11

1 Overview of eaPay **13**
What's New in eaPay 3.0? 15
Payment Transaction Manager and Payment Cartridges 15
eaPay Jobs 16

2 Enrollment **21**
Enrollment Overview 21
eaPay Enrollment 24

3 Check Payments **25**
Check Payment Overview 25
Adding a Check Account 25
Enrollment Status Flow 27
Prenote Status Flow 28
Check Payment Transactions 29
Check Payment Status Flow 31
User Options 32
CheckFree Direct Pay (CDP) 33
CheckFree Files and Directories 33
CheckFree Transactions 35
ACH 37
ACH Change Codes 37
ACH Return Codes 37
NOC Transactions 39
ACH Effective Date 39

ACH Settlement Date 40
ACH Addenda Records 40
Payment Portal Feature 40
4 Credit Card Payments 41
Credit Card Payment Status 42
Credit Card Payment Transactions 43
Instant Credit Card Payments 43
Scheduled Credit Card Payments 44
User Options 45
Using VeriSign as a Payment Gateway 46
AVS (Address Verification Service) 46
5 Recurring Payments 47
Overview 47
Recurring Payment Transaction Cycle 49
Tables Affected by Recurring Payments 51
Recurring Payment Examples 51
Scheduling Payment Jobs 62
Payment Job Status Monitoring 62
Payment Job Plug-In 62
To Configure Recurring Payments 63
Testing Recurring Payment 63
Case 1: Pay Amount Due *X* days Before Due Date 64
Case 2: Pay Amount Due on a Fixed Date 65
Case 3: Pay fixed Amount *X* Days Before Due Date 66
Case 4: Pay Fixed Amount On A Fixed Date 67
6 Email Notifications 69
7 Configuring Payment Gateways 71
Configuring a Payment Gateway 71
Check Payment Gateway 73
ACH Gateway Parameters 74
ACH Federal Holidays 80
CheckFree Gateway Parameters 80
Credit Card Payment Gateway 85
Updating a Payment Gateway Configuration 90
Deleting a Payment Gateway Configuration 91

8 Configuring eaPay Jobs 93
pmtAllCheckTasks Job 93
pmtARIntegrator Job 94
Configuration 94
pmtCheckSubmit Job 97
Scheduling and Holidays 98
Configuration 99
pmtCheckUpdate Job 101
pmtConfirmEnroll Job 106
pmtCreditCardSubmit Job 107
pmtCustom 109
pmtNotifyEnroll Job 110
pmtNotifyEnroll Job Email Format 111
pmtPaymentReminder Job 112
pmtPaymentReminder Job Configuration 112
pmtPaymentReminder Operation 114
pmtRecurPayment Job 116
pmtSubmitEnroll Job 119
pmtSubmitEnroll Configuration 119
9 eaPay Reports 121
Overview 121
Viewing eaPay Reports 121
Credit Card Gateways 123
Viewing eaPay Module Error Logs 123
10 eaPay Administration 127
eaPay Database 127
Preventing Multiple Payments 127
UI Actions and Database Changes 128
Table Sizing 130
Table Maintenance 133
Backup and Recovery 133
Schema 134
Table Column Definitions 134
eaPay Tables 134
Payment indexes 153
Database Migration 155

Changes From eaPay 2.1 to eaPay 3.0 155
Job Scheduling 156
11 Example User Interface 157
Navigating eaPay Example Interface 157
Enrolling for Bill Payment and Account Management 158
The Account Payment Cycle 162
Other Payment Pages 165
Payment History 165
Payment Reminder 166
Future Payments 168
Recurring Payments 169
Making an Instant Payment 171
Issuing Credit 172
12 Email Template Customization 175
Email templates and when they are used 175

Preface

About Online Account Management and the eaSuite™

Online Account Management solutions provide organizations and their business and consumer customers with the capability to service every aspect of their account relationship over the Internet. The account encapsulates all pertinent customer information, along with the activity and charges associated with the exchange of goods or services between a customer and an organization, and as such serves as the foundation for the relationship.

Whether an organization provides physical goods or services, the account relationship is one-time or recurring; or the customer is a large business with a global presence or is an individual consumer, the needs remain the same. Today, both organizations and their customers are demanding the ability to efficiently and conveniently manage, analyze accounts and have visibility into account initiations and modifications, detailed account history, orders, disputes and payments.

eaSuite – The Proven Platform for Online Account Management

edocs has developed the most comprehensive and sophisticated software platform and services for Online Account Management. Organizations that deploy edocs Online Account Management solutions increase productivity and improve profitability by:

- Reducing support and processing costs
- Improving operational effectiveness

- Increasing customer satisfaction
- Enhancing marketing effectiveness

The eaSuite has been deployed at organizations worldwide in some of the most complex business and rigorous technical environments, and has proven to be the industry's most scalable, standards-based platform with a total cost of ownership lower than any competing packaged or custom developed solution.

eaDirect™ is the platform product of the eaSuite that creates the core infrastructure of enterprise Online Account Management solutions. Organizations use eaDirect to access customer account data and develop ways of presenting the data for viewing, analysis and modification online. By providing business and consumer customers with online account information and self-service capabilities, eaDirect enables organizations to reduce support and processing costs, improve operational effectiveness, and increase customer satisfaction.

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

eaService™ reduces call center management costs and increases customer satisfaction by providing critical account data access to customer service representatives (CSRs). With eaService, a contact center is able to provide better service by more efficiently resolving customer account inquiries at the point of customer contact. Through its intuitive browser interface, eaService allows CSRs to take advantage of what are generally thought of as customer-facing online capabilities of an Online Account Management system, to draw out account information.

eaMarket™ is the personalization, campaign and content management solution that enables organizations to increase revenue and improve customer satisfaction by weaving personalized marketing and customer service messages throughout the Online Account Management experience. eaMarket leverages the transactional account data that is the foundation of an Online Account Management system to present personalized relevant marketing and customer service messages to customers.

eaPost® is the account content distribution system that provides an organization's customers with the convenience of All Portals Addressable™ access to account information. eaPost handles all the complexities of securely distributing summary account information to any endpoint while also bringing customers back the organization's Website to manage and control their online experience.

About This Guide

eaPay plugs into eaDirect 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 eaPay in the Command Center, and how to define payment gateways. It also describes how to use eaPay in a production environment.

Related Documentation

Online Help for command center functions, and a PDF version of this guide are also available.

Online	How to Access
Help	Select Help from eaPay command center screens.
A PDF of this guide	A PDF of this guide is available on the eaPay product CD-ROM.

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

Print Document	Description
eaPay Installation and Configuration Guide: For the Windows Operating System	How to install and configure eaPay on a Windows system.
eaPay Installation and Configuration Guide: Sun Solaris Operating Environment™ Software and the BEA WebLogic® Server	How to install and configure eaPay on a Sun Solaris system using the BEA WebLogic application server.
eaPay Installation and Configuration Guide: Sun Solaris Operating Environment™ Software and the IBM WebSphere® Application Server	How to install and configure eaPay on a Sun Solaris system using the IBM WebSphere application server.
eaDirect Production Guide	Contains instructions for creating applications and jobs, publishing design files, and managing the production environment. It also provides sample enrollment procedures for enabling customers to view and pay bills online, as well as perform a number of other online-bill payment transactions.

The eaSuite products eaDirect, eaPost, eaMarket, and eaXchange provide their own documentation.

If you need help

Technical support is available to customers who have valid maintenance and support contracts with edocs. Technical support engineers can help you install, configure, and maintain your edocs application.

To reach the U.S. Service Center, located in Natick, MA (Monday through Friday 8:00am to 8:00pm EST):

Telephone: 508.652.8400

Toll Free: 877.336.3362

e-support: support.edocs.com. This requires a one time online registration.

E-mail: support@edocs.com

When you report a problem online, by email, or by telephone, please be prepared to provide us the following information:

- What is your name and role in your organization?
- What is your company's name?
- What is your phone number and best times to call you?
- What is your e-mail address?
- In which edocs product did a problem occur?
- What is your Operating System version?
- What were you doing when the problem occurred?
- How did the system respond to the error?
- If the system generated a screen message, please send us that screen message.

- If the system wrote information to a log file, please send us that log file.
- If the system crashed or hung, please tell us.

Do you have an idea for something we could improve? Don't hesitate to send us e-mail at support@edocs.com

Overview of eaPay

1

eaPay™ is the electronic payment solution that decreases payment processing costs, accelerates receivables and improves operational efficiency. eaPay 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.

eaPay 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.

eaPay 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 eaPay retrives an invoice from eaDirect, it keeps it in the payment 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 GUI's

  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 eaDirect™ 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

What's New in eaPay 3.0?

A/R File Generation - eaPay now can export payment related information to any file format, which is controlled by an XLST template.

Line Item Reversals - eaPay can now credit accounts to support line item dispute.

SDK - Key APIs are exposed to enable third-party customization, with the purchase of the SDK for eaPay.

Job Names Changed - All eaPay jobs in the command center are now prefixed with "pmt" to make them easier to identify.

Payment Transaction Manager and Payment Cartridges

eaPay 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.

eaPay 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. eaPay 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. eaPay 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 eaPay must also have access to the Internet from the production environment. CheckFree also uses PGP file encryption for security; you must manage PGP encryption.

eaPay Jobs

Payment jobs transfer information between a biller and a payment gateway, and perform general maintenance tasks associated with an online billing system. eaPay jobs are configured in the Command Center.

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

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

eaPay offers a number of jobs that are used to create payment applications. These jobs can be selected and configured in the Command Center.

The following payment jobs are provided:

Job Type	Description
pmtAllCheckTasks	Runs all the payment jobs sequentially, in the required order.
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.
pmtConfirmEnrollment	Activates pending payment accounts after three days (by default), as long as there has been no error returned. Applies to ACH only.
pmtCreditCardSubmit	Submits scheduled credit card payments to a credit card gateway.

Job Type	Description
pmtCustom	This job must be customized using the procedures described in the ea*SDK: Customizing and Extending eaPay*.
pmtNotifyEnroll	Sends email notification to customers about the status of their payment account activation, plus any changes to their payment account.
pmtPaymentReminder	Sends payment reminder email notifications to customers.
pmtSubmitEnroll	Submits enrollment information to a payment gateway. Currently this only applies to the ACH payment gateway.

To configure a Payment Job Type, click **`Configure Job and Continue`** on the Create New Job page in the Command Center. See Configuring eaPay Jobs for an explanation of the configurable parameters associated with each payment Job Type. See the *eaDirect Administration Guide* for information about managing jobs in the Command Center.

Scheduling Payment Jobs

Schedule payment jobs to run when there is not much customer activity, for example, 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 eaDirect that creates a chain of jobs. Both ensure that no two jobs will run at the same time. See the section *Configuring eaPay Jobs* for more information.

For check jobs, run pmtRecurPayment, 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.

Payment 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 EaPay Settings for each payment gateway.

Payment Job Plug-In

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

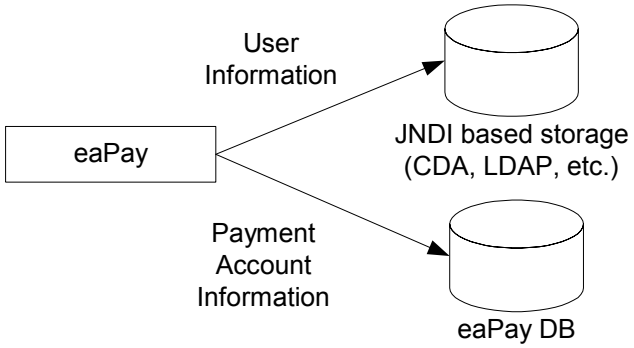
- pmtCheckUpdate
- pmtPaymentReminder
- pmtRecurPayment

Enrollment 2

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 eaDirect/eaPay 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-DDN / Multiple-DDN

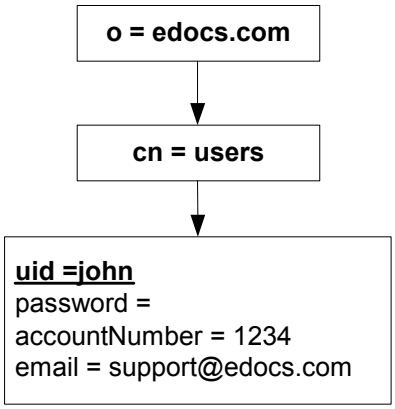
Choose single or multiple Data Definition Name (DDN, also referred to as biller or statement provider) based on whether the user will have only one biller with one account, or multiple billers with multiple accounts. The following sections describe the DDN models:

Single-DDN Model

This model is demonstrated by the eaPaySimple sampleapplication, which is packaged as ear-eapay-simple.ear and can be accessed as http://host:port/eaPaySimple/Payment?app=Payment&ddn=ddn_name, where ddn_name is the DDN name defined through the Command Center.

In this model, a single user can have only one account number per DDN. This model is compatible with the eaDirect eaSample application. Therefore, a user who enrolls through eaSample should be able to login to eaPaySimple, and vice versa.

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

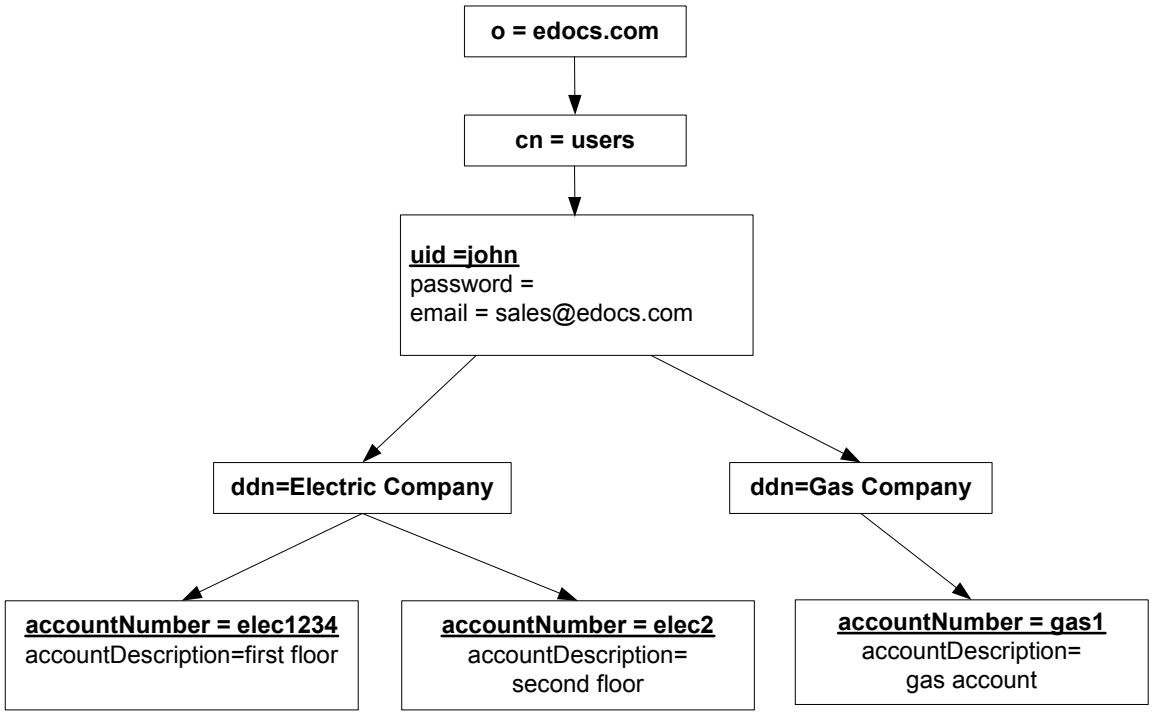


Multiple-DDN Model

This model is demonstrated by the eaPayComplex sample application, which is packaged as ear-eapay-complex.ear and can be accessed as http://host:port/eaPayComplex/Payment?app=Payment.

