Payment Designer Guide

V4.5
Document ID: EDXG-06-4.5-01
Date Published: 8.30.2004

© 1997–2004 edocs® Inc. All rights reserved.

edocs, Inc., One Apple Hill Dr., 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 is 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.

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

Printed in the USA.

Table of Contents

Preface **5**

1 Check Payments **9**

Check Payment Overview 9

Adding a Check Account 9

Check Account Enrollment Status Flow 10

Check Payment Transactions 11

Check Payment Status Flow 12

Credit 13

ACH 13

Supported SEC Codes 13

ACH Change Codes (NOC) 13

ACH Return Codes 14

NOC Transactions 15

ACH Effective Date 15

ACH Settlement Date 15

ACH Addenda Records 15

2 Credit Card Payments **17**

Credit Card Payment Status 17

Credit Card Payment Transactions 18

Instant Credit Card Payments 18

Scheduled Credit Card Payments 19

Reversals 20

User Options 20

Using VeriSign as a Payment Gateway 20

AVS (Address Verification Service) 20

3 Recurring Payments **23**

Overview 23

Recurring Payment Transaction Cycle 24

Tables Affected by Recurring Payments 26

Recurring Payment Examples 26

Scheduling Payment Jobs 34

Payment Job Status Monitoring 34

Payment Job Plug-In 34

To Configure Recurring Payments 34

Testing Recurring Payment 35

Case 1: Pay Amount Due *X* days Before Due Date 35

Case 2: Pay Amount Due on a Fixed Date 36
Case 3: Pay fixed Amount *X* Days Before Due Date 37
Case 4: Pay Fixed Amount On A Fixed Date 37
Rebill and Recurring Payment 38
Description 38
Payment Settings 38
Payment History 39
Email 39
Logic 39
4 Configuring Payment Gateways 41
Configuring a Payment Gateway 41
Payment Global Configuration 42
Check Payment Gateways 44
ACH Gateway Parameters 44
ACH Federal Holidays 49
CheckFree Gateway Parameters 49
Credit Card Payment Gateways 53
Updating a Payment Gateway Configuration 56
Deleting a Payment Gateway Configuration 57
Table Column Definitions 57
5 Payment Tables 59
Payment indexes 69
Payment Database Migration 70
Index 71

Preface

About Customer Self-Service and Payment™

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

About This Guide

This guide describes the tasks required to setup and configure Payment.

This guide is intended for the designer of the Payment implementation, who will design a payment production environment.

This guide assumes you have installed your edocs application(s).

Related Documentation

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

Print Document	Description
Installation Guide	The installation guide for your application that explains how to install and configure it in a distributed environment.
Payment Administration Guide	How to set up and run Payment with your J2EE application .
Payment Design Guide	How to design and implement Payment in a production environment.

Obtaining edocs Software and Documentation

You can download edocs software and documentation directly from Customer Central at https://support.edocs.com. After you log in, click on the Downloads button on the left. When the next page appears, you will see a table displaying all of the available downloads. To search for specific items, select the Version and/or Category and click the Search Downloads button. If you download software, an email from edocs Technical Support will automatically be sent to you (the registered owner) with your license key information.

If you received an edocs product installation CD, load it on your system and navigate from its root directory to the folder where the software installer resides for your operating system. You can run the installer from that location, or you can copy it to your file system and run it from there. The product documentation included with your CD is in the Documentation folder located in the root directory. The license key information for the products on the CD is included with the package materials shipped with the CD.

If You Need Help

Technical Support is available to customers who have an active maintenance and support contract with edocs. Technical Support engineers can help you install, configure, and maintain your edocs application.

This guide contains general troubleshooting guidelines intended to empower you to resolve problems on your own. If you are still unable to identify and correct an issue, contact Technical Support for assistance.

Information to Provide

Before contacting edocs Technical Support, try resolving the problem yourself using the information provided in this guide. If you cannot resolve the issue on your own, be sure to gather the following information and have it handy when you contact technical support. This will enable your edocs support engineer to more quickly assess your problem and get you back up and running more quickly.

Please be prepared to provide Technical Support the following information:

Contact information:

- Your name and role in your organization.
- Your company’s name
- Your phone number and best times to call you
- Your e-mail address

Product and platform:

- In which edocs product did the problem occur?
- What version of the product do you have?

- What is your operating system version? RDBMS? Other platform information?

Specific details about your problem:

- Did your system crash or hang?
- What system activity was taking place when the problem occurred?
- Did the system generate a screen error message? If so, please send us that message. (Type the error text or press the Print Screen button and paste the screen into your email.)
- Did the system write information to a log? If so, please send us that file. For more information, see the troublshooting guide for your application.
- How did the system respond to the error?
- What steps have you taken to attempt to resolve the problem?
- What other information would we need to have (supporting data files, steps we'd need to take) to replicate the problem or error?

Problem severity:

- Clearly communicate the impact of the case (Severity I, II, III, IV) as well as the Priority (Urgent, High, Medium, Low, No Rush).
- Specify whether the problem occurred in a production or test environment.

Contacting edocs Technical Support

You can contact Technical Support online, by email, or by telephone.

edocs provides global Technical Support services from the following Support Centers:

US Support Center
Natick, MA
Mon-Fri 8:30am – 8:00pm US EST
Telephone: 508-652-8400

Europe Support Center
London, United Kingdom
Mon-Fri 9:00am – 5:00 GMT
Telephone: +44 20 8956 2673

Asia Pac Rim Support Center
Melbourne, Australia
Mon-Fri 9:00am – 5:00pm AU
Telephone: +61 3 9909 7301

Customer Central
https://support.edocs.com

Email Support
mailto:support@edocs.com

Escalation Process

edocs managerial escalation ensures that critical problems are properly managed through resolution including aligning proper resources and providing notification and frequent status reports to the client.

edocs escalation process has two tiers:

1. **Technical Escalation -** edocs technical escalation chain ensures access to the right technical resources to determine the best course of action.

