Administration Guide for Oracle Self-Service E-Billing

Version 6.0.4, Rev. A

December 2011

ORACLE

Contents

Chapter 1: What's New in This Release

Chapter 2: Enrolling and Logging In to the Command Center

Enrolling a System Administrator User 11

Creating a Password for Logging In to the Command Center 12

Logging In to the Command Center 13

Changing an Administrator's Password or Personal Information 13

Deactivating and Reactivating the Secure Administrator ID 14

Logging Out of the Command Center 14

Chapter 3: Creating Applications and Jobs

Creating a New Application 15

About Oracle Self-Service E-Billing Jobs 16

Creating a New Job 16

Changing a Job Configuration 17

Deleting a Job 17

About Data Protection 17

Chapter 4: Configuring Jobs

Billing Job Types 19

- Configuring a StatementReady Job 20
- Configuring a Notifier Job 20
- Configuring a BatchReportScheduler Job 25
- Configuring a PasswordExpNotify Job 25
- Configuring a Purge Logs Job 26
- Configuring a ReportCleanUp Job 26
- Configuring a ShellJob 27
- Configuring a HierarchyImporter Job 27
- Configuring a HierarchyCopy Job 28
- Configuring a HierarchyCleanUp Job 29

Configuring a HierarchyPurge Job 30
Payment Job Types 30
Configuring a pmtRecurringPayment Job 32
Configuring a pmtSubmitEnroll Job 35
Configuring a pmtConfirmEnroll Job 36
Configuring a pmtPaymentReminder Job 36
Configuring a pmtCheckSubmit Job 38
Configuring a pmtCheckUpdate Job 41
Configuring a pmtARIntegrator Job 45
Configuring a pmtAllCheckTasks Job 46
Configuring a PaymentDueNotification Job 47
Configuring a pmtCreditCardSubmit Job 48
Configuring a pmtPaymentRefund Job 49
Configuring a ThresholdExceedNotify Job 54
Configuring a pmtCreditCardExpNotify Job 54
Configuring a pmtCustom Job 55

Chapter 5: Administering the Live Production Process

Administering Jobs 57
Monitoring Production Jobs 58
Viewing Job Status 59
Viewing and Verifying Task Status 60
Canceling and Rerunning Failed Jobs 62
Starting a Job Manually 63

Chapter 6: Scheduling Jobs

Scheduling Jobs 65
Process of Configuring a Blackout Calendar 67
Creating a Blackout Calendar 68
Editing a Blackout Calendar 68
Deleting a Blackout Calendar 69
Applying a Blackout Calendar to a Job Schedule 69
Process of Configuring Job Alerts 69
Creating Alert Groups 70
Editing Alert Groups 70
Deleting Alert Groups 71
Creating Alert Profiles 71

Updating an Alert Profile 72
Deleting an Alert Profile 73
Applying Alerts to a Job Schedule 73

Chapter 7: Configuring the Payments Module

About the Payment Module Features 75
About the Transaction Manager and Payment Cartridges 76
Scheduling Payment Jobs 77
About Email Notifications 77
Setting Up PayPal Payflow Pro Certification (IBM WebSphere Users Only) 78
Configuring a Payment Gateway 79

Chapter 8: Payment Transactions

About Check Payment Transactions 93
Adding a New Customer Account for Check Payment Services 93
The Cycle of an ACH Check Payment Transaction 94
About Check Payment Status Flow 98
About Credit Card Payment Transactions 98
About Instant Credit Card Payments 99
About Scheduled Credit Card Payments 100
About Credit Card Payment Status 101
About the Address Verification Service 101
Viewing Payment Reports 102

Chapter 9: Recurring Payments

About Recurring Payments 103
About the Multiple Recurring Payment Feature 104
About the Recurring Payment Transaction Cycle 104
Database Tables Affected by Recurring Payments 106
Examples of Recurring Payment 107

Chapter 10: Reviewing Production Activity

About Job Reports 117
About Message Log Files 118
Monitoring Service Status 119

About Administrator Activity Auditing 120

Chapter 11: Administering the Database

Managing the Payment Module Database 121

Setting Enrollment Properties 126

Deleting an Application 126

About Purging Data 127

Purging Payment Account and Transactional Data 128

Purging Hierarchies and Hierarchy Assignments 134

Purging User Data 136

Purging Statement and Invoice Fact Data 138

Purging Validation Codes 139

Purging Administrator Activity 140

Purging Locked Administrators in Command Center 140

Purging Data in Batch 141

Process of Purging Sample Data 142

Auditing Purged Data 142

Running the Master Key Update 143

Chapter 12: Running the ETL Jobs Using Oracle Workflow Manager

Running the ETL_SUPER_PF_2 (ETL Loader) Job 147

Running the OLAP_ACCEPT_REJECT Job 147

Retrying the OLTP_LD_2 Job Process 150

Appendix A: Error Messages

Command Center Job Error Messages 153

Payment Error Messages 159

Index

1 What's New in This Release

What's New in Administration Guide for Oracle Self-Service E-Billing, Version 6.0.4, Rev. A

Table 1 lists changes described in this version of the documentation to support release 6.0.4, Rev. A of the software.