In this 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 eaPayComplex implementation is based on eaDirect CDA enrollment, and its CDA schema is shown the following diagram:



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

eaPay Enrollment

eaPay requires that an existing user be enrolled with eaDirect. eaDirect presents bills or account statements to that user, and eaPay provides the ability to make payments against those accounts. eaPay 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 eaPay 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. eaPay 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 pmtConfirmEnrollment 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 pmtConfirmEnrollment 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.

Check Payments 3

Check Payment Overview

Check PaymentseaPay 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 eaPay 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. eaPay 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 will not send a *return* file, or any other form of acknowledgement.

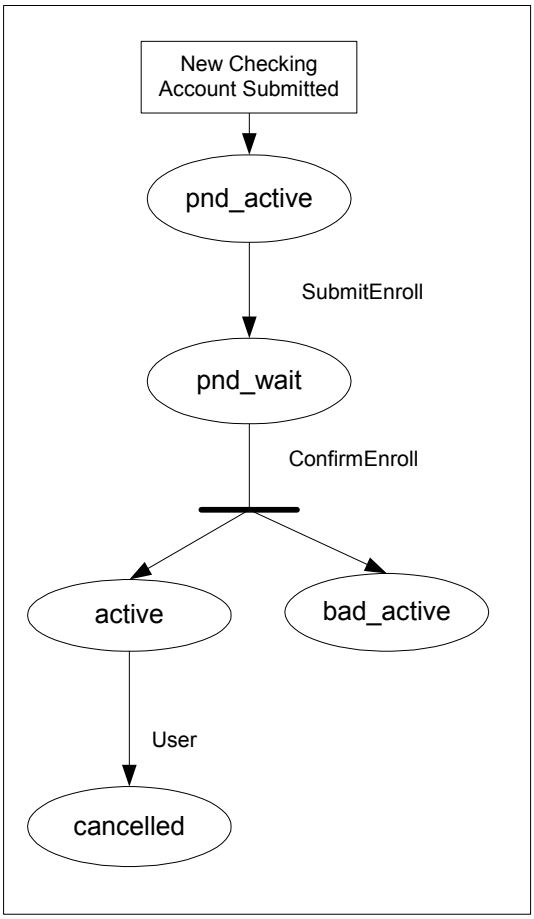
4. The pmtConfirmEnrollment job runs. This job updates the status of the customer enrollment status to "active "if there are no problems after a specified number of days (by default, three days).

   If the payment enrollment information is not correct, the pmtConfirmEnrollment 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.

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.

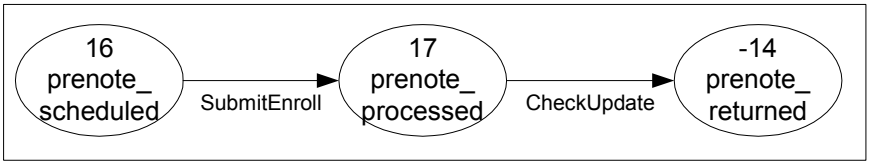


The following table describes each status:

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.

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.

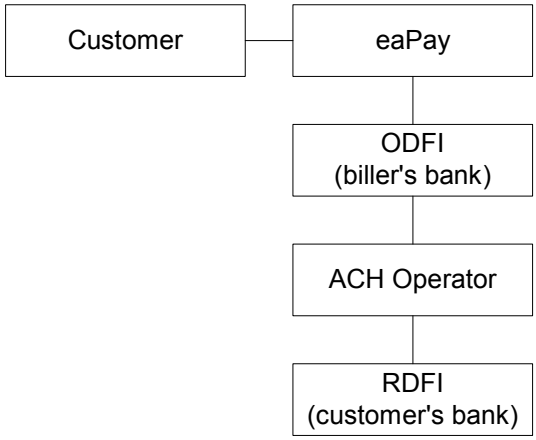


Transaction Status	Description
prenote_scheduled (16)	eaPay scheduled a prenote, and it is ready to send to ACH.
prenote_processed (17)	eaPay processed a prenote and sent it to ACH.
prenote_returned (-14)	ACH returned a prenote.

See *Check Payment Transactions* for more information about a check payments.

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. eaPay 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 eaDirect/eaPay 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 eaPay.

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.

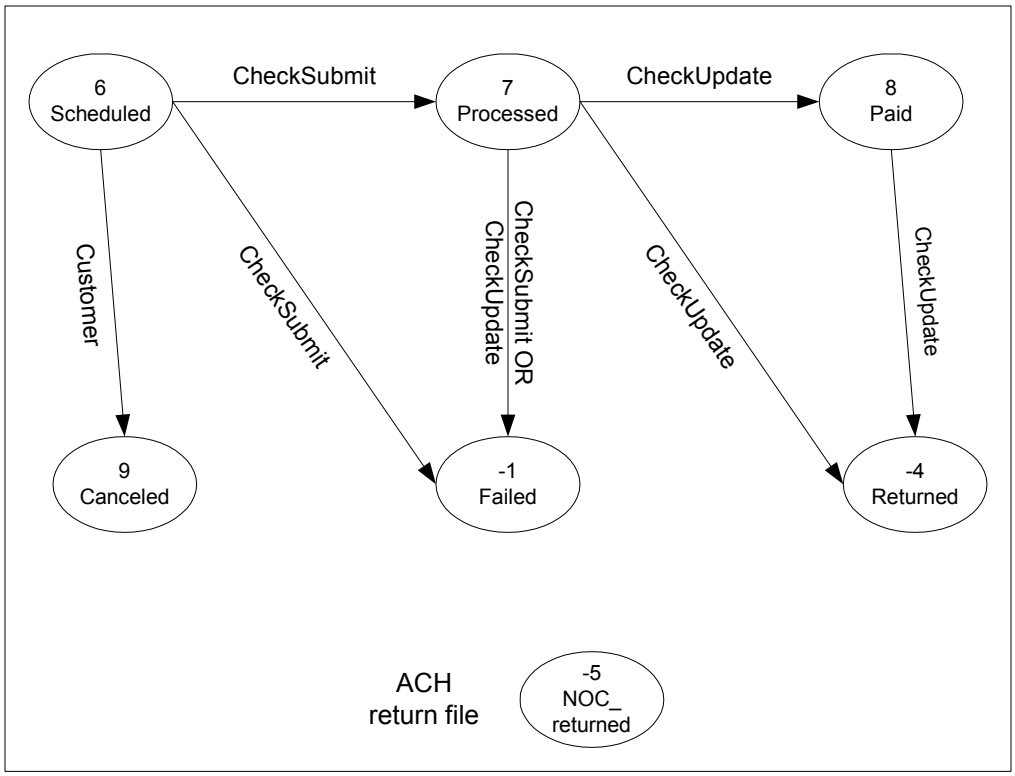
   If there is an error other than "returned", pmtCheckUpdate changes the check status to "failed".

   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 eaPay 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 payment database.

Transaction Status	Description
Scheduled(6)	A customer scheduled a new check payment.
Processed(7)	eaPay 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.

User Options

The user interface you create for eaPay can offer a variety of check payment options. Some of those options require you to configure fields in eaPay 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. See*Recurring Payments* for more information.

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". eaPay creates the debit file; the other files are created by CheckFree and returned to eaPay.
- **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 eaPay'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 eaPay.

The following directory structure shows an example setup:

/payments/out/bd - contains output files produced by eaPay

/payments/out/ftp - contains copies of the output files produced by eaPay

/payments/out/history - a work directory that contains PGP-encrypted versions of the output files produced by eaPay, 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 eaPay

Incoming Files

The steps required to process incoming files are:

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 will now be 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.

Outgoing Files

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

The file types and contents are:

- **Confirmation**(*confirm*) - This file contains 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*) - This file contains a more detailed summary of the debit file. eaPay does not process this file.
- **Transaction Journal** (*tranjrnl*) - This file contains details for each record received in the debit file, and includes a status for each transaction.

- **Settlements** (*setlment*) - This file contains details on expected deposit amounts, listed by payment date. This file is not processed by eaPay.

- **Returns** - This file contains information on any transactions that were returned by the bank, usually due to insufficient funds.

CheckFree will return a payment with one of the following statuses:

- **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 eaPay configuration is incorrect. These type of transactions will be 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 successfully made. Their convention is that if a transaction has not been returned within five business days, then it is assumed to have completed successfully.

ACH

ACH Change Codes

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

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
R04	Invalid Account Number
R05	Reserved

Code	ACH Return Code Description
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 eaPay 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. eaPay 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*.

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

eaPay 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 edocs Professional Services or your development team for more information about supporting ACH addenda records.

Payment Portal Feature

eaPay can act as a payment portal to host the payments from different billers. For ACH, eaPay can put checks from different billers (DDNs) into one ACH file, and can process a return file with checks from different DDNs. See the pmtCheckSubmit and pmtCheckUpdate jobs for details.

Credit Card Payments 4

Credit card payments are supported for immediate or future (scheduled) payments. Credit card payments require two steps: authorization and settlement. Authorization verifies the customer account and puts a hold on the account for the amount of the payment. Settlement occurs when the payment is actually made. eaPay 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 eaPay, which requires signing up with VeriSign Payment Services. An edocs Professional Services representative can walk you through that process. In addition, other cartridges can be created by edocs Professional Services to support other payment processors.

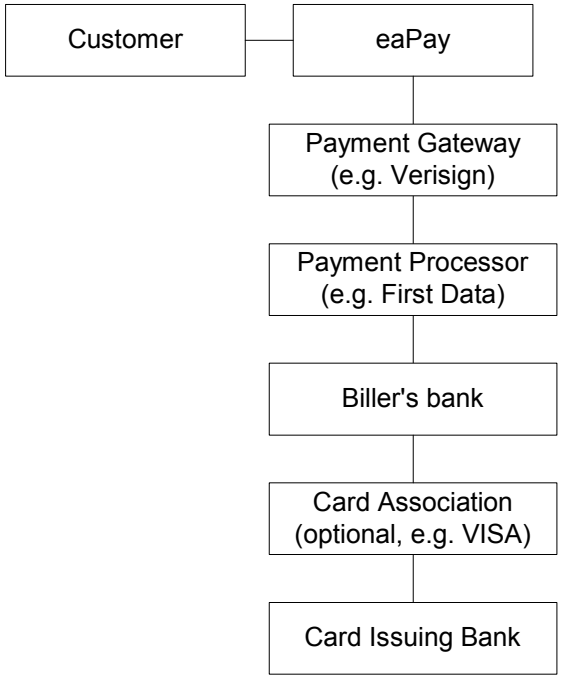
Credit Card Payment Status

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 payment 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	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 that it will be tried again. There is no need for eaPay to retry an instant payment; the user will see the error message and optionally retry payment.

Credit Card Payment Transactions

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



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. eaPay sends the payment to credit card cartridge in real time.

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

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

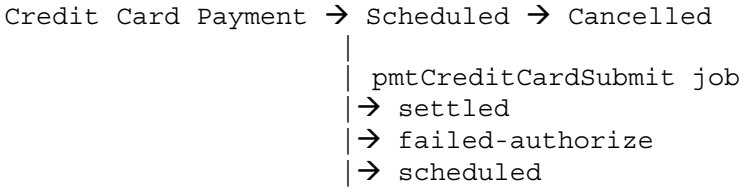
   If there is a network problem, the state is set to "failed".

4. The card is inserted into *creditcard_payments* table.

5. The result of the transaction is presented to the user.

Scheduled Credit Card Payments

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



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.

   Before the scheduled credit card payment is processed by pmtCreditCardSubmit, the user can modify or cancel it.

2. 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 eaPay 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 *pmtCreditCardSubmit Job*. For information about configuring job plugins, contact Professional Services.

3. 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.

4. 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.

User Options

The user interface to eaPay can offer a variety of credit card payment options. Some of those options require that fields be configured in eaPay Settings for a credit card payment gateway. See *Credit Card Payment Gateway* under *Configuring Payment Gateways* for more information about configuration.

Using VeriSign as a Payment Gateway

A cartridge for VeriSign is provided with eaPay. 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 eaPay Settings page will have 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 eaPay.

AVS (Address Verification Service)

eaPay sends the address to the card issuer. If the address doesn't match, the payment gateway returns an AVS code, which is ignored by eaPay. This must be implemented by a plug-in. Writing a plug-in is described in the *eaSDK: Customizing and Extending eaPay* document that comes with the eaPay SDK.

The address is optional and won't affect whether the payment is accepted or not. However, using an address may get a lower rate from card issuer. eaPay could be enhanced for a future release of eaPay to process AVS code.

Recurring Payments 5

Overview

eaPay 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 (eaPay) 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 five 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 the eaDirect Composer. The name and format of the (minimum) amount due must be specified in the eaPay Settings section of 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 the eaDirect Composer. The name and format of the due date must be specified in the eaPay Settings section of 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 will be made at 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 will be 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 recurring payment will be deactivated. By default, a recurring payment will only start 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.

There is also a script that can be run after installation that prevents a bill from being payed twice. For more information about that script, see the *eaPay Installation Guide* for the platform you are running.

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.

The next section provides examples for the first four recurring payment types. The section after that explains how to test those payment types.

Recurring Payment Transaction Cycle

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

Recurring payments can support only one customer account per biller. Recurring payments do not support multiple customer accounts with a single biller.

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

1. pmtRecurPayment contacts eaDirect 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 eaDirect 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 eaDirect, which means there is no bill information associated with this recurring payment.

2. pmtRecurPayment schedules payments (inserts a payment with status of "scheduled "in the *check_payments* or *creditcard_payments* table so that the payments will be processed. This process is called **scheduling**. A payment will be 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 pmtRecurPayment 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.

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 pmtRecurPayment job. See the section about the pmtRecurPayment job for more information about how a payment is made.