2. **Managerial Escalation** - All severity 1 cases are immediately brought to the attention of the Technical Support Manager, who can align the necessary resources for resolution. Our escalation process ensures that critical problems are properly managed to resolution, and that clients as well as edocs executive management receive notification and frequent status reports.

By separating their tasks, the technical resources remain 100% focused on resolving the problem while the Support Manager handles communication and status.

To escalate your case, ask the Technical Support Engineer to:

1. Raise the severity level classification.

2. Put you in contact with the Technical Support Escalation Manager.

3. Request that the Director of Technical Support arrange a conference call with the Vice President of Services.

4. Contact VP of Services directly if you are still in need of more immediate assistance.

1 Check Payments

Check Payment Overview

Payment 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 Payment 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. Payment saves the information in the *payment_accounts* table with an enrollment status of "pnd_active".

2. The pmtSubmitEnroll job runs to submit the enrollment information to the payment gateway. It changes the enrollment status to "pnd_wait". 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.

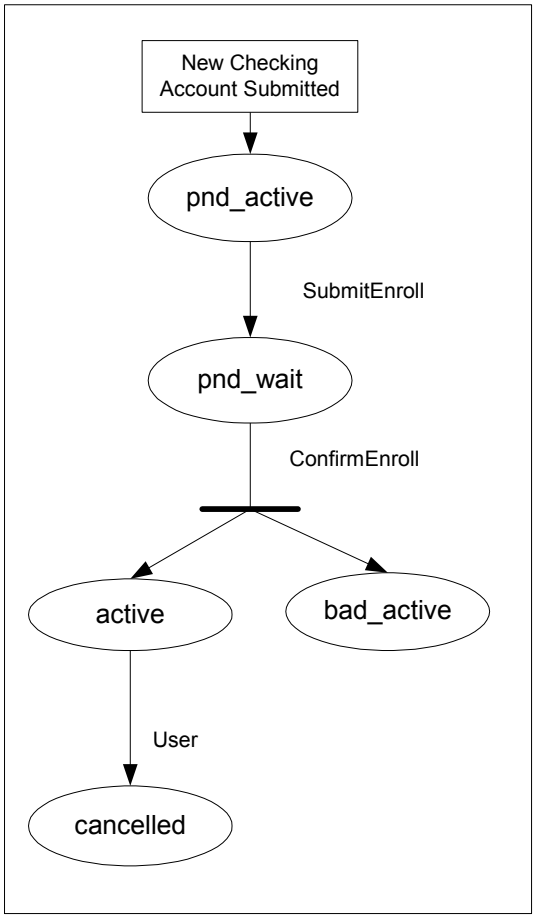
4. The pmtConfirmEnrollment job runs. This job updates the status of the customer enrollment status to "active "if there are no problems after a specified number of days (by default, three days).

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.

Check Account Enrollment Status Flow

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



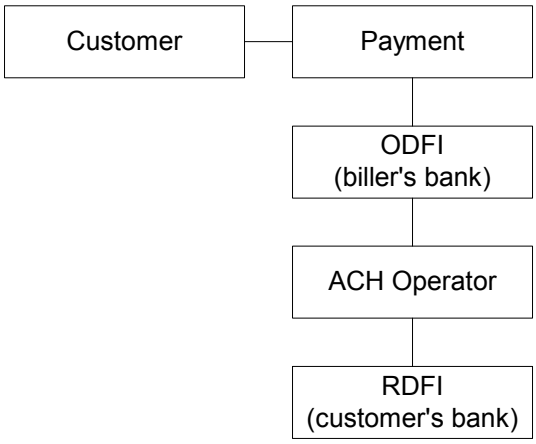
The following table describes each status:

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

Enrollment Status	Description
bad_active	The check account failed to be activated.

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. Payment 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 Payment 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 Payment.

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".

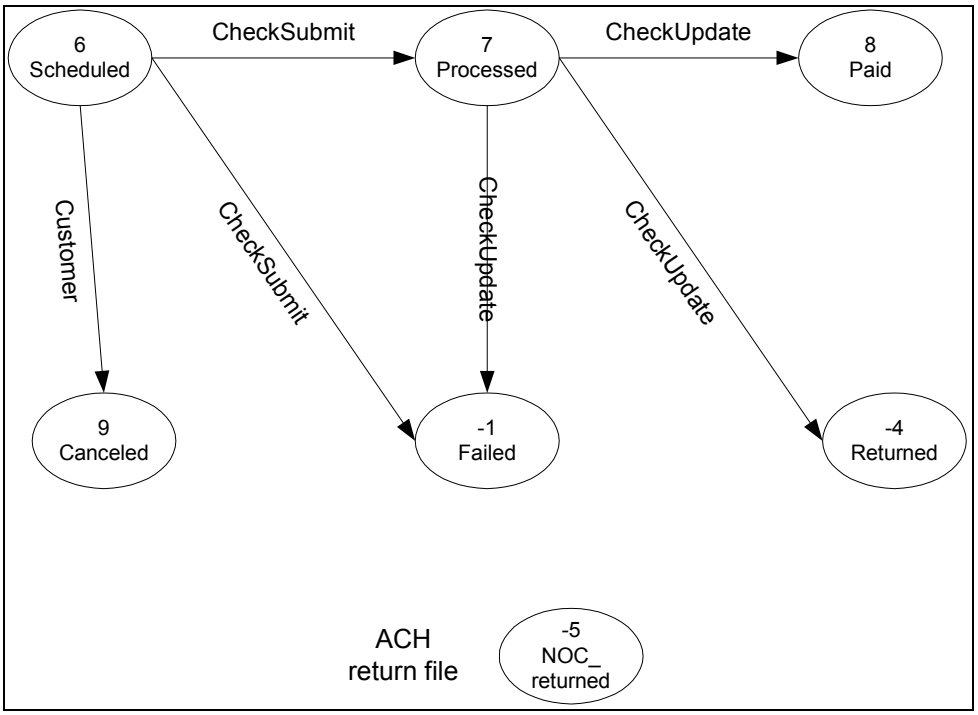
If the payment gateway returns the check, the pmtCheckUpdate job updates the check's status to "returned", and saves the reason code in the *txn_err_msg* field of the *check_payments* table. An exception report is generated to summarize the information in the returned file, which can be viewed from Command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

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

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 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)	Payment 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".