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 eaDirect tables and saved into the *payment_bill_summaries* table. After the pmtRecurPayment 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. The *recurring_payments* table is updated with the *payment_id*.

Recurring Payment Examples

The first four cases of recurring payment are described next, 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 will only be deactivated when the pay date is after end date

2. The eaDirect 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 pmtRecurPayment 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 eaDirect 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. pmtRecurPayment 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 eaDirect, pmtRecurPayment 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 paid. In this case, since there is no last payment, the bill bill3 will be 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 pmtRecurPayment runs between 04/11/2001 and 05/10/2001, nothing happens to this recurring payment because synchronization and scheduling will not happen. The table remains unchanged.

7. On 05/11/2001 11:59:00PM, three days before *next_pay_date*, pmtRecurPayment runs again. The recurring payment mentioned previously won't be synchronized, because it's *bill_scheduled* is "N". However, it will be scheduled. pmtRecurPayment 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
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, pmtRecurPayment 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 will be 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 pmtRecurPayment job runs again. It contacts eaDirect 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 extra 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

Column Name	Value
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. pmtRecurPayment 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.

Column Name	Value
last_pay_date	01/01/1970; unchanged.
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*, pmtRecurPayment 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 will be changed to "inactive".

If no bills arrive for a month, then *next_pay_date* will be automatically moved to next month. For example, if there is no bill for April, then the *next_pay_date* will be 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
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 pmtRecurPayment 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 due date, 05/15/2001.
bill_id	bill3
curr_num_payments	0

3. On 05/11/2001, three days before *next_pay_date*, pmtRecurPayment 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 will be changed 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

Column Name	Value
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*, pmtRecurPayment 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 will be changed to "inactive".

Scheduling Payment Jobs

We recommend that you schedule payment jobs to run during periods of low customer activity, for example. 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 eaDirect to create a chain of jobs, both of which ensure that no two jobs will run at the same time. See the section pmtAllCheckTasks for more information.

There is no strict ordering of payment jobs. For check jobs, we recommend that you run pmtRecurPayment, 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 eaDirect being indexed. In order to ensure that recurring jobs run successfully for all users, it is recommended that the indexer job run before the recurring payment job.

Payment 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 eaPay Settings.

Payment Job Plug-In

Some payment jobs support plug-ins to extend core payment functionality. See the *eaSDK: Customizing and Extending eaPay* for more details.

To Configure Recurring Payments

Recurring payments are configured by the pmtRecurPayment job.

Recurring payments can be configured to provide email notification when a payment is scheduled by eaPay, 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 the pmtRecurPayment job for details about 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 edocs *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 will cause incorrect behavior.

The following example test sequences validate the operation of Recurring Payment. 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 onn the pmtRecurPayment 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 will see 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 pmtRecurPayment 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 will 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 thatf the pmtRecurPayment 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 10^{th} 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 pmtRecurPayment job. The payment should be shown as "scheduled", if the “number of days before pay date to schedule the payment” on the pmtRecurPayment 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 pmtRecurPayment 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 pmtRecurPayment 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 will 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 pmtRecurPayment 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 pmtRecurPayment task. The payment should be shown as "scheduled", if the `number of days before pay date to schedule the payment` on the pmtRecurPayment 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. There will be no record 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.

Email Notifications

6

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
pmtRecurPayment	Sends email notification when a payment is scheduled by a recurring payment, and when the recurring payment expires.
PmtReminder	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.
All Jobs	Email can also be sent to the **eaPay administrator** about 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.

The email message format and content is controlled by the email template files, the path to which is defined by the `Email template for job status notification` parameter in eaPay Settings. The default templates are stored in *<PaymentHomeDir>\lib\payment_resources* (for Windows) or *<PaymentHomeDir>/lib/payment_resources* (for Unix). The default email template files must be customized, which is usually done by edocs Professional Services during installation. Contact edocs Professional Services or your development team for more information.

Email Template Customization describes the template files and describes the variables available for use in template files.

Configuring Payment Gateways

7

Configuring a Payment Gateway

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

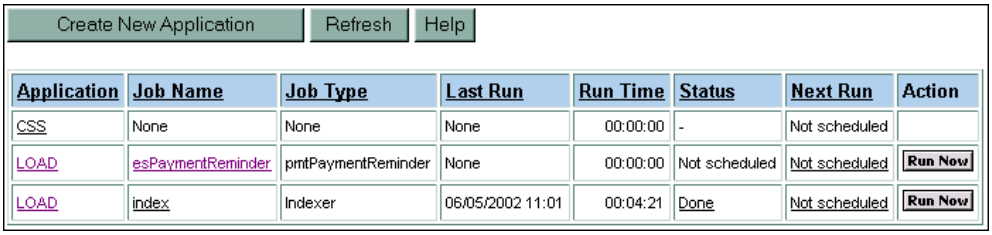
Configuration information that is specific to a payment gateway must be provided by the payment gateway or by the biller's bank. This information should be in hand before configuring.

To configure a payment gateway:

1. Access the Command Center by pointing your web browser to the virtual directory on the Web Server (<*server_name*>/*eaDirect*). Log on using the **`admin`** username and its password (the default is "edocs).

2. From the Command Center menu, click **`Settings`**. The Settings page appears.

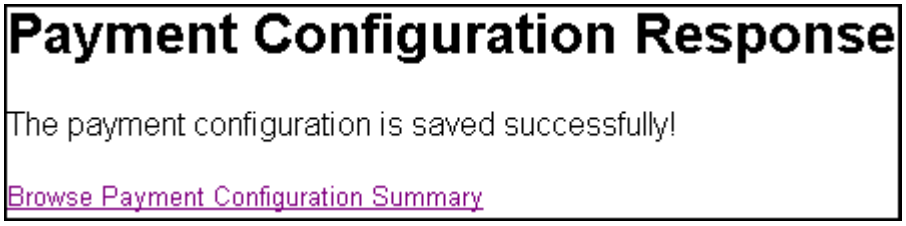
3. Click the **`eaPay Settings`** tab. A Browse Payment Configuration page appears showing the name of the DDN (Application) and Payment Types that can be configured.



4. Select a DDN (Application) and Payment Type and click **`Create`**. The Create New Payment Configuration page appears listing payment configuration forms for the payment type you have chosen. Only **one** payment gateway can be configured at a time. See the following sections for descriptions of the configuration parameters for each type of payment gateway.

5. The type of configuration screen you will see depends on the Payment Type: check or credit card. See the topics on those payment gateway types for information about configuring them.

6. After configuring the payment gateway, click **`Create`**. You will see the following message:

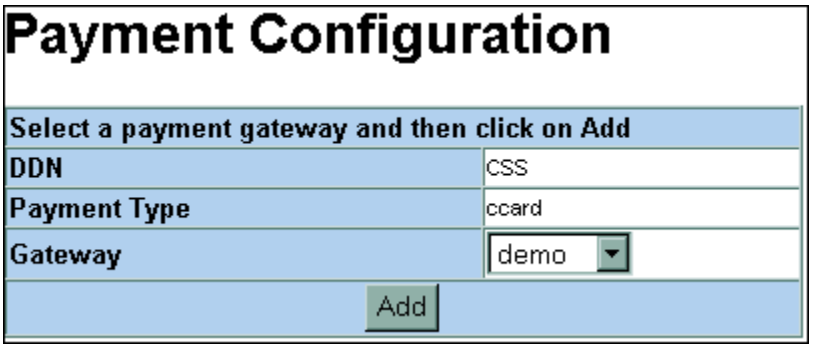


7. After you have configured a payment gateway, you should return to the Main Console page and begin the process of creating jobs for the payment applications you want to run.

When creating jobs for eaPay, you must select a Job Type (also referred to as a *task*) and define configuration parameters for it. To access a job configuration page for a selected Job Type, click **`Configure Job and Continue`** on the **`Create New Job`** page in the Command Center. Then, configure the job as necessary.

Check Payment Gateway

If you chose ACH or CheckFree for the payment gateway type, you will see the following configuration screen:



The first two entries in the Payment Configuration page list the DDN (also referred to as Biller or eaDirect Application) and Payment Type you chose to create for this payment gateway. When you choose the Gateway from the drop-down list on this form, the parameters for the selected gateway type will be displayed on a new screen.

As part of the check payment gateway configuration process, you must manually create inbound and outbound directories to store transaction files that will be sent to and received from each payment gateway. Configuration requires the full pathnames of those directories. These directories are used to store payment transaction files that are sent to and received from the payment gateway. eaPay **does not** automatically create the directories for you when you save the payment configuration.

Select a **`Gateway`** from the drop down list and complete the form as required. The following tables describe the fields on the forms for the ACH and CheckFree payment gateways:

ACH Gateway Parameters

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

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

Gateway - The type of payment gateway: ACH.

The following parameters are shared by all DDNs:

Encrypt Check Account Number - **`Y`** causes eaPay to encrypt check account numbers in the eaDirect/eaPay database.

The following parameters are shared by both the credit card cartridge (if any) and check cartridge used by this DDN:

Batch Size for Payment Reminder Table - Specifies the number of payment reminders to be read into memory from the payment database for the pmtReminder job. Note that specifying a batch size that is too small will increase 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.

Name of Due Date in eaDirect 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 eaDirect database, during data processing, for use in eaPay. Please see the *eaDirect Production Guide* for details.

This field only applies to recurring payments. If you will not be 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 eaDirect 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 eaDirect database, during data processing, for use in eaPay. Please see the *eaDirect Production Guide* for details.

This field only applies to recurring payments. If you will not be 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 eaDirect Index Table - The name of the variable defined by the DefTool for minimum amount due, which must be indexed by eaDirect. 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.

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 eaPay jobs that support job status notification. Additional email information is specified in the following fields.

Mail server for job status notification - The name of the mail server to be used for job status notification.

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.

Email template for job status notification - The path to the email template to be used to format the email used for job status notification.

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 will handle getting eaDirect user information, which is determined by the type of enrollment supported. The options are:

com.edocs.payment.payenroll.usracct.JNDISingleDDNUserAccountAccessor, for when single DDN eaDirect user information is stored in CDA.

or

com.edocs.payment.payenroll.usracct.JNDIMultipleDDNUserAccountAccessor, for when multiple DDN eaDirect user information is stored in CDA.

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

Implementation of IPaymentAccountUserAccessor - The name of the class that will handle getting eaPay user information. The options are:

com.edocs.payment.payenroll.payacct.SSOPaymentAccountAccessor, for when eaPay user information is stored in CDA (not recommended).

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

The following parameters are specific to check payment gateways:

Batch Size for 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 will increase 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.

Days to Clear Checks - 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 will be rejected for the causes stated in the prenote file. The default is three days.

Update eaPay enrollment in Case of NOC - "Y" causes the pmtCheckUpdate job to update enrollment status when a NOC is returned. "N" means that this value will be 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 will not receive email when a NOC is returned.

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

Generate empty ACH file when there are 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.

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 ID - The routing number 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 Batch Header record's Company ID 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.

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

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

ACH Template Directory - Directory where ACH XML files are stored. The default for Unix is: *$EAPAY_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 edocs 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 eaPay 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

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

The following table describes the configurable parameters for a (CheckFree) CDP payment gateway:

Gateway - The type of payment gateway, either CDP or ACH. You can change the type of payment gateway here, which refreshes the screen with the fields for the new gateway type.

Batch Size for Payment Reminder Table - Specifies the number of payment reminders to be read into memory from the payment database as a batch job. Note that specifying a batch size that is too small will increase 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.

Name of Due Date in eaDirect 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 eaDirect database, during data processing, for use in eaPay. Please see the *eaDirect Production Guide* for details.

This field only applies to recurring payments. If you will not be 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 eaDirect Index Table - The name of the field you have defined to extract the Amount Due from each statement. This Field **must** be routinely indexed in the eaDirect database, during data processing, for use in eaPay. Please see the *eaDirect Production Guide* for details.

This field only applies to recurring payments. If you will not be 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 eaDirect Index Table - The name of the variable defined by the DefTool for minimum amount due, which must be indexed by eaDirect. 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.

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 eaPay jobs that support job status notification. Additional email information is specified in the following fields.

Mail server for job status notification - The name of the mail server to be used for job status notification.

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.

Email template for job status notification - The path to the email template to be used to format the email used for job status notification.

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

com.edocs.payment.payenroll.usracct.JNDISingleDDNUserAccountAccessor, for when single DDN eaDirect user information is stored in CDA.

or

com.edocs.payment.payenroll.usracct.JNDIMultipleDDNUserAccountAccessor, for when multiple DDN eaDirect user information is stored in CDA.

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

Implementation of IPaymentAccountAccessor - The name of the class that will handle getting eaPay user information. The options are:

com.edocs.payment.payenroll.payacct.SSOPaymentAccountAccessor, for when eaPay user information is stored in CDA (not recommended).

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

Batch Size for Payment Table - Specifies the number of scheduled checks to be read into memory from the payment database for the pmtCheckSubmit and pmtCheckUpdate jobs. Note that specifying a batch size that is too small will increase 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.

Days to Clear Checks - 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 will be rejected for the causes stated in the prenote file.

Encrypt Check Account Number - Determines whether to encrypt check account numbers in the eaDirect/eaPay database.

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 eaDirect 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 that 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 Path - Directory where pmtCheckSubmit will create debit files.

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

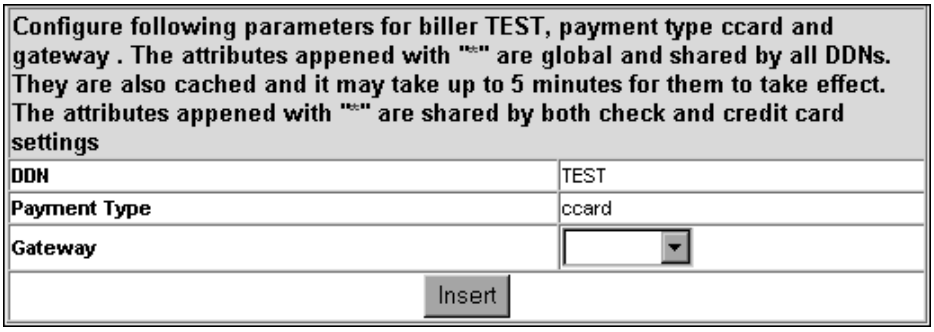
Template File Directory - For a standard installation, should be:

/opt/EDCSbd/payment_resources/ checkfree/cdp/template (Unix)

C:\EDCSpay\lib\payment_resources\checkfree\cdp\template (Windows).

Credit Card Payment Gateway

If you chose to create a credit card type payment gateway, you will see the following screen, where you choose the type of credit card gateway from the list of available credit card processors:



Additional credit card processor support can be added. Contact Professional Services for more information.

This section describes the configurable parameters for a Credit Card Payment gateway using the VeriSign payment processor:

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 shared by all DDNs:

Encrypt credit card number - Specifies whether to encrypt credit card information in the eaDirect/eaPay database. This is highly recommended if you choose to save the entire credit card number, which is specified in the previous configuration parameter.

The following parameters are shared by both the credit card cartridge and check cartridge (if any) used by this DDN:

Save Full Credit Card Account Number in payment DB ("N" means only save last four digits)- Specifies whether to save the entire credit card account number in the database, or just the last four digits of the account. This should be decided by the relationship with the credit card processor.

The following parameters are shared by both the credit card and check cartridges for this DDN:

Batch Size for Payment Reminder Table - Specifies the number of credit card reminders to be read into memory from the payment database for the pmtPaymentReminder job. Note that specifying a batch size that is too small will increase 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.

Name of Due Date in eaDirect 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 eaDirect database, during data processing, for use in eaPay. Please see the *eaDirect Production Guide* for details.

This field only applies to recurring payments. If you will not be 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 eaDirect 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 eaDirect database, during data processing, for use in eaPay. Please see the *eaDirect Production Guide* for details.

This field only applies to recurring payments. If you will not be 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 eaDirect Index Table - The name of the variable defined by the DefTool for minimum amount due, which must be indexed by eaDirect. 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.

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 eaPay jobs that support job status notification. Additional email information is specified in the next few fields.

Mail Server for Job Status Notification - The mail server(s) to be used to send eaPay job status notification emails. This is usually the name or IP address of the SMTP server, or the name of the Microsoft Exchange server. If using multiple mail servers, separate the names by semi-colons.

Mail-to Addresses for Job Status Notification - List the email addresses of the people to whom job status notification email should be sent by eaPay jobs that support this feature.

Email Template for Job Status Notification - Path to the template used to create the job status notification email. The example template for Unix is in **<EDOCS_DIR>***/lib/payment_resources/notifyPaymentTask.txt*.

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 will handle getting eaDirect user information, which is determined by the type of enrollment supported. The options are:

com.edocs.payment.payenroll.usracct.JNDISingleDDNUserAccountAccessor for when single DDN eaDirect user information is stored in CDA.

or

com.edocs.payment.payenroll.usracct.JNDIMultipleDDNUserAccountAccessor, for when multiple DDN eaDirect user information is stored in CDA.

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

Implementation of IPaymentAccountAccessor - The name of the class that will handle getting eaPay user information. The options are:

com.edocs.payment.payenroll.payacct.SSOPaymentAccountAccessor, for when eaPay user information is stored in CDA or LDAP.

Batch Size for Credit Card Payment Table - Specifies the number of scheduled credit card payments to be read into memory from the payment database for the pmtCreditCardSubmit job. Note that specifying a batch size that is too small will increase 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.

The following parameters are specific to a Verisign gateway:

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**.

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.

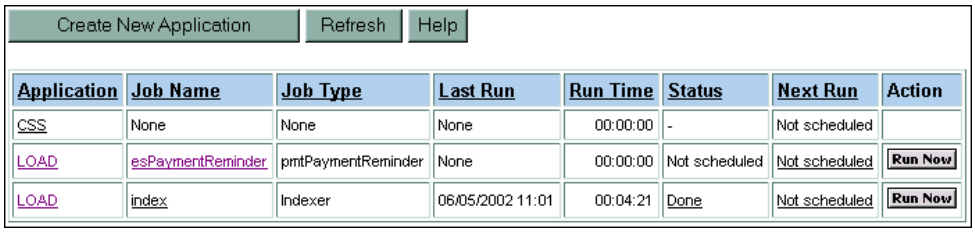
For information about implementing this class, contact edocs Professional Services. The default is:
com.edocs.payment.cassette.verisign.VeriSignCreditCardSubmitPlugIn.

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

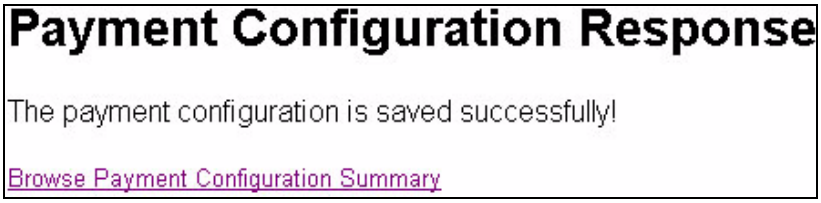
Updating a Payment Gateway Configuration

To update a payment gateway's settings:

1. Click **`Settings`** from the Command Center menu. The Settings page appears.
2. Click the **`eaPay Settings`** tab. The Browse Payment Configuration Summary page appears, indicating the name of a payment gateway in the Gateway column.



3. Click **`Create`** for a DDN (payee). The Create New Payment Configuration page appears. The configuration parameters on the update screens are the same as those used for creating a new payment gateway. See the previous sections for descriptions of those fields.
4. Click **`Save`** when you are finished. The following message is displayed:



Deleting a Payment Gateway Configuration

To delete a payment gateway configuration:

1. From the Browse Payment Configuration Screen, click **`Delete`** for a payee. A View Payment Configuration page appears.
2. Click **`Delete`**. A confirmation box appears.
3. Click **`OK`**. The following message is displayed:

Configuring eaPay Jobs

8

pmtAllCheckTasks Job

Configuring eaPay JobspmtAllChecksTasks runs the following eaPay jobs sequentially.

- pmtRecurPayment
- pmtCheckSubmit
- pmtCheckUpdate
- pmtPaymentReminder
- pmtSubmitEnroll
- pmtConfirmEnroll
- pmtNotifyEnroll

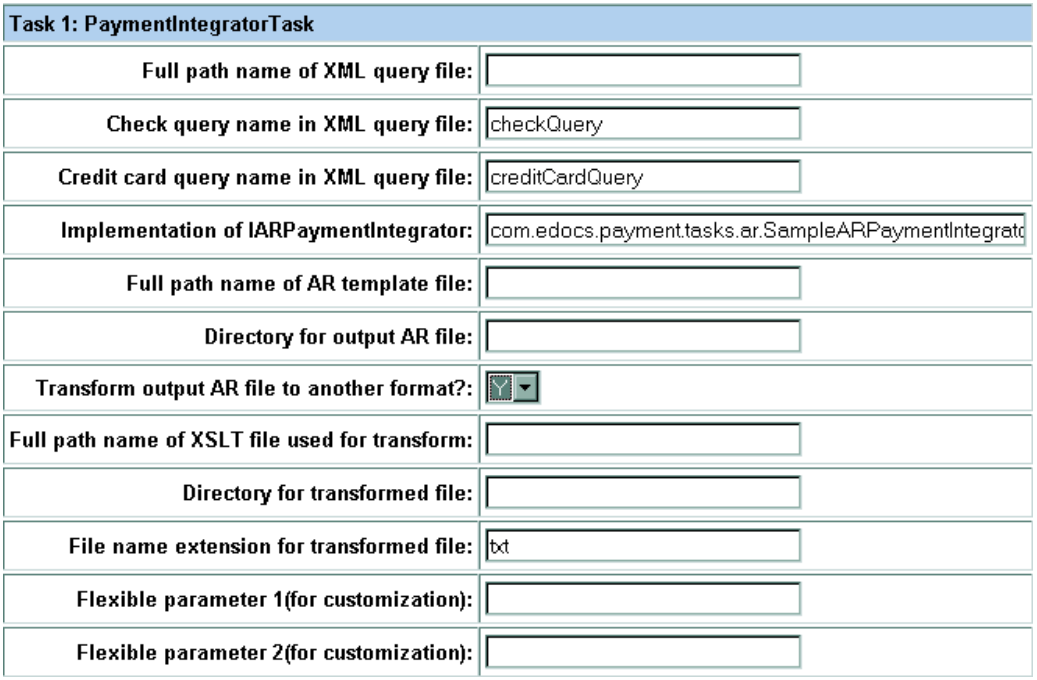
This job presents all the configuration menus from all the other jobs, so you can configure all the listed jobs from this job.

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

pmtARIntegrator Job

pmtARIntegrator uses an XLST template to translate data extracted from payment 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 of Query Template File - Enter the path to the general query template file. The default */opt/EDCSpay/lib/payment_resources/ar/ARQuery.xml*, which you can modify to fit your needs. See the example *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.tasks.payment.ar.ARPaymentIntegrator*. The `IPaymentIntegrator` interface defines a method to use eaPay to generate A/R files in a specific format. See the *SDK: eaPay Module* 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 */opt/EDCSpay/lib/payment_resources/ar/ARPaymentFlat.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. Two other sample template files are provided: *XMLPayment.xml* and *XMLPayment.xml*.

Directory for Output AR File - The directory to put the output file. The default value is */opt/EDCSpay/Output/ar*.

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.

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 XSLT 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 sample is */opt/EDCSpay/lib/payment_resources/ar/PaySample.xsl*.

After specifying the preceding parameters and scheduling the job, the A/RIntegrator job will generate 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 *SDK: eaPay Module*.

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 */opt/EDCSpay/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 that, if provided, will be used in the AR query.

Flexible Parameter 2 (for customization) - Optional parameter that, if provided, will be used in the AR query.

pmtCheckSubmit Job

pmtCheckSubmit selects scheduled check payments that are ready to pay who's 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 who's 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 *<eaPay_install_dir>/lib/payment_resources/template/ach* (*<eaPay_install_dir>\lib\payment_resources\template\ach* for Windows). See edocs 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 will change 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 eaPay database. For more information, see the *eaSDK: Customizing and Extending eaPay*, or contact edocs Professional Services.
gateway_payment_id	Populated only if you are using gateway payment id to match a check returned from ACH back to a check in the eaPay database. For more information, see the *eaSDK: Customizing and Extending eaPay*, or contact edocs Professional Services.

Scheduling and Holidays

By default, eaPay 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 a weekday, then it is a bank holiday.
- If the holiday is on Sunday, then the following Monday is a bank holiday.

- If the holiday is on Saturday, then even if the previous Friday is a Federal holiday, the holiday is not a bank holiday.

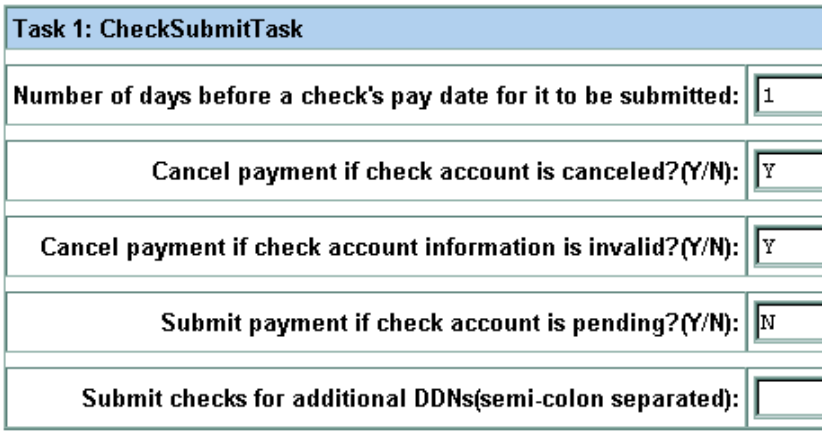
ACH is closed on bank holidays, but it is okay to send a file to ACH which requires transfer of money on holidays: ACH simply processes the checks on the next available bank business day. However, some banks require ACH files to skip bank holidays. By default, eaPay skips bank holidays.

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 eaPay 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.

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.

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 will be "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 eaPay 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.

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.

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 eaPay 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 cannot find a match, an exception will be raised and the job will fail.

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.

If pmtCheckUpdate will be processing ACH return files containing multiple DDNs, then the eaPay 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
pa#_txn_err	return ACH code
txn_message	return ACH code
pa#_status	bad_active
account_status	bad_active

Column updated	Value
pa#_txn_notify_status	“N”
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 (eaPay matches it with the *user_id* column in the *payment_accounts* table).

If the auto update flag `Update eaPay enrollment in Case of NOC` field was set to `true` ("Y") in the eaPay 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.

pa#_notify_status is set to “N” if `Notify NOC` flag is set to “Y” in eaPay Settings. If `Send Email Notification in Case of NOC` is set to "Y" in eaPay Settings, then *notify_status* is set to N". eaPay 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
pa#_txn_err	NOC_CODE::NEW_ADDENDA_INFO::OLD_ADDENDA_INFO
pa#_number	Changed for C01, C03, C06, C07
txn_message	NOC_CODE::NEW_ADDENDA_INFO::OLD_ADDENDA_INFO
account_number	Changed for C01, C03, C06, C07
pa#_routing	Changed for C02, C03, C07
routing_transit	Changed for C02, C03, C07
pa#_type	Changed for C05, C06, C07
account_type	Changed for C05, C06, C07
pa#_txn_date	current time
txn_date	current time
pa#_txn_notify_status	“N”
notify_status	“N”

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 eaPay Settings for the ACH payment gateway.

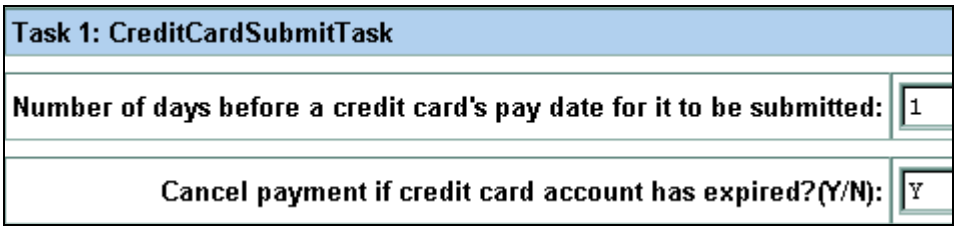
pmtConfirmEnroll Configuration

No configuration is required when creating this job.

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.

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 eaPay 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).

eaPay 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.

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 EaPay 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. eaPay recognizes the failure and keeps the payment's status as scheduled, so that the failed payments will be processed the next time CreditCardSubmit 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.

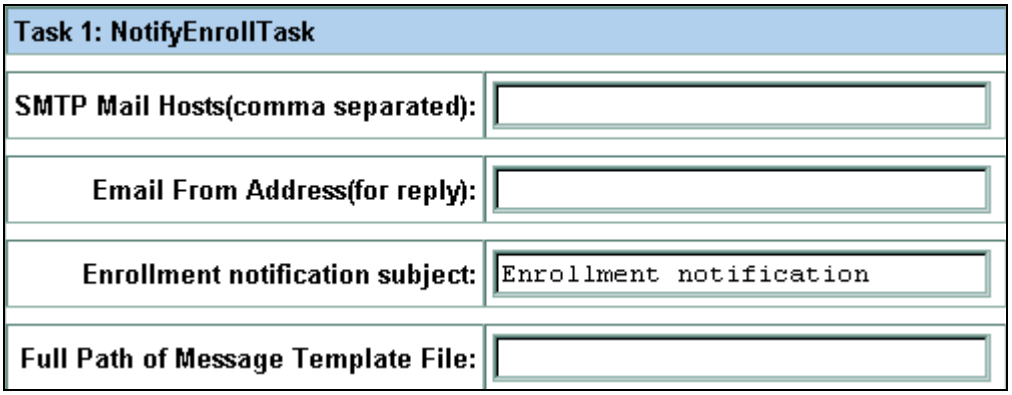
pmtCustom

The pmtCustom job can be customized to perform tasks that are not part of other eaPay jobs. See the *eaSDK:Customizing and Extending eaPay* docment 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 .