Transaction Status	Description
Returned(-4)	ACH returned a check.
noc_returned(-5)	This customer's payment account information needs to be changed.

Credit

Credit reversals are supported.

ACH

Supported SEC Codes

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

- **Web**- Internet initiated entry (default for Payment).

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

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

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

ACH Change Codes (NOC)

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

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

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

ACH Return Codes

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

Code	ACH Return Code Description
R01	Insufficient Funds
R02	Account Closed
R03	No Account/Unable to Locate Account
R04	Invalid Account Number
R05	Reserved
R06	Returned per ODFI's Request
R07	Authorization Revoked by Customer (adjustment entries)
R08	Payment Stopped or Stop Payment on Item
R09	Uncollected Funds
R10	Customer Advises Not Authorized; Item Is Ineligible, Notice Not Provided, Signatures Not Genuine, or Item Altered (adjustment entries)
R11	Check Truncation Entry Return (Specify) or State Law Affecting Acceptance of PPD Debit Entry Constituting Notice of Presentment or PPD Accounts Receivable Truncated Check Debit Entry
R12	Branch Sold to Another DFI
R14	Representative Payee Deceased or Unable to Continue in that Capacity
R15	Beneficiary or Account Holder (Other Than a Representative Payee) Deceased
R16	Account Frozen
R17	File Record Edit Criteria (Specify)
R20	Non-Transaction Account
R21	Invalid Company Identification
R22	Invalid Individual ID Number
R23	Credit Entry Refused by Receiver
R24	Duplicate Entry
R29	Corporate Customer Advises Not Authorized
R31	Permissible Return Entry (CCD and CTX only)
R33	Return of XCK Entry

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

NOC Transactions

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

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

ACH Effective Date

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

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

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

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

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

Payment 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.

2 Credit Card Payments

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. Payment 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 Payment, 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.

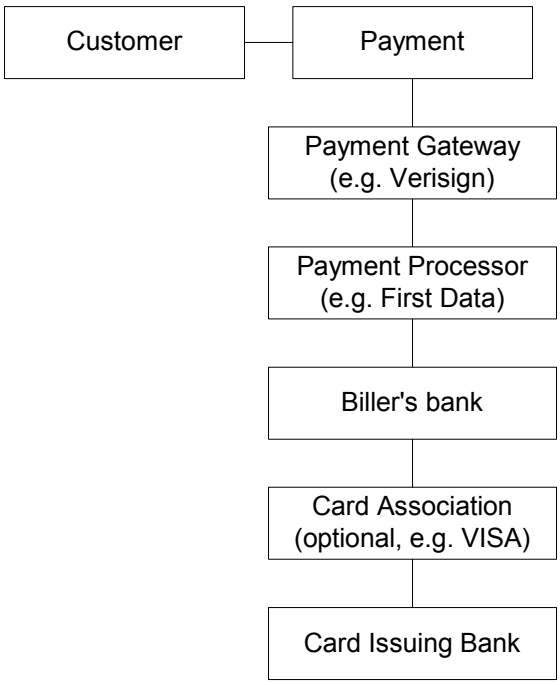
Credit Card Payment Status

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 (-1)	A credit card payment failed because of network problems. This state occurs only for instant payments. For scheduled payments or recurring payments, the state stays "scheduled" if there is a network problem, so that it will be tried again. There is no need for Payment 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:



Credit card processing usually goes through the following steps:

1. A user enters credit card number and other card related information.

1. The card information is sent to the card-issuing bank for authorization. Authorization only guarantees that the money is available at the time of authorization.

2. The merchant issues a settlement request to issuing bank so that the money can be transferred. The merchant usually does this after fulfillment (sending out ordered goods). For bill payments, the biller does not send out ordered goods, so authorization and synchronization are combined into one operation; a credit card payment is settled at the same time it is authorized.

3. Since credit card is processing is real-time, real-time and not batch-based, the life cycle of credit card is simpler than check processing.

Instant Credit Card Payments

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

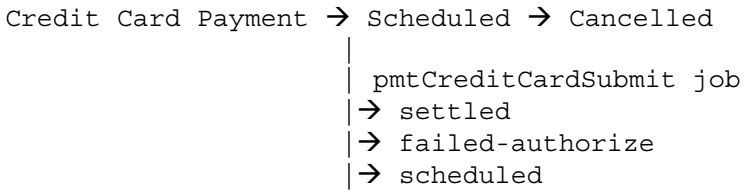
```
Credit Card Payment → settled
                    |-> failed-authorize
                    |-> Failed
```

1. A user submits an instant credit card payment from the UI.

2. Payment 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".
4. If the card failed to authorize, the state is set to "failed_authorize".
5. If there is a network problem, the state is set to "failed".
6. The card is inserted into *creditcard_payments* table.
7. The result of the transaction is presented to the user.
8. The pmtPaymentReminder job runs and (optionally) sends emails to users who have made an instant payment.

Scheduled Credit Card Payments

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



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 Payment Settings, this job modifies the credit card payments that are scheduled to be paid, or takes other actions related to the selected credit card payments. Functions in the plugin are called before and after credit card payment processing. For more information about the pmtCreditCardSubmit job and its *IVerisignCreditCardSubmitPlugin*, see the description of the pmtCreditCardSubmit job in the *Payment Administration Guide*. 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.

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

Reversals

Credit reversals are supported.

User Options

The user interface to Payment can offer a variety of credit card payment options. Some of those options require that fields be configured in Payment 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 Payment. Before configuring a VeriSign credit card payment gateway, you must obtain a digital certificate through VeriSign. When choosing VeriSign as the credit card gateway type, the Payment Settings page 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 Payment.

You must also configure your application server to support a VeriSign payment gateway. See the *Payment Installation and Configuration* guide for details.