Creating a new pmtNotifyEnroll job in the Command Center of type pmtNotifyEnroll returns a dialog similar to the following:



The configurable parameters for pmtNotifyEnroll are:

SMTP Mail Hosts -Name(s) of mail host system.

Email From Address - Sender's email address.

Enrollment notification subject - Text message to appear on email subject line.

Full Path of Message Template File - Full directory path to the formatted message template file.

For Windows, this path is *%EAPAY_HOME%\lib\payment_resources\NotifyEnroll.txt*.

For Unix, the path is *$EAPAY_HOME/lib/payment_resources/NotifyEnroll.txt*.

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".

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".

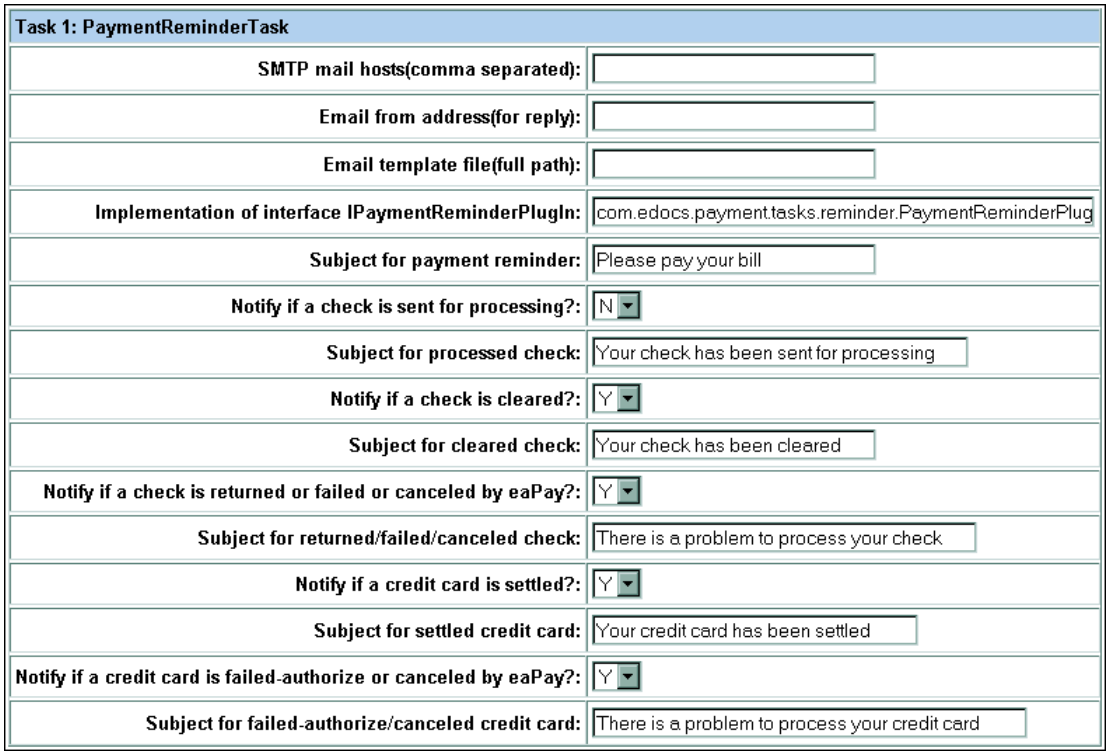
The format of the email message that is sent is generated based on the contents of an email template text file. The `Full Path of Message Template File` configuration parameter specifies the location of this template file. The default is *<install_dir>/lib/payment_resources/NotifyEnroll.txt*.

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



Task 1: PaymentReminderTask	
SMTP mail hosts(comma separated):	
Email from address(for reply):	
Email template file(full path):	
Implementation of interface IPaymentReminderPlugIn:	com.edocs.payment.tasks.reminder.PaymentReminderPlug
Subject for payment reminder:	Please pay your bill
Notify if a check is sent for processing?:	N
Subject for processed check:	Your check has been sent for processing
Notify if a check is cleared?:	Y
Subject for cleared check:	Your check has been cleared
Notify if a check is returned or failed or canceled by eaPay?:	Y
Subject for returned/failed/canceled check:	There is a problem to process your check
Notify if a credit card is settled?:	Y
Subject for settled credit card:	Your credit card has been settled
Notify if a credit card is failed-authorize or canceled by eaPay?:	Y
Subject for failed-authorize/canceled credit card:	There is a problem to process your credit card

The configurable parameters for this job are:

SMTP Mail Hosts - Name of the mail host.

The current version of eaPay **does not** support user authentication for SMTP servers. If the SMTP server requires a user name and password for sending mail, an exception must be made for email originating from the machine on which the payment Application Server is installed.

Email From Address (for reply) - Sender's email address, which the customer's email will use as Reply-To.

Email Template File (full path)- Full directory path to the formatted message template file. This template file is used to generate the body of email messages. It can be customized.

The default path to the formatted message template file for Windows is *%EAPAY_HOME%\lib\payment_resources\paymentreminder.txt*.

For Unix it is *$EAPAY_HOME/payment/lib/payment_resources/paymentreminder.txt*.

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 edocs Professional Services. The default is *com.edocs.payment.tasks.reminder.PaymentReminderPlugin*.

Subject for payment reminder - Enter the text to use for the subject line of emails sent to users to remind them that a payment is about to become due.

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

Subject for processed check - Enter the text to use for the subject line of emails sent to users to notify them that a check has been sent for processing (to make a payment).

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

Subject for cleared check - Enter the text to use for the subject line of emails sent to users to notify them that a check has cleared.

Notify if a check is returned or failed or canceled by eaPay- Indicates whether notification is to be sent for checks that were returned by the payment gateway as canceled, returned or failed settlement.

Subject for returned/failed/canceled check - Enter the text to use for the subject line of emails sent to users to notify them that a check has did not settle because it was returned, failed or was canceled. The ACH return file will explain the exact cause.

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

Subject for settled credit card - Enter the text to use for the subject line of emails sent to users to notify them that a credit card payment has settled.

Notify if a credit card is failed-authorize or canceled by eaPay- Indicates whether email should be sent for credit card payments that failed to authorize or were canceled by eaPay.

Subject for failed-authorize/canceled credit card - Enter the text to use for the subject line of emails sent to users to notify them that a credit card payment has either failed to authorize or the card has been cancelled.

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". No emails are sent for instant credit card payments. 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.

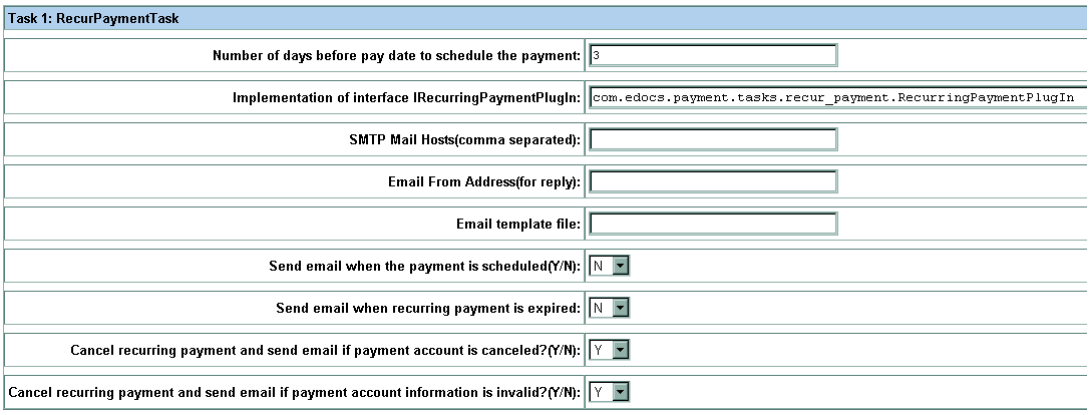
Email Template

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 *<install_dir>/lib/payment_resources* directory. See the section *Email Template Customization* for information about editing the email template to modify email content.

pmtRecurPayment Job

The pmtRecurPayment 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.

pmtRecurPayment Configuration



The configurable parameters for this job are:

Number of days before pay date to schedule the payment – The check payment will be scheduled *N* days before the pay date by pmtRecurPayment. *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.

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 pmtRecurPayment 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, 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 edocs Professional Services. The default value is: *com.edocs.payment.tasks.recur_payment.IRecurringPaymentPlugIn*, which takes no action.

SMTP Mail Hosts - List all the SMTP hosts that are available to send email notifications. This is not required if *Send email notification when the payment is scheduled* is **N**.

Email From Address (for Reply) - The email address that replies should be sent to.

Email template file - The email template file to use when creating a recurring payment notification email.

Subject for recurring payment related emails - Enter the text to use for the subject line of emails sent to users to notify them that a recurring payment has scheduled a payment.

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

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 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 eaPay as having incorrect account information. The user is enrolled, but the this 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 payment account is canceled? - Specify whether to send email to notify the user that their payment account was cancelled.

pmtRecurPayment Operation

pmtRecurPayment 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, pmtRecurPayment must query eaDirect to determine when to schedule the payment and/or how much to pay. For that reason, the pmtRecurPayment job should be run after the eaDirect Indexer job runs.

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

For additional information about how recurring payments work and the different types, see the section *Recurring Payments*.

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

No configuration is required.

eaPay Reports

9

Overview

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

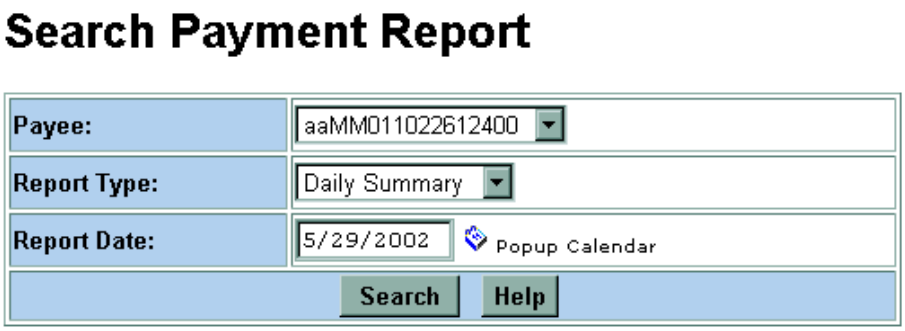
eaPay also logs additional important information: warnings and errors. Based on the format of the logging messages, eaPay can hook up with other monitor systems, such as EMC patrol.

Viewing eaPay Reports

After a payment job runs, payment reports can be viewed for entries from that job based on two report types: Daily Summary and Daily Exceptions.

To view eaPay reports:

1. In the Command Center, click **`Reporting`**. The Reporting and Log page appears.
2. Select the **`eaPay Reports`** tab. The Search Payment Report page appears.
3. Select a Payee from the drop-down menu and specify 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.



5. Click **`Search`**. Depending on the type of payment report specified in the Report Type field (Daily Summary or Daily Exception), you will see reports 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 eaPay Module Error Logs

eaPay jobs may generate error messages. eaPay 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 eaPay Module 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.
`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.

Error Message	Description
`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 payment 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 payment 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.

Error Message	Description
`DuplicateKeyException`	Indicates an exception that occurs when there are two check payments in the database with the same payment id. eaPay uses a timestamp (in milliseconds) to assign payment id's. 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 payment database. This type of error is typically corrected by the application during processing.

eaPay Administration 10

eaPay Database

Preventing Multiple Payments

By default, eaPay 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 *$EAPAY_HOME/db/set_unique_bill_id.sql* to set the unique constraint. The *bill_id* in eaPay is the same as the doc id in eaDirect.

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 eaPay 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 example interface and the eaPay command center jobs, and describes their impact on the eaPay database. This example uses the enrollment for ACH where eaPay keeps payment information in a separate database:

UI Action	Payment 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.
User enrolls	User information is inserted into CDA and payment accounts are inserted into *payment_accounts*.
Run SubmitEnroll	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 *pa#_txn_date* to the current date (yyyyMMddHHmm).
Run ConfirmEnroll	Changes *pa#_txn_status* to "bad_active" if returned, or to "active" if there is no return after three days. Updates *pa#_txn_notify_status* to "N".
Run NotifyEnroll	Finds all payment accounts whose *pa#_txn_notify_status* is "N", and send emails. Updates *pa#_txn_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.

UI Action	Payment Database
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.

UI Action	Payment Database
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 eaDirect/eaPay database will grow depends on the number of enrolled users.

The following tables describe the eaPay tables, and the eaDirect 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.

eaPay

eaPay 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 eaPay.
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_counters	< 400 per year	One row is inserted each day. Data older than one year should be backed up and deleted.

eaPay Table	Projected Row Count	Notes
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.
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.

Enrollment

Table	Projected Row Count	Notes
Ach_account (nt_ach_account on SQL Server)	100k based on 100k users	Customer dependent. There is one row for each user. This table only applies to ACH.
Afcdp_account (nt_cfcdp_account on SQL Server)	100k based on 100K users	Customer dependent. There is one row for each user. This table only applies to CheckFree CDP.

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 payment database. This will ensure proper performance for eaPay 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.

Be careful when deciding how long to keep a check in the payment 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 eaDirect/eaPay database transactions operate in their own transaction context. If a single operation fails (for example, failure to enroll or submit a payment), the eaDirect/eaPay 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 eaPay 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 edocs Professional Services.

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

Schema

The eaDirect/eaPay database schema is defined in the following files:

For Unix:

```
$EAPAY_HOME/db/create_payment_schema.sql
$EAPAY_HOME/db/alter_payment_schema.sql
```

For Microsoft Windows:

```
%EAPAY_HOME%\db\create_payment_schema.sql
%EAPAY_HOME%\db\alter_payment_schema.sql
```

Table Column Definitions

eaPay 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.

Name	Null?	Type	Description
CREATE_TIME	NOT NULL	DATE	The time when this check is created.
NEEDS_BACKUP	NOT NULL	CHAR(1)	Not used.
BILL_ID		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.
TXN_TIMESTAMP_1		DATE	Flexible date field. Can be used for customization.
TXN_TIMESTAMP_2		DATE	Flexible date field. Can be used for customization.