AVS (Address Verification Service)

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

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

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

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

3

Recurring Payments

Overview

Payment 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 (Payment) 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 Composer. The name and format of the (minimum) amount due must be specified in the Payment 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 Composer. The name and format of the due date must be specified in the Payment 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 paid twice. For more information about that script, see the 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.

Recurring payment supports only one customer account per biller. Recurring payment does not support multiple customer accounts with a single biller.

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.

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

1. pmtRecurringPayment gets 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 the command center database 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 the comand center database, which means there is no bill information associated with this recurring payment.

2. pmtRecurringPayment schedules payments (inserts a payment with status of "scheduled "in the *check_payments* or *creditcard_payments* table so that the payments 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 pmtRecurringPayment job:

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

There is no payment inserted into *check_payments* or *creditcard_payments* table when a recurring payment is created by the user. Payments are inserted by the pmtRecurringPayment job. See the section about the pmtRecurringPayment 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 command center tables and saved into the *payment_bill_summaries* table. After the pmtRecurringPayment job runs, the *payment_bill_summaries* table is populated, and the *bill_id* of the *recurring_payments* table is also populated.

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

recurring_payments Column Name	Value
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 command center Indexer job runs and indexes one bill (the doc id is bill1, in this example) on 03/10/2001. On 04/10/2001, the indexer job runs again and indexes two more bills: bill2 and bill3.

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

3. The pmtRecurringPayment job runs on 04/10/2001 23:59:00PM, after the Indexer job. The job searches the *recurring_payments* table to find all recurring payments whose *bill_scheduled* is "Y" and status is "active". It finds the example recurring payment and then asks command center to return all bills whose account number is acct1111 and whose *Z_DOC_DATE* is between 04/10/2001 (*last_process_time*) and 04/10/2001 23:59:00PM (job run time). Two bills, bill2 and bill3 will be returned. pmtRecurringPayment then finds the bill with latest due date bill3. bill2 is ignored because only the latest bill is paid.

4. After finding the latest bill from command center, pmtRecurringPayment checks whether the due date of this bill is after the due date of the bill used in the last payment (last bill info can be retrieved from *payment_bill_summaries* using the *bill_id*). If not, that means this is an old bill and should not 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 pmtRecurringPayment 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*, pmtRecurringPayment runs again. The recurring payment mentioned previously won't be synchronized, because it's *bill_scheduled* is "N". However, it will be scheduled. pmtRecurringPayment finds all recurring payments whose *bill_scheduled* is N, *status* is "active "and *next_pay_date* is equal to or before 05/14/2001 (05/11/2001 + 3 days). The previously mentioned recurring payment is picked up and a payment is inserted into the *check_payments* or *creditcard_payments* table. The amount of the payment is $100.00, and the pay date is 05/14/2001. After this, the recurring payment table is changed to:

Column Name	Value
payer_account_number	acct1111
bill_scheduled	Y, means this bill has been paid
status	"active" because *next_pay_date* is within the effective period
last_process_time	04/10/2001 23:59:00PM, unchanged since there was no synchronization
last_pay_date	05/14/2001, change to check's pay date
next_pay_date	05/14/2001, unchanged
bill_id	bill3
payment_id	points to the new *payment_id* inserted into the *check_payments* or *creditcard_payments* table

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

8. On 05/12/2001 23:59:00PM, pmtRecurringPayment runs again and finds bills whose doc date is between 04/10/2001 11:59:00PM and 05/12/2001 23:59:00PM. No bills exist, and the last process time 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 pmtRecurringPayment job runs again. It contacts command center 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
last_pay_date	01/01/1970
next_pay_date	4/30/2001; the first available pay date after 04/10/2001 (because there is no April 31).
bill_id	null
end_date	01/01/3000; The end date is so far in the future that the recurring payment will only be deactivated when the number of payments reaches maximum allowed.
curr_num_payments	0; no payments yet.

The index table has the following values:

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

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

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

Column Name	Value
payer_account_number	acct1111
bill_scheduled	N; this bill has not been paid.
status	"active"; *curr_num_payments* is less than max_num_payments.
last_process_time	04/10/2001 23:59:00PM; changes to job run time.
last_pay_date	01/01/1970; unchanged.
next_pay_date	04/30/2001; there is no April 31.
bill_id	bill3
curr_num_payments	0

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

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

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

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

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

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

Repeat steps 2, 3 and 4 until *next_pay_date* is after *end_date*, when status 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
last_pay_date	01/01/1970
next_pay_date	05/01/2001
bill_id	null
end_date	06/10/2001
curr_num_payments	0; no payment yet.

2. On 04/28/2001, three days before *next_pay_date*, pmtRecurringPayment runs again. There is no synchronization (*bill_scheduled* is always "N") but a payment is inserted into the *check_payments* or *creditcard_payments* table. The amount of the check is $50.00 and its pay date is 05/01/2001. The columns in the *recurring_payments* table are updated as follows:

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

Repeat step 2 until *next_pay_date* is after *end_date*. Then the status 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 the command center 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 pmtRecurringPayment, pmtCheckSubmit, pmtCheckUpdate and pmtPaymentReminder in that order. For enroll jobs, we recommend that you run pmtSubmitEnroll, pmtConfirmEnroll and pmtNotifyEnroll in that order. There is only one credit card job, pmtCreditCardSubmit, and we suggest that you run it and then the pmtPaymentReminder job.

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

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 Payment Settings.

Payment Job Plug-In

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

To Configure Recurring Payments

Recurring payments are configured by the pmtRecurringPayment job.

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

See the pmtRecurringPayment 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 on the pmtRecurringPayment job configuration page, the number of days before pay date to schedule the payment defaults to three. This does not affect when a payment is **made**; it affects **how long** before the pay date the customer 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 pmtRecurringPayment task. The payment should be shown as "scheduled", because it is now one day before the due date. In the *recurring_payments* table, *bill_scheduled* should equal "Y," and both *next_pay_date* and *last_pay_date* should be 04/09/2002. These values 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 that the pmtRecurringPayment job had not been run on April 2, but only after the system date was set to April 9, then the bill would be synchronized and the payment would be scheduled at the same time.