Name	Null?	Type	Description
PARTITION_ID	NOT NULL	NUMBER(10)	This number is used to partition the table into fixed-size buckets for performace tuning. It is configured by eaPay settings.
TXN_STATUS		VARCHAR2(20)	The transaction status returned from the payment gateway. May not be available for some gateways.
TXN_FEE		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		VARCHAR2(255)	The transaction error message returned from payment gateway. For ACH, this is the ACH return error code.
TXN_NUMBER		VARCHAR2(40)	Transaction number sent to or assigned by the payment gateway. For ACH, this is the ACH trace number.
PAYER_ACCOUNT_ NUMBER		VARCHAR2(40)	User's account number with the biller.

Name	Null?	Type	Description
MEMO		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		VARCHAR2(10)	“personal” or “business”.
CHECK_NUMBER		NUMBER(10)	Check number. Not used.
ACTION_CODE		VARCHAR2(20)	Payment gateway action code. For ACH: 27(checking debit) or 37(saving debit). For Checkfree: “ADD”
LOG_ID		NUMBER(28)	This login id points to a log id in the *payment_log* table. It associates the check with a payment report.
GATEWAY_ PAYMENT_ID	NOT NULL	VARCHAR2(255)	For Checkfree, this is the payment id assigned by Checkfree. For ACH, it matchs 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		DATE	For ACH, this is the effective batch entry date for the check.
TXN_END_DATE	NULL	DATE	Reserved.

Name	Null?	Type	Description
PAYMENT_SOURCE	NOT NULL	CHAR(1)	R/S: “S” means paid from UI, “R” means paid from recurring payment
FLEXIBILE_FIELD_1		VARCHAR2(255)	Flexible field for customization.
FLEXIBILE_FIELD_2		VARCHAR2(255)	Flexible field for customization.
FLEXIBILE_FIELD_3		VARCHAR2(255)	Flexible field for customization.
PID		VARCHAR2(255)	The unique id used to identify this payment account.
LINE_ITEM_ID		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 explains the legal check payment status and their meanings.

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. eaPay 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		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

Name	Null?	Type	Description
PAYER_ACCOUNT_ NUMBER		VARCHAR2(40)	User's account number with the biller.
CARD_HOLDER_ NAME		VARCHAR2(40)	Name on the card.
CARD_NUMBER		VARCHAR2(255)	The account number on the card. The contents depend on the payment settings.
CARD_TYPE		VARCHAR2(40)	Type of card. For example, VISA, MC, AMEX.
PID		VARCHAR2(255)	The unique id used to identify this payment account.
CARD_EXPIRE_DATE		DATE	Card expiration date.
CARD_EXPIRE_ DATE_FORMAT		VARCHAR2(40)	Format of the expiration date.
CARD_STREET		VARCHAR2(255)	Street address of card holder.
CARD_CITY		VARCHAR2(40)	City of card holder.
CARD_STATE		VARCHAR2(40)	State of card holder.
CARD_ZIP		VARCHAR2(40)	Zipcode of card holder.
CARD_COUNTRY		VARCHAR2(40)	Country of card holder.
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		VARCHAR2(20)	The transaction status returned from payment gateway. May not be available for some gateways.
TXN_ERR_MSG		VARCHAR2(255)	The transaction error message returned from payment gateway. For ACH, this is the ACH return error code.
TXN_NUMBER		VARCHAR2(40)	Transaction number sent to or assigned by payment gateway. For ACH, this is the ACH trace number.
TXN_FEE		NUMBER(28,2)	The transaction fee charged by payment gateway. May not be always available.
TXN_AUTH_CODE		VARCHAR2(40)	Transaction authentication code.
TXN_AVS_CODE		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 eaPay Settings.
LOG_ID		NUMBER(28)	ID of the summary report in the *payment_log* table.
TXN_START_DATE		DATE	Transaction start date.
TXN_END_DATE		DATE	Transaction end date.

Name	Null?	Type	Description
PAYMENT_SOURCE		CHAR(1)	Describes which eaPay function scheduled this payment. R" for recurring and "S" for single payment.
TXN_TIMESTAMP_1		DATE	For customization.
TXN_TIMESTAMP_2		DATE	For customization.
MEMO		VARCHAR2(255)	Check memo, which can be used for customization.
FLEXIBLE_FIELD_1		VARCHAR2(255)	Flexible field for customization.
FLEXIBILE_FIELD_2		VARCHAR2(255)	Flexible field for customization.
FLEXIBILE_FIELD_3		VARCHAR2(255)	Flexible field for customization.
LINE_ITEM_ID		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 exactly the same as the *creditcard_payments* table, except that the *payment_id* is no longer a primary key.

CHECK_PAYMENTS_STATUS

Describes the possible status that a check payment can have, which is stored in the *check_payments* table.

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)	Identifyies this payment account.
USER_ID	NOT NULL	VARCHAR2(40)	The user who owns this payment account.

Name	Null?	Type	Description
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.
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.

Name	Null?	Type	Description
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.

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		DATE	Bill due date.

Name	Null?	Type	Description
BILL_AMOUNT_DUE		NUMBER(28,2)	Bill amount due.
MIN_AMOUNT_DUE		NUMBER(28,2)	Minimal amount due.
PAYMENT_ID		NUMBER(28)	The payment id of the payment made against this bill
FLEX_FIELD_1		VARCHAR2(255)	Available for customization.
FLEX_FIELD_2		VARCHAR2(255)	Available for customization.

PAYMENT_COUNTERS

This table generates counters. eaPay 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 eaDirect.

Name	Null?	Type	Description
INV_ID	NOT NULL	NUMBER(28)	Unique invoice ID generated by eaPay.
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 eaPay and it can be customized.
INV_NUMBER	NULL	VARCHAR2(40)	A string assigned by the biller to identify this invoice. Not used by eaPay, 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 eaPay, so it can be used for customization.
INV_ISSUER	NULL	VARCHAR2(40)	The entity that issued the invoice. Not used by eaPay, so it can be used for customization.

Name	Null?	Type	Description
INV_MEMO	NULL	VARCHAR2(250)	Invoice memo. Not used by eaPay, so it can be used for customization.
INV_ISSUER	NULL	VARCHAR2(40)	The entity issues the invoice. Not used by eaPay, 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 eaPay, 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 eaPay, so it can be used for customization.
TRANSACTION_ DATE	NULL	DATE	Invoice transaction date. Not used by eaPay, so it can be used for customization.
FLEXIBLE_FIELD_1	NULL	VARCHAR2(255)	Flexible field. Not used by eaPay, so it can be used for customization.
FLEXIBLE_FIELD_2	NULL	VARCHAR2(255)	Flexible field. Not used by eaPay, so it can be used for customization.
FLEXIBLE_FIELD_3	NULL	VARCHAR2(255)	Flexible field. Not used by eaPay, so it can be used for customization.

Name	Null?	Type	Description
FLEXIBLE_FIELD_4	NULL	VARCHAR2(255)	Flexible field. Not used by eaPay and is customizable.
FLEXIBLE_FIELD_5	NULL	VARCHAR2 (1000)	Flexible field. Not used by eaPay, 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 eaPay Settings information entered through theCommand 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		DATE	Not used by eaPay.

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		VARCHAR2(512)	Report parameter value.
BATCH_INDEX	NOT NULL	NUMBER(38)	Payment internal use, record the batch indexes in a payment record.

PAYMENT_REMINDERS

This table records the payment reminders set by the users through Payment 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 eaPay 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”;p check or credit card.

Name	Null?	Type	Description
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
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 have been paid.
STATUS	NOT NULL	VARCHAR2(10)	"active" or "inactive": whether the recurring payment has ended.
EMAIL_IND	NOT NULL	CHAR(1)	"Y" or "N": whether to send email when amount due is more than the amount specified.

Name	Null?	Type	Description
BILL_ID		VARCHAR2(255)	DOC id of the bill being paid, if applicable.
BILL_SCHEDULED	NOT NULL	CHAR(1)	"Y" or "N": whether the current payment has been scheduled.
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		VARCHAR2(255)	Flexible field. Not used by eaPay, so it can be used for customization.
FLEX_FIELD_2		VARCHAR2(255)	Flexible field. Not used by eaPay, so it can be used for customization.
PID		VARCHAR2(255)	The unique id that identifies this payment account.

Payment indexes

The following table lists the Indexes defined on payment 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

Table name	Index name	Indexed columns
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
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 eaPay database is designed to migrate from a previous version to new version, whenever eaPay upgrades the database schema. If you have a payment database from an older version of eaPay, 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 eaPay database depends on the *document_definition_name* table from eaDirect, 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 eaPay.

Changes From eaPay 2.1 to eaPay 3.0

For eaPay 3.0, five tables have added a new column:

Table Name	New Column
CREDITCARD_PAYMENTS_HISTORY	LINE_ITEM_ID
CREDITCARD_PAYMENTS	LINE_ITEM_ID
CHECK_PAYMENTS_HISTORY	LINE_ITEM_ID
CHECK_PAYMENTS	LINE_ITEM_ID
PAYMENT_REMINDERS	USE_ENROLLMENT_EMAIL

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, PmtReminder 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 eaPay jobs sequentially. You can also edit pmtAllCheckTasks to not run specific jobs, if you wish to tailor your environment.

Example User Interface 11

Navigating eaPay Example Interface

eaPay provides a sample interface that demonstrates the standard features available for a customer interface. Each customer will present their own interface, but the sample interface is useful for testing the system.

This section describes the eaPay example interface. At the top of each page is a menu containing a number of payment and account management options:

The menu options are:

Menu Option	Description
Edit Profile	Opens a page showing subscription information completed during enrollment. This page can be used to modify enrollment information.
Bill Summary	Opens a page of all payment activity for the this account.
View Invoice	Opens a list of invoices for paid bills.
Recurring Payment	Opens a page showing the types of recurring payments available, and allows you to set them up for an existing account.
Future Payments	Opens a page listing all future bill payments for the customer account.

Menu Option	Description
Payment History	Opens a page from which you can view a summary of payment history for the customer account.
Schedule Payment	Opens a page to schedule a payment.
Instant Payment	Opens a page where you can make an instant payment using a credit card.
Issue Credit	Schedules a credit on the specified bill.
Payment Reminder	Opens a page listing all accounts, where you can configure a payment reminder for any account that will cause eaPay to send email notification to remind you to pay a bill.
Logout	Logs out of the payment interface and returns to the enrollment login page.

Enrolling for Bill Payment and Account Management

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

Customer enrollment only needs to be done once. After that, customers 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. Connect to the Command Center and 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:

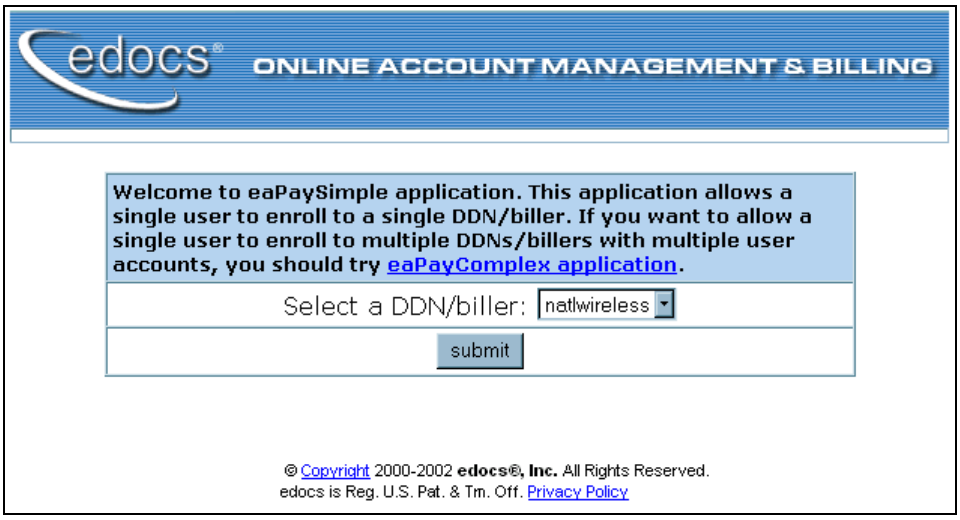
   `http://<webserver>:<port number>/eaPaySimple`

   or

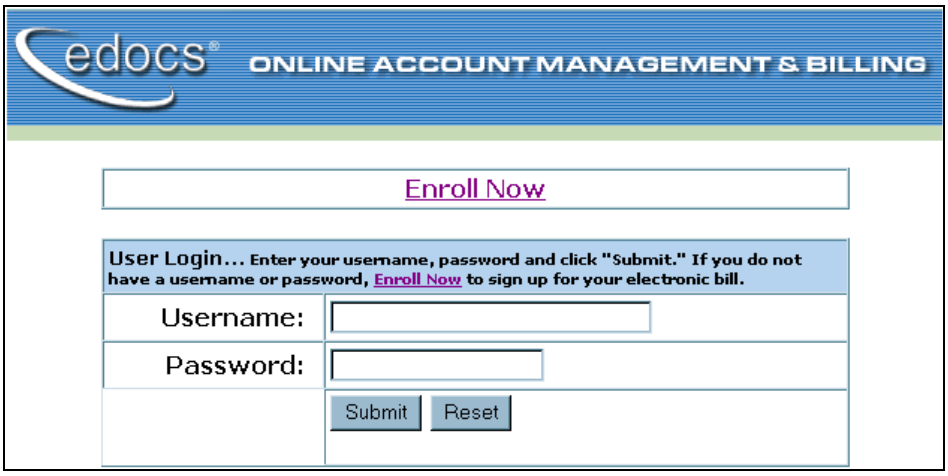
   `http://<webserver>:<port number>/eaPayComplex`

   depending on which application is installed. This document shows the screens from the eaPaySimple application.