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

Case 2: Pay Amount Due on a Fixed Date

1. Some billing data is already indexed, and a user is enrolled with a valid payment account.

2. On April 1, a user sets up a recurring payment, to pay the amount due on the 10th day of the month. Only the amount due must be an indexed field. The start date is April 2, 2002 (the earliest possible start date).

3. Move system date ahead to April 2, 2002.

4. Index another bill.

5. Run the recurring payment task.

6. The bill in question should now be recorded in the *payment_bill_summaries* table; it has been synchronized, but not yet scheduled. The *recurring_payments* record should show this bill's *bill_id*, *bill_scheduled* should equal "N," and the *next_pay_date* should be 04/10/2002 (the customer's desired pay date).

7. Move the system date ahead to April 7, 2002.

8. Run the pmtRecurringPayment job. The payment should be shown as "scheduled", if the "number of days before pay date to schedule the payment" on the pmtRecurringPayment configuration page is set to 3. In the *recurring_payments* table, *bill_scheduled* should equal "Y," *next_pay_date* should be 05/10/2002, and *last_pay_date* should be 04/10/2002.

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

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

Case 3: Pay fixed Amount *X* Days Before Due Date

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

Case 4: Pay Fixed Amount On A Fixed Date

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

4. Run the pmtRecurringPayment task. The payment should be shown as "scheduled", if the number of days before pay date to schedule the payment on the pmtRecurringPayment configuration page is set to "3". In the *recurring_payments* table, *bill_scheduled* should equal "Y," *next_pay_date* should be 05/10/2002, and *last_pay_date* should be 04/10/2002. 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.

Rebill and Recurring Payment

Description

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

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

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

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

Payment Settings

The following parameter belongs to the **pmtRecurringPayment** job: "When to synchronize recurring payment with the command center".

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

The following additional parameter has been added to **Payment Settings**:

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

- *com.edocs.payment.imported.eadirect.BillDepot* - default

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

Payment History

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

Email

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

Logic

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

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

1. If the setting of When to Synchronize recurring payment with the command center is **`Whenever job runs`**, the pmtRecurringPayment job synchronizes the bill with the command center every time pmtRecurringPayment runs. If it is **`Only after the current bill is scheduled`**, a rebill will not be synchronized unless it is time to schedule the payment.

2. The pmtRecurringPayment job runs again. If the setting of Synchronize with the command center Every Time the Job Runs is Yes, and a rebill arrives, the pmtRecurringPayment job synchronizes the bill with the command center.

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

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

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

4

Configuring Payment Gateways

Configuring a Payment Gateway

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

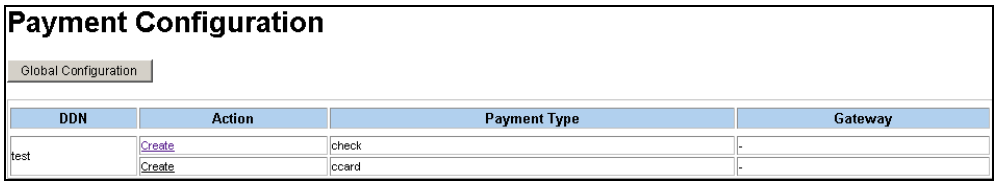
Configuration information that is specific to a payment gateway must be provided by the payment gateway or by the biller's bank. This information should be in hand before configuring.

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*>/*edocs*). 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.

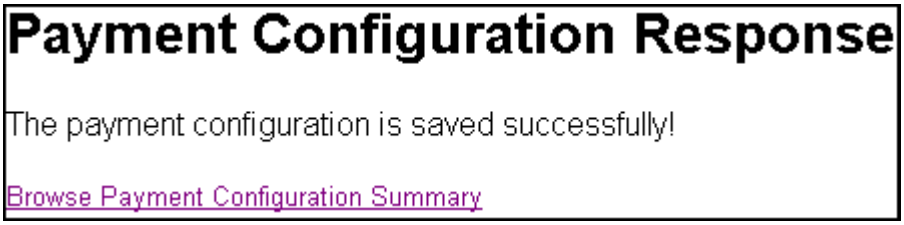
3. Click the **`Payment Settings`** tab. A Browse Payment Configuration page appears showing the name of the DDN (Application) and Payment Types that can be configured, along with a Global Configuration button.



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, click on the Global Configuration button and configure those parameters.

8. After completing configuration, 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 Payment, 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.

The following sections describe the parameters for Global Configuration, and the Check and Credit Card payment gateways.

Payment Global Configuration

Clicking the Global Configuration button allows you to set these parameters:

Credit Card Account

Save Full Credit card Number: **Y** saves the full Credit card Number in Payment database. **N** saves only last four digits of the credit card number in the Payment database.

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

Check Account

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

Audit Configuration

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

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

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

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

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

Email Notification Audit

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

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

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

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

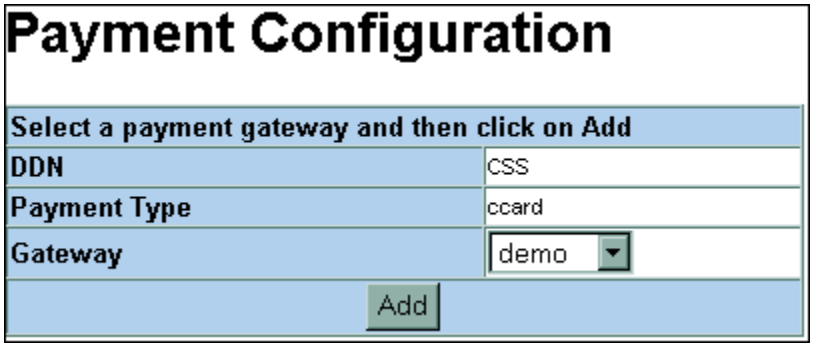
Enable Payment Status Notification Audit: Y causes payment status notification emails to be audited for both check and credit card payments. **N** disables payment status notification email auditing. The default is **N**.

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

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

Check Payment Gateways

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 command center 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. Payment **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 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.

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

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

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

Name of Due Date in command center 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 command center database, during data processing, for use in Payment. Please see the *Administration 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 command center 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 command center database, during data processing, for use in Payment. Please see the *Administration 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 command center Index Table - The name of the variable defined by the DefTool for minimum amount due, which must be indexed by the command center. 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 Payment 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 command center user information, which is determined by the type of enrollment supported. The options are:

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

or

com.edocs.payment.payenroll.usracct.JNDIMultipleDDNUserAccountAccessor, for when multiple DDN command center 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 Payment user information. The options are:

com.edocs.payment.payenroll.payacct.SSOPaymentAccountAccessor, for when Payment 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.

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.

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

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 Payment enrollment in Case of NOC - "Y" causes the pmtCheckUpdate job to update enrollment status when a NOC is returned. "N" means that this value 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. Payment does not create this directory.

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

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

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

For information about implementing this class, contact 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 Payment 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.

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

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

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

JNDI Name of DataSource - Determines the type of datasource used for this DDN. *edx/ejb/edocsDataSource* tells Payment to look in the command center/Payment database. *edx/ejb/XSDataSource* tells Payment to look in the XS store. The datasource used is determined by which job the command center used to read the data; either the Indexer or XML Loader.

Name of Due Date in command center 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 command center database, during data processing, for use in Payment. Please see the *Administration 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 command center 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 command center database, during data processing, for use in Payment. Please see the *Administration 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 command center Index Table - The name of the variable defined by the DefTool for minimum amount due, which must be indexed by the command center. 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 Payment 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 command center user information, which is determined by the type of enrollment supported. The options are:

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

or

com.edocs.payment.payenroll.usracct.JNDIMultipleDDNUserAccountAccessor, for when multiple DDN command center 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 Payment user information. The options are:

com.edocs.payment.payenroll.payacct.SSOPaymentAccountAccessor, for when Payment 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.

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.

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

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.

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 the command center 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.

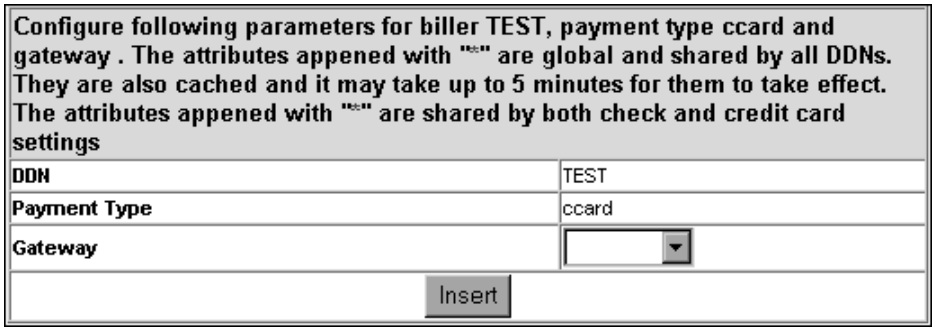
Template File Directory - For a standard installation, should be:

/opt/PAYMENT_HOME/lib\payment_resources/ checkfree/cdp/template (Unix)

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

Credit Card Payment Gateways

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 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.

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

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

- If the rebill has a zero amount, the due date is set to the previous bill's due date.

- If the due date is "ON RECPT", the due date is changed to the current date.

Name of Due Date in command center 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 command center database, during data processing, for use in Payment. Please see the *Administration 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 edocs 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 command center database, during data processing, for use in Payment. Please see the *Administration 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 edocs Index Table - The name of the variable defined by the DefTool for minimum amount due, which must be indexed by the command center. 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 Payment 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 Payment 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 Payment 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 **`$EDX_HOME`***/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 command center user information, which is determined by the type of enrollment supported. The options are:

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

or

com.edocs.payment.payenroll.usracct.JNDIMultipleDDNUserAccountAccessor, for when multiple DDN command center 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 Payment user information. The options are:

com.edocs.payment.payenroll.payacct.SSOPaymentAccountAccessor, for when Payment 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.

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

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**.

Enable verisign address verification service - Y (default) enables address verification for credit card payments. AVS support must also be set up with Verisign. See the section about AVS on page 20 for more information.

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

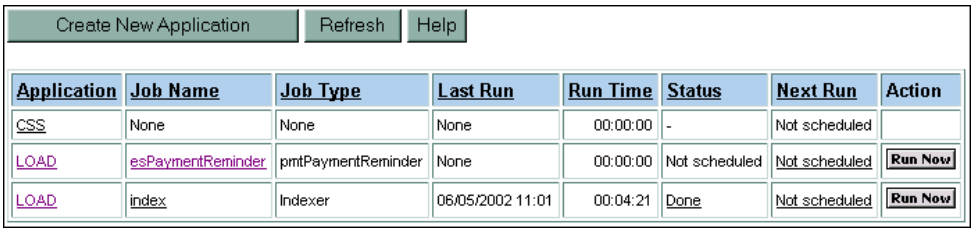
For information about implementing this class, contact edocs Professional Services. The default is: *com.edocs.payment.cassette.verisign.VeriSignCreditCardSubmitPlugIn*.

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

Updating a Payment Gateway Configuration

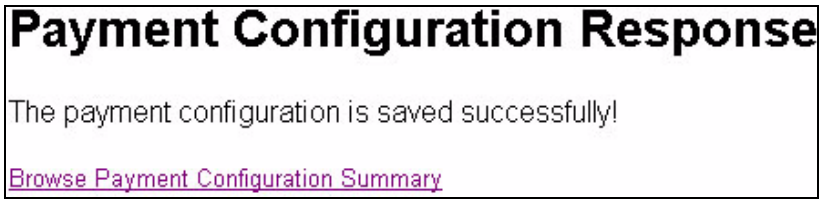
To update a payment gateway's settings:

1. Click **`Settings`** from the Command Center menu. The Settings page appears.
2. Click the **`Payment 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:

Table Column Definitions

5 Payment Tables

CHECK_PAYMENTS

This table saves check payment information.