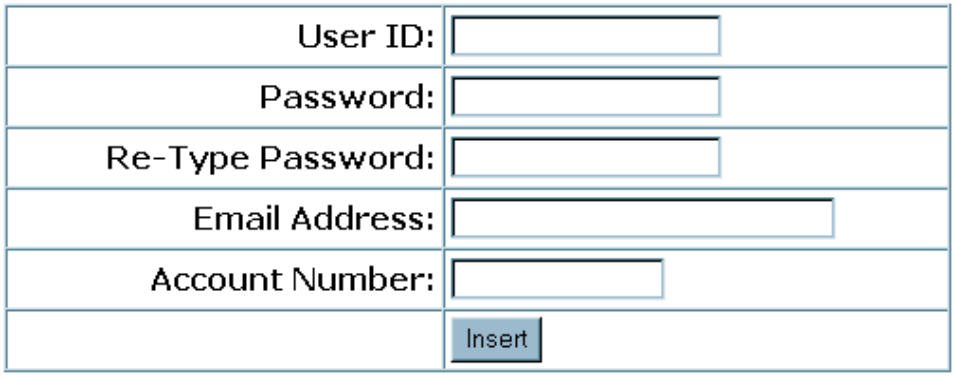
3. Choose the DDN you wish to work with, and click **`submit`**. The page appears where you select the biller to work with:



4. Log on or click **`Enroll Now`** to create a new user profile.

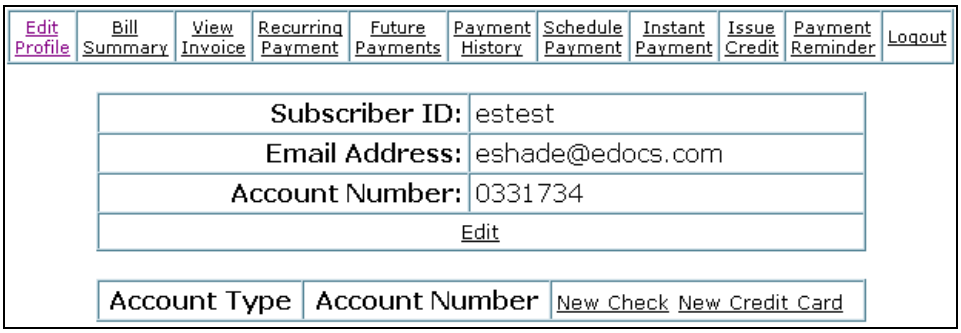


5. Clicking **`Enroll Now`** creates the following screen:

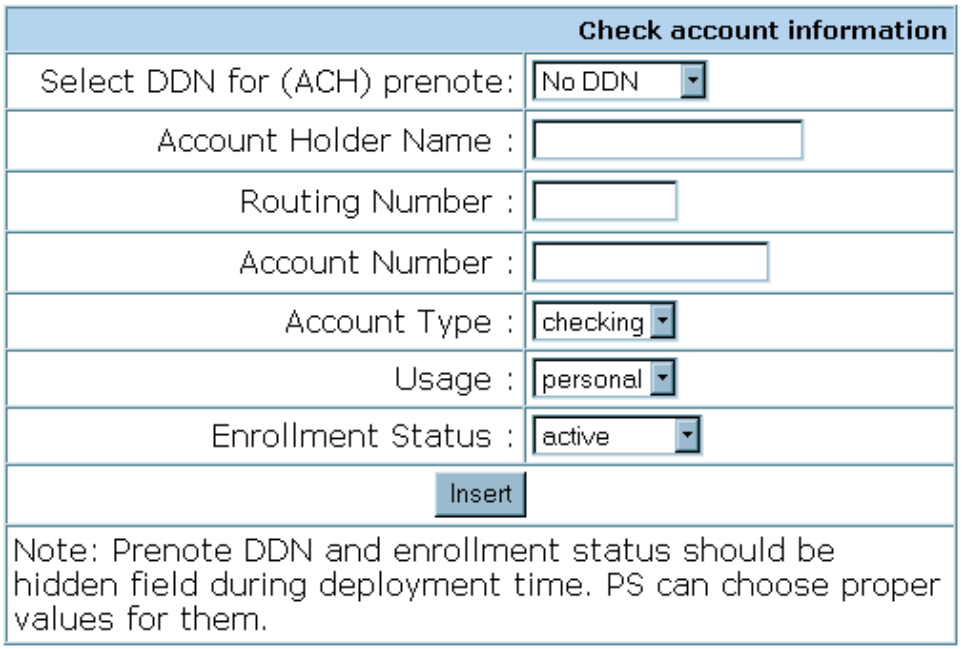


Fill in the form with the required information. Click **`Insert`**, and a new login page appears, where you can login using your new ID.

6. Before you can use the payment features, you must create a profile that includes payment account information. Click on **`Edit Profile`**, and the current profile information is displayed:



7. Create one or more accounts. For example, clicking on **`New Check`** brings up the following screen:

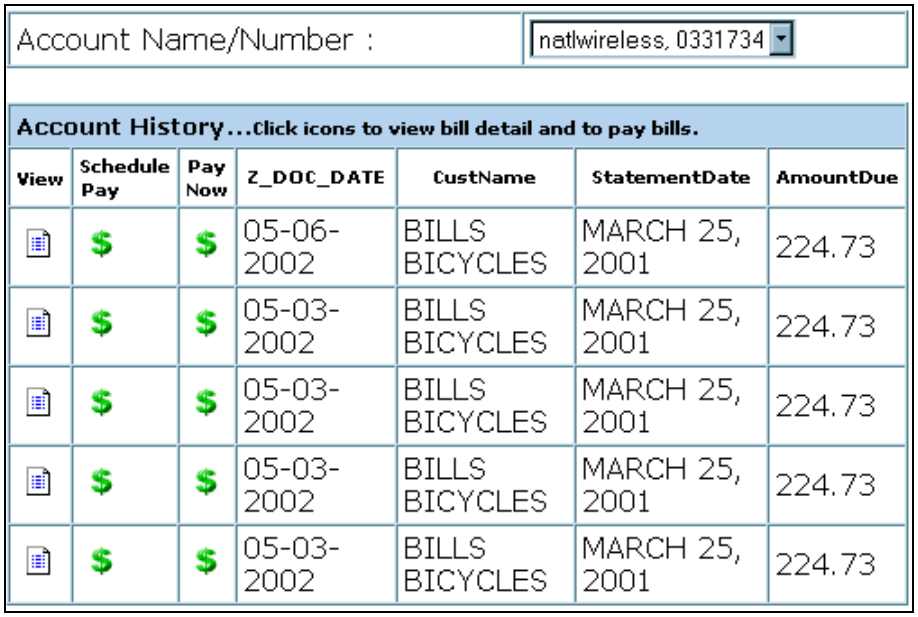


Choose from the available Applications (also known as biller or DDN), enter the name you use with that biller plus the account information you have with that biller. Then click **`Add Check Account`** to add this to the list of billers whose bills you can pay.

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 up due, and make a payment.

1. Click on **`Bill Summary`** to view available bills to pay:



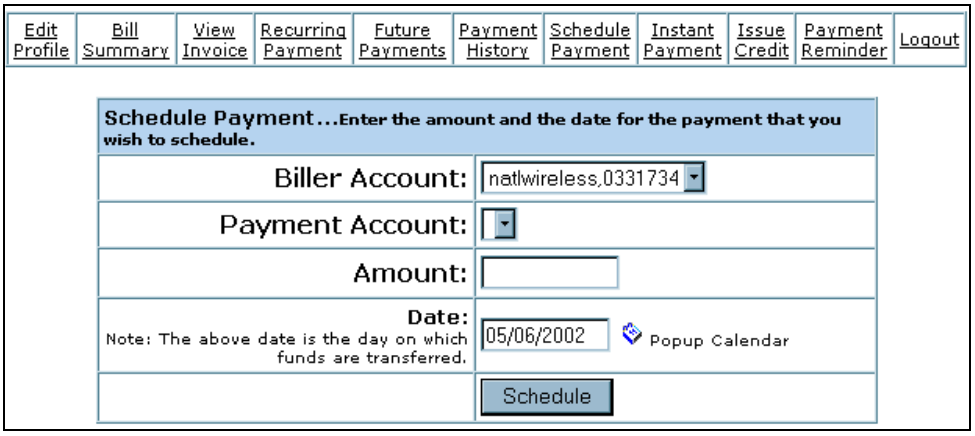
Account Name/Number : natlwireless, 0331734

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

Click the icon to view a detailed version of the bill.

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



3. To simulate a check payment session, enter:

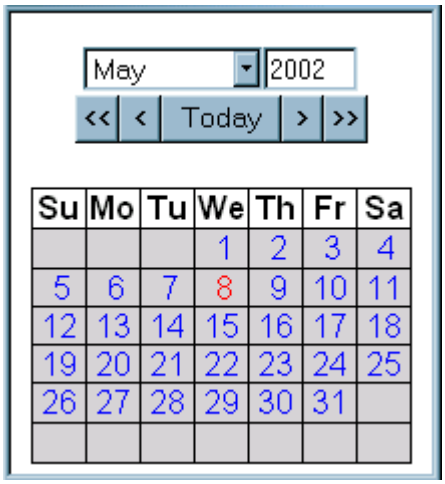
   The Biller Account to make a payment against

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

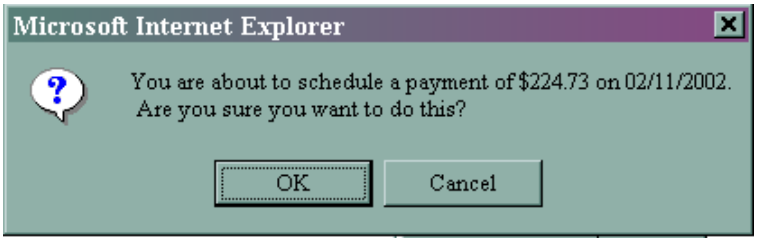
   The amount to pay

   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. eaPay returns a dialog showing the name of the payee, date on which the payment is scheduled, and the payment amount.

Other Payment Pages

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

Link from Top Menu	Description
Payment History	Displays a history of payments.
Payment Reminder	Configures a payment reminder.
Future Payments	View payments that are scheduled to be paid in the future.
Auto/Recur Payment	Schedules a recurring or automatic payment.
Instant Payment	Makes an credit card payment now (not scheduled).

Payment History

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.

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

eaPay pages that list payment summary information, such as Payment History and Future Payments, will list at least one invoice for each payment. It is possible to create a link for each invoice to the invoice details on these pages.

These links are created by customizing JSP files to call *showbill.jsp*. *Showbill.jsp* is shipped as an example of how to retrieve the invoice details.

Payment Reminder

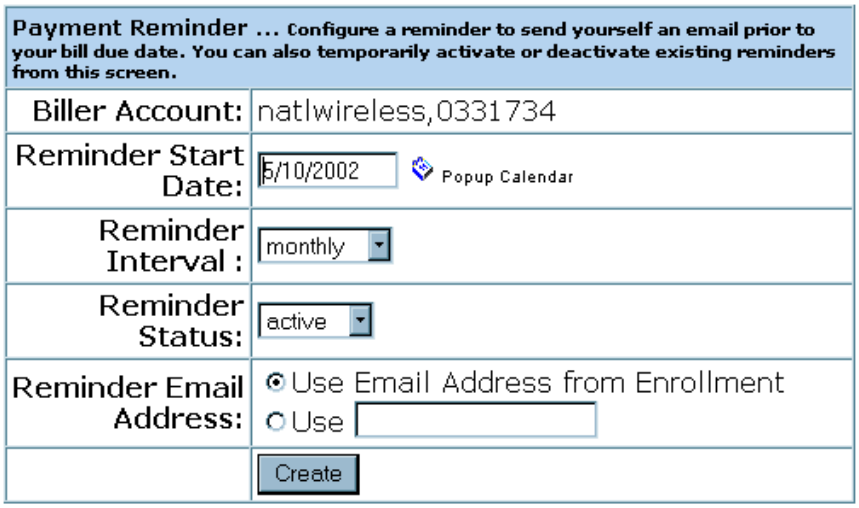
By configuring a payment reminder, you can have eaPay automatically send you an email notification prior to the date your bill is due. Payment reminders can be set to a number of intervals, ranging from weekly to quarterly. From this page you an also deactivate previously configured payment reminders.

To configure a payment reminder:

1. Click **`Payment Reminder`** from 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.



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.

3. Click **`Create`**, and a screen similar to the following appears:

Future Payments

To view a summary of future bill payments:

1. Click **`Future Payments`** on the navigation bar. 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

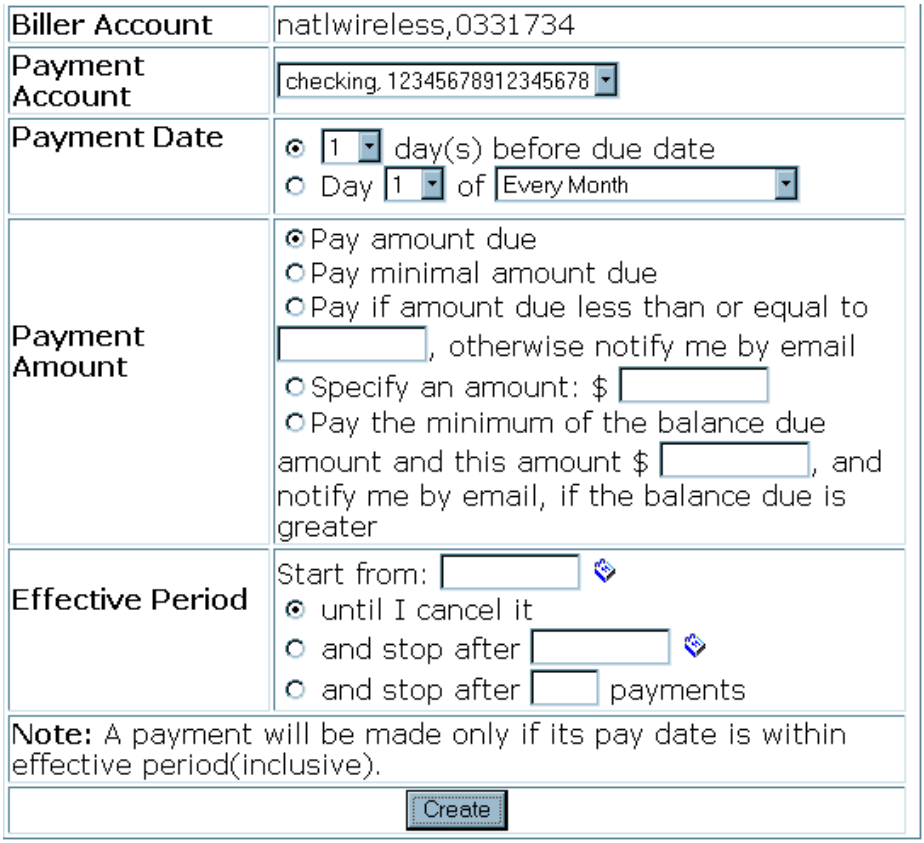
Check payments scheduled in the future:

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

Recurring Payments

The following screen shows an example of the recurring payment example user interface after selecting an account:



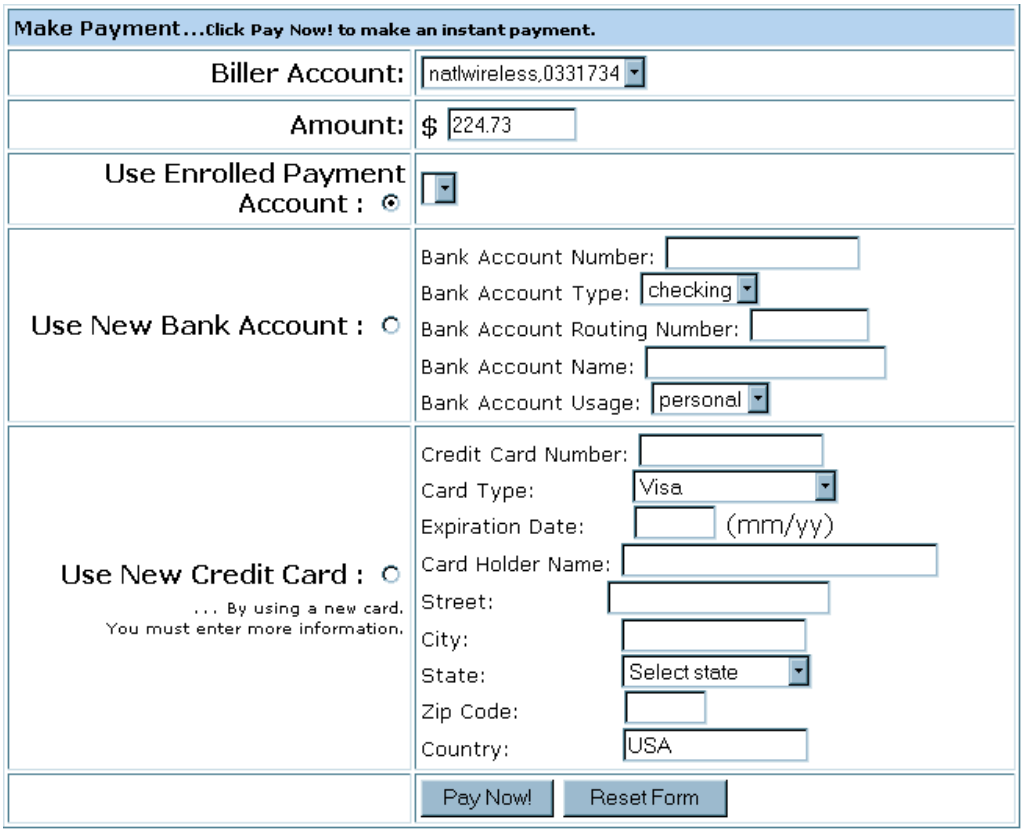
The fields in this dialog are self-explanatory, and display all the recurring payment options. If the pay date is related to the due date, or the pay amount is related to amount due, then eaPay must query eaDirect for information about the bill, which incurs extra processing during the pmtRecurPayment job.

After creating a new recurring payment, you will see a confirmation screen similar to the following:

Following recurring payment has been created!	
Biller Account:	natlwireless,0331734
Pay Date:	1 day before due date
Amount:	Minimal amount due
When Start :	05/09/2002
When Stop :	Until being cancelled

Making an Instant Payment

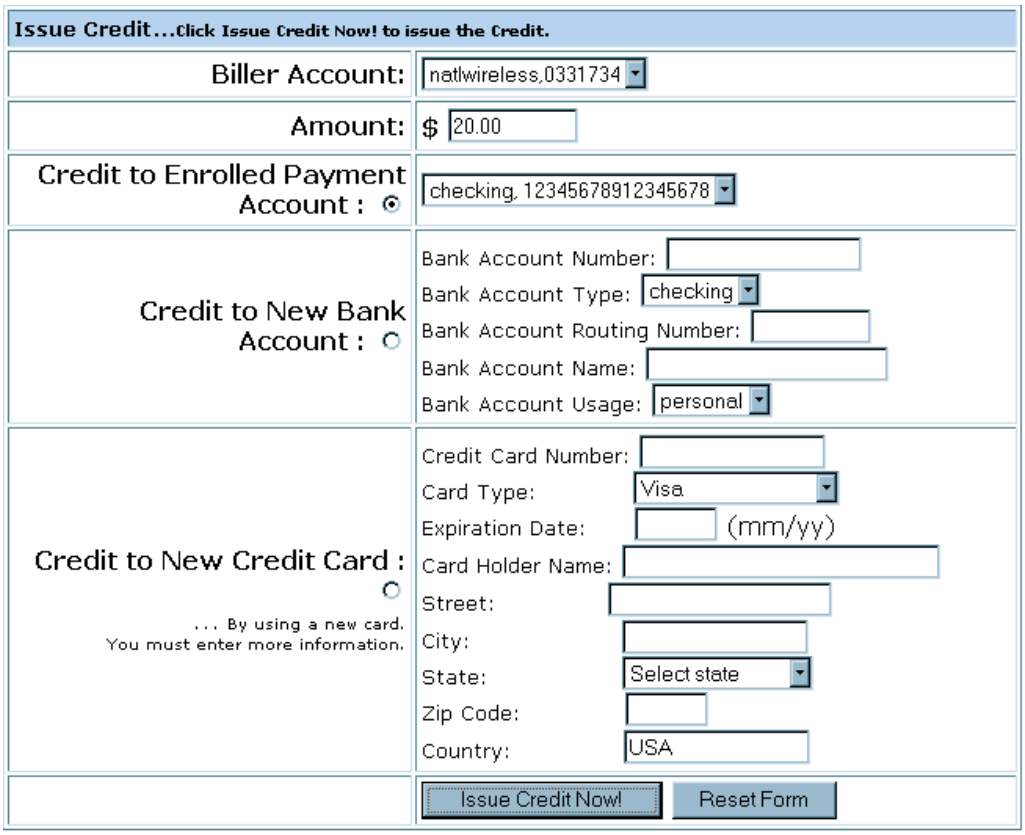
The following screen shows the Instant Payment page provided by the example interface:



You cannot make an instant payment with check accounts. Only credit card accounts support instant payments.

Issuing Credit

You can issue a credit on an account, also known as a payment reversal. The following screen appears if you click on the **`Issue Credit`** link on the navigation bar:



Clicking **`Issue Credit Now`** 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. It also displays a status screen confirming that the credit was scheduled, similar to the following:

Following credit transaction has been scheduled.	
Biller Account:	natlwireless,0331734
Date:	05/08/2002
Amount:	-20.00

Email Template Customization 12

Email templates and when they are used

eaPay uses template files to generate customized text that will be sent in a notification email. Professional Services will customize the email templates for you as part of system installation. This appendix 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	pmtRecurPayment	*recurringNotify.txt*
eaPay command center job status	All eaPay jobs	*notifyPayment.txt*

For Unix, the default path to the email template files is *$EAPAY_HOME/lib/payment_resources/*.
For Windows, it is:
%EAPAY_HOME%\lib\payment_resources.

You can modify the text in these templates, and use the existing variables that reference the eaPay template engine. To add new variables, see the *eaSDK:Customizing and Extending eaPay document*, or contact edocs Professional Services.

Index

A

ACH
- and pmtCheckUpdate, 104
- and the pmtCheckUpdate task, 103
- batch file, 99
- cartridge overview, 15
- change codes, 37
- effective date, 39
- effective date and holidays, 78
- federal holidays, 80
- File Input Directory, 80
- File Output Directory, 80
- immediate destination, 78
- immediate origin, 78
- prenote, 102
- return codes, 37
- return directory, 80
- settlement date, 40
- submit directory, 80
- Template Directory, 80

AllCheckTask, 95

amount due
- format, 88
- Name of Amount Due in eaDirect Index Table, 88

Amount Due Format, 75, 82

B

Batch Size for Credit Card Payment Table, 90

Batch Size for Payment Reminder Table, 81, 87
- gateway parameter, 74

Batch Size for Payment Table, 77, 84

Bill Summary menu, 162

C

Cartridges, 15

CDA
- configuration, 76
- gateway configuration, 77

CDP
- File Input Path, 85
- File Output Path, 85

Check Payment Gateway
- overview, 73
- parameters, 74

check payments

Cancel payment if account information is invalid?, 102
Cancel payment if check account is canceled?, 102
clearing checks, 77
Number of days before a check's pay date for it to be submitted, 102
number of days before pay date to schedule, 118
past payment requirements, 135
payment status notification, 114
pmtCheckUpdate, 103
processing multiple DDNs, 103
scheduling for CheckFree, 35
Submit checks for additional DDNs, 103
Submit payment if check account is pending?, 103
transaction cycle, 29
transaction statuses, 32
user Options, 32
check status
pmtCheckSubmit, 100
CheckFree
and pmtCheckUpdate, 108
and the pmtCheckUpdate task, 103
batch file, 99
cartridge overview, 16
file handling and directories, 34
file names, 33
file types, 35
gateway configuration, 81
overview, 16
PGP, 34
scheduling payments for, 35
statuses, 36
CheckPayment menu, 162
Client ID, 84
Company Entry Description, 79
Company ID, 79
Company Name, 79
credit card
Batch Size for Credit Card Payment Table, 90
Cancel Payment if Credit Card Account has Expired, 109
cartridge overview, 15

Number of Days Before a Credit Card's Pay Date for it to be Submitted, 109
overview, 41
Payment Gateway, 86
Save Full Credit Card Account Number in payment DB, 87
statuses, 42
transaction overview, 43
user options, 45
Verisign overview, 46

D

Database
backup and recovery, 135
maintaining tables, 135
schema, 136
sizing tables, 132
Days to Activate Pending Subscribers, 77, 84
Days to Clear Checks, 77, 84
Due Date Format, 82, 88
gateway parameter, 75

E

eaDirect, 8
eaMarket, 9
eaPay, 8
eaPay jobs
overview, 17
eaPost, 9
eaService, 9
eaSuite, 7
EditProfile menu, 162
email notification
about enrollment, 112
and check scheduling, 118
Email Template for Job Status Notification, 89
enrollment notification, 112
job status configuration, 83
Mail Addresses for Job Status Notification, 89
Mail Server for Job Status Notification, 89
overview, 69
payment reminder menu, 170
payment reminders, 114
recurring payment account cancelled, 119
recurring payment account deleted, 120
recurring payment account invalid, 120
recurring payment expired, 119

recurring payment scheduled, 119
Send Email Notification when Payment Jobs are Done, 89
when a check is cleared, 115
when a check is returned or failed, 116
when a check is sent for processing, 115
when a credit card fails authorization, 116
when a credit card is settled, 116
Email template for job status notification, 76, 83
Encrypt Check Account Number, 74, 84
encryption
chck account number, 74
Encrypt credit card number, 87
enrollment
configuring activation delay, 77
configuring for payment information, 90
configuring for user information, 89
detailed flow, 24
Enrollment XML file(for flat enrollment only), 76, 89
Implementation of IPaymentAccountUserAccessor, 90
Implementation of IUserAccountUserAccessor, 89
payment configuration, 83
sizing tables, 134
user configuration, 83
Error Logs, 125

F

File Transmission Mode, 85
Flexible Field 1 and 2
ACH, 80
CheckFree, 91
FTP User Name, 84
Future Payments menu, 172

G

Gateway
ACH gateway parameter, 74
CheckFree gateway parameter, 81
Generate empty ACH file when there are no checks to submit, 78
Generate Empty File, 85

H

Help
documentation, 10

I

Immediate Destination, 78
Immediate Destination Name, 79
Immediate Origin, 78
Immediate Origin Name, 79
Implementation of ICheckSubmitPlugIn, 80
Implementation of IPaymentAccountUserAccessor, 77, 83
Implementation of IUserAccountUserAccessor, 76, 83
Invoice details menu, 170

J

job status
Email Template for Job Status Notification, 89
Mail Addresses for Job Status Notification, 89
Mail Server for Job Status Notification, 89
Send Email Notification when Payment Jobs are Done, 89
jobs, 103
pmtConfirmEnroll, 108
pmtCreditCardSubmit, 109
pmtNotifyEnroll, 112
pmtRecurPayment, 118
PmtReminder, 114
Jobs
pmtAllCheckTasks, 95
scheduling, 95, 160

L

Log files
errors, 125
exceptions, 123

M

Mail server for job status notification, 76, 83
Mail-to addresses (separated by ";", semicolon) for job status notification, 76, 83
Menus, 162, 169
BillSummary, 162
CheckPayment, 162
EditProfile, 162
payment history, 169
minimum amount due
Minimum Amount Due Format, 88

Name of Minimum Amount Due in eaDirect Index Table, 88
Minimum Amount Due Format, 82
Minimum Amount Due Format, 75

N

Name of Amount Due in eaDirect Index Table, 82
Name of Amount Due in eaDirect Index Table, 75
Name of Due Date in eaDirect Index Table, 82, 87
Name of Due Date in eaDirect Index Table, 75
Name of Minimum Amount Due in eaDirect Index Table, 82
Name of Minimum Amount Due in eaDirect Index Table, 75
noc
enrollment update, 77
NOC
and enrollment, 112

O

ODFI
ACH payment gateway setting, 79

P

Payee Address, 85
Payee City, 85
Payee Name, 85
Payee Number, 85
Payee Short Name, 85
Payee State, 85
Payee Zip, 85
Payment gateway, 72
configuring, 71
deleting, 93
updating, 91
Payment Gateway
check overview, 73
Payment History Menu, 169
Payment Reminder Menu, 170
Payment Reports
Daily Exceptions, 123
Daily Summary, 123
PGP and CheckFree, 34
plugin
ACH gateway, 80
pmtCheckSubmit
ACH company name, 79
ACH immediate origin, 78
and ACH effective date, 39

and recurring payments, 120
company entry description, 79
date, 40
empty ACH configuration, 78
empty CheckFree configuration, 85
immediate destination, 78
ODFI configuration, 79
scheduling for CheckFree, 35
pmtCheckUpdate
ACH company name, 79
ACH immediate destination, 78
ACH immediate origin, 78
and ACH change codes, 37
and ACH return codes, 37
enrollment and NOC, 77
pmtConfirmEnroll
and gateway configuration, 108
pmtCreditCardSubmit
and the payment transaction cycle, 44
pmtNotifyEnroll
ACH NOC, 78
and recurring payments, 120
pmtRecurPayment
and Indexer, 120
eaDirect dependency, 121

R

recurring payments
configuring, 63
Recurring Payments
overview, 47

S

security
Encrypt credit card number, 87
Save Full Credit Card Account Number in payment DB, 87
Send Email Notification in Case of NOC, 78
Send Email Notification when Payment Jobs are Done (with or without error), 83
Send Email Notification when Payment Jobs are Done(with or without error), 76
Sender ID
CheckFree parameter, 84
Skip non-business days for batch effective entry date, 78

T

template
CheckFree, 85
for pmtNotifyEnroll, 112
payment reminders, 115
recurring payment email notification, 119

U

Update eaPay enrollment in Case of NOC, 77

V

Verisign
Certificate Path, 91
Host Name, 90
Host Port, 90
Implementation of IVeriSignCreditCardSubmitPlugIn, 91
Number of Threads, 91
Partner, 91
Password, 91
Timeout Period for Transactions, 90
User, 90
Vendor, 91
Viewing, 169, 170, 172
error logs, 125
exception reports, 123
future payments menu, 172
invoice details menu, 170
Payment History menu, 169