Name	Null?	Type	Description
PAYMENT_ID	NOT NULL	NUMBER(28)	Unique for each check. It's time stamp value.
LAST_MODIFY_TIME	NOT NULL	DATE	The last time this check was updated.
CREATE_TIME	NOT NULL	DATE	The time when this check is created.
NEEDS_BACKUP	NOT NULL	CHAR(1)	Not used.
BILL_ID		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 performance tuning. It is configured by Payment 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.
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 matches a returned check from the ACH file to the database, when it is not possible to populate a check payment ID into the ACH file.
TXN_START_DATE		DATE	For ACH, this is the effective batch entry date for the check.
TXN_END_DATE	NULL	DATE	Reserved.
PAYMENT_SOURCE	NOT NULL	CHAR(1)	R/S: "S" means paid from UI, "R" means paid from recurring payment
FLEXIBILE_FIELD_1		VARCHAR2(255)	Flexible field for customization.
FLEXIBILE_FIELD_2		VARCHAR2(255)	Flexible field for customization.
FLEXIBILE_FIELD_3		VARCHAR2(255)	Flexible field for customization.

Name	Null?	Type	Description
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. Payment 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
PAYER_ACCOUNT_ NUMBER		VARCHAR2(40)	User's account number with the biller.
CARD_HOLDER_ NAME		VARCHAR2(40)	Name on the card.

Name	Null?	Type	Description
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 cardholder.
CARD_CITY		VARCHAR2(40)	City of cardholder.
CARD_STATE		VARCHAR2(40)	State of cardholder.
CARD_ZIP		VARCHAR2(40)	Zip code of cardholder.
CARD_COUNTRY		VARCHAR2(40)	Country of cardholder.
PAY_DATE	NOT NULL	DATE	Payment date.
AMOUNT	NOT NULL	NUMBER(28,2)	Payment amount.
STATUS	NOT NULL	NUMBER(2)	Payment status.
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 Payment 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 Payment 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)	Identifies this payment account.
USER_ID	NOT NULL	VARCHAR2(40)	The user who owns this payment account.
DDN	NULL	VARCHAR2(18)	The DDN name, used for ACH pre-note
PAYMENT_TYPE	NOT NULL	VARCHAR2(10)	Either “check” or “ccard”
ACCOUNT_HOLDER_ NAME	NOT NULL	VARCHAR2(40)	The customer’s name for the payment account.
ACCOUNT_NUMBER	NOT NULL	VARCHAR2(255)	The customer’s payment account number.
ACCOUNT_TYPE	NOT NULL	VARCHAR2(40)	For check: “checking” or “saving”. For credit card: the card type, such as “visa”, "AMEX", etc.
ACCOUNT_USAGE	NULL	VARCHAR2(40)	“personal” or “business”.
ACCOUNT_STATUS	NULL	VARCHAR2(40)	Can be “active”, “inactive”, "bad_active", "pnd_active", "pnd_wait".
ROUTING_TRANSIT	NULL	VARCHAR2(9)	For check: the check routing number.
EXPIRATION_DATE_FORMAT	NULL	VARCHAR2(20)	The date format of the credit card account expiration date.
EXPIRATION_DATE	NULL	DATE	The date when the payment account expires.
STREET	NULL	VARCHAR2(255)	Billing address.
CITY	NULL	VARCHAR2(40)	Billing address.
STATE	NULL	VARCHAR2(40)	Billing address.
ZIPCODE	NULL	VARCHAR2(40)	Billing address.
COUNTRY	NULL	VARCHAR2(40)	Billing address.
NOTIFY_SOURCE	NULL	CHAR(1)	Used for ACH prenote notification.
NOTIFY_STATUS	NULL	CHAR(1)	For ACH prenote notification. Indicates whether the payment account has been notified.
TXN_MESSAGE	NULL	VARCHAR2(255)	Contains the error message for ACH prenote or NOC.
TXN_DATE	NULL	DATE	For ACH prenote. Date of the transaction happens.
FLEX_FIELD_1	NULL	VARCHAR2(255)	Used for customization.
FLEX_FIELD_2	NULL	VARCHAR2(255)	Used for customization.

Name	Null?	Type	Description
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.
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. Payment 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 the command center.

Name	Null?	Type	Description
INV_ID	NOT NULL	NUMBER(28)	Unique invoice ID generated by Payment.
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 Payment and it can be customized.
INV_NUMBER	NULL	VARCHAR2(40)	A string assigned by the biller to identify this invoice. Not used by Payment, 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 Payment, so it can be used for customization.
INV_ISSUER	NULL	VARCHAR2(40)	The entity that issued the invoice. Not used by Payment, so it can be used for customization.
INV_MEMO	NULL	VARCHAR2(250)	Invoice memo. Not used by Payment, so it can be used for customization.
INV_ISSUER	NULL	VARCHAR2(40)	The entity issues the invoice. Not used by Payment, 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 Payment, 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 Payment, so it can be used for customization.
TRANSACTION_ DATE	NULL	DATE	Invoice transaction date. Not used by Payment, so it can be used for customization.

Name	Null?	Type	Description
FLEXIBLE_FIELD_1	NULL	VARCHAR2(255)	Flexible field. Not used by Payment, so it can be used for customization.
FLEXIBLE_FIELD_2	NULL	VARCHAR2(255)	Flexible field. Not used by Payment, so it can be used for customization.
FLEXIBLE_FIELD_3	NULL	VARCHAR2(255)	Flexible field. Not used by Payment, so it can be used for customization.
FLEXIBLE_FIELD_4	NULL	VARCHAR2(255)	Flexible field. Not used by Payment and is customizable.
FLEXIBLE_FIELD_5	NULL	VARCHAR2 (1000)	Flexible field. Not used by Payment, so it can be used for customization.
BILL_ID	NULL	VARCHAR2(255)	The bill id associated with the invoice.

PAYMENT_PROFILE

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

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

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

RECURRING_PAYMENTS

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

Name	Null?	Type	Description
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.
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 Payment, so it can be used for customization.
FLEX_FIELD_2		VARCHAR2(255)	Flexible field. Not used by Payment, 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
check_payments	nuk_check_payments_2	status, reminded

Table name	Index name	Indexed columns
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

Payment Database Migration

The Payment database is designed to migrate from a previous version to new version, whenever Payment upgrades the database schema. If you have a payment database from an older version of Payment, 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 Payment database depends on the *document_definition_name* table from the command center, it's very important to make sure that this table is migrated, too. If a DDN name/reference is removed or changed during migration, that can cause problems for Payment.

Index

A

ACH
- change codes, 13
- effective date, 15
- effective date and holidays, 47
- federal holidays, 49
- File Input Directory, 48
- File Output Directory, 48
- immediate destination, 47
- immediate origin, 47
- return codes, 14
- return directory, 48
- settlement date, 15
- submit directory, 48
- Template Directory, 48

Amount Due Format, 45, 50, 54

B

bank holidays
- listed, 49

Batch Size for Credit Card Payment Table, 55

Batch Size for Payment Reminder Table, 49, 53
- gateway parameter, 44

Batch Size for Payment Table, 46, 51

C

CDA
- configuration, 46

CDA configuration, 46

CDP
- File Input Path, 52
- File Output Path, 52

Check Payment Gateway
- overview, 44
- parameters, 44

check payments
- clearing checks, 47
- transaction cycle, 11
- transaction statuses, 12

CheckFree
- gateway configuration, 49

Client ID, 52

Company Entry Description, 48

Company ID, 48

Company Name, 48

credit card
- Batch Size for Credit Card Payment Table, 55
- overview, 17
- Payment Gateway, 53
- scheduled payment logic, 19
- statuses, 17
- transaction overview, 18
- user options, 20
- Verisign configuration, 20
- Verisign overview, 20

D

Days to Activate Pending Subscribers, 47, 51

Days to Clear Checks, 47, 51

Due Date Format, 50, 54

gateway parameter, 45

E

email notification
Email Template for Job Status Notification, 54
job status configuration, 50
Mail Addresses for Job Status Notification, 54
Mail Server for Job Status Notification, 54
Send Email Notification when Payment Jobs are Done, 54
Email template for job status notification, 46, 51
enrollment
configuring activation delay, 47
configuring for payment information, 55
configuring for user information, 55
Enrollment XML file(for flat enrollment only), 46, 55
Implementation of IPaymentAccountUserAccessor, 55
Implementation of IUserAccountUserAccessor, 55
payment configuration, 51
user configuration, 51

F

File Transmission Mode, 52
Flexible Field 1 and 2
ACH, 48
CheckFree, 56
FTP User Name, 52

G

Gateway
ACH gateway parameter, 44
CheckFree gateway parameter, 49
Generate empty ACH file when there are no checks to submit, 47
Generate Empty File, 52

H

Help
technical support, 6
holidays
federal, 49

I

Immediate Destination, 47
Immediate Destination Name, 47
Immediate Origin, 47
Immediate Origin Name, 48
Implementation of ICheckSubmitPlugIn, 48
Implementation of IPaymentAccountUserAccessor, 46, 51
Implementation of IUserAccountUserAccessor, 46, 51
instant payments
logic, 18

J

job status
Email Template for Job Status Notification, 54
Mail Addresses for Job Status Notification, 54
Mail Server for Job Status Notification, 54
Send Email Notification when Payment Jobs are Done, 54

M

Mail server for job status notification, 46, 51

Mail-to addresses (separated by ";", semicolon) for job status notification, 46, 51
minimum amount due
Minimum Amount Due Format, 54
Name of Minimum Amount Due in Index Table, 54
Minimum Amount Due Format, 50
Minimum Amount Due Format, 45

N

Name of Amount Due in Index Table, 50, 54
Name of Amount Due in Index Table, 45
Name of Due Date in Index Table, 50, 54
Name of Due Date in Index Table, 45
Name of Minimum Amount Due in Index Table, 50
Name of Minimum Amount Due in Index Table, 45
noc
enrollment update, 47
NOC
codes, 13

O

ODFI
ACH payment gateway setting, 48

P

Payee Address, 52
Payee City, 52
Payee Name, 52
Payee Number, 52
Payee Short Name, 52
Payee State, 52
Payee Zip, 52
Payment gateway, 42
configuring, 41
deleting, 57
updating, 56
Payment Gateway
check overview, 44
plugin
ACH gateway, 48
pmtCheckSubmit
ACH company name, 48
ACH immediate origin, 47
and ACH effective date, 15
company entry description, 48
date, 15
empty ACH configuration, 47
empty CheckFree configuration, 52
immediate destination, 47
ODFI configuration, 48
pmtCheckUpdate
ACH company name, 48
ACH immediate destination, 47
ACH immediate origin, 47
and ACH change codes, 13
and ACH return codes, 14
enrollment and NOC, 47
pmtCreditCardSubmit
and the payment transaction cycle, 19
pmtNotifyEnroll
ACH NOC, 47

R

recurring payments
configuring, 34
Recurring Payments
overview, 23

S

Send Email Notification in Case of NOC, 47

Send Email Notification when Payment Jobs are Done (with or without error), 50
Send Email Notification when Payment Jobs are Done(with or without error), 46
Sender ID
CheckFree parameter, 52
Skip non-business days for batch effective entry date, 47

T

template
CheckFree, 52

U

Update Payment enrollment in Case of NOC, 47

V

Verisign
Certificate Path, 56
configuration, 20
Host Name, 55
Host Port, 55
Implementation of IVeriSignCreditCardSubmitPlugIn, 56
Number of Threads, 56
Partner, 56
Password, 56
Timeout Period for Transactions, 55
User, 55
Vendor, 56