Table 1. New Product Features in Administration Guide for Oracle Self-Service E-Billing, Version 6.0.4, Rev. A

Topic	Description
"Scheduling Jobs" on page 65	New topic. Added instructions for scheduling production jobs in the Command Center Scheduler.
"Process of Configuring Job Alerts" on page 69	Modified topic. Added procedure for applying job alerts in Scheduler.
"Running the ETL_SUPER_PF_2 (ETL Loader) Job" on page 147 "Running the OLAP_ACCEPT_REJECT Job" on page 147 "Retrying the OLTP_LD_2 Job Process" on page 150	Modified topics. Clarified procedures for running the Extract Transform Loading (ETL) jobs in Oracle Warehouse Builder.

What's New in Administration Guide for Oracle Self-Service E-Billing, Version 6.0.4

Table 2 lists changes described in this version of the documentation to support release 6.0.4 of the software.

Table 2. New Product Features in Administration Guide for Oracle Self-Service E-Billing, Version 6.0.4

Topic	Description
"About Data Protection" on page 17	Modified topic. Added the new ThresholdExceedNotify job to the list of Payment job types available in the Command Center.
"Configuring a ThresholdExceedNotify Job" on page 54	New topic. Describes the new ThresholdExceedNotify job.

What's New in Administration Guide for Oracle Self-Service E-Billing, Version 6.0.3

Table 3 lists changes described in this version of the documentation to support release 6.0.3 of the software.

Table 3. New Product Features in Administration Guide for Oracle Self-Service E-Billing, Version 6.0.3.

Topic	Description
"About Data Protection" on page 17	Modified topic. Changed the name of the ShellCommand job to Shell job.
"About Administrator Activity Auditing" on page 120	New topic. Describes the new administrator activity audit.
Chapter 2, "Enrolling and Logging In to the Command Center"	New chapter. Describes the new secure administrator enrollment and login procedures for compliance with the Payment Card Industry Data Security Standard (PCI DSS). Each administrative user must now have unique personal login credentials.
"Configuring a HierarchyCleanUp Job" on page 29	Modified topic. Clarified the syntax of the datasource example.
"Parameters for Configuring Global Payment Settings" on page 80 "Adding a New Customer Account for Check Payment Services" on page 93 "About Check Payment Status Flow" on page 98 "About Instant Credit Card Payments" on page 99	Modified topics. Added content to support compliance with the Payment Card Industry Data Security Standard (PCI DSS) and changes to the schema.
"Configuring the Payments Module" on page 75	Removed topics to support modifications for compliance with the Payment Card Industry Data Security Standard (PCI DSS) and changes to the schema.

Table 3. New Product Features in Administration Guide for Oracle Self-Service E-Billing, Version 6.0.3.

Topic	Description
"About Purging Data" on page 127 "Purging Payment Account and Transactional Data" on page 128 "Purging Hierarchies and Hierarchy Assignments" on page 134 "Purging User Data" on page 136 "Purging Statement and Invoice Fact Data" on page 138 "Purging Validation Codes" on page 139 "Purging Administrator Activity" on page 140 "Purging Locked Administrators in Command Center" on page 140 "Purging Data in Batch" on page 141 "Process of Purging Sample Data" on page 142 "Auditing Purged Data" on page 142	New topics. Content describes the new script for purging various types of data from the Oracle Self-Service E-Billing database.
"Managing the Payment Module Database" on page 121	Removed database table schemas.
"Running the Master Key Update" on page 143	New topic. Describes how to update the master key, required if you loaded sample data before going live, and also required to run once a year to remain in compliance with the Payment Card Industry Data Security Standard (PCI DSS).
Use of Checkfree payment gateway removed.	References to using Checkfree payment gateway have been removed because it is no longer available with this product.

Table 4 lists changes previously made in this guide to support release 6.0.2 of the software.

Table 4. New Product Features Added in Administration Guide for Oracle Self-Service E-Billing, Version 6.0.2

Topic	Description
"Setting Up PayPal Payflow Pro Certification (IBM WebSphere Users Only)" on page 78	New topic. This topic describes how to set up PayPal Payflow Pro certification (required on IBM WebSphere only).
"Parameters for Configuring Check and Credit Card Gateways" on page 83	New fields have been added, class names updated, and other fields renamed for use in configuring a PayPal Payflow Pro credit card gateway for a regular payee DDN.

2 Enrolling and Logging In to the Command Center

This chapter describes how to enroll as a system administrator user and log in to the Command Center. It includes the following topics:

- Enrolling a System Administrator User on page 11
- Creating a Password for Logging In to the Command Center on page 12
- Logging In to the Command Center on page 13
- Changing an Administrator's Password or Personal Information on page 13
- Deactivating and Reactivating the Secure Administrator ID on page 14
- Logging Out of the Command Center on page 14

Enrolling a System Administrator User

Oracle Self-Service E-Billing comes with a preconfigured, secure administrator ID and password for initial access. For help with the preconfigured administrator ID and password, contact your Oracle sales representative to request assistance from Oracle's Professional Services.

Each system administrator must enroll and create a unique administrator ID and password. For information about setting the enrollment properties, see "Setting Enrollment Properties" on page 126.

After accessing Oracle Self-Service E-Billing initially, you must immediately enroll, creating a personal administrator user ID.

NOTE: Preserve the secure preconfigured administrator ID and password. You can never change them. A database administrator (DBA) can disable the default access account and can reactivate it.

To enroll the initial system administrator user

1. Start your Internet browser, and enter the URL for the Command Center.

   https://*localhost*:*7003*/eBilling

   where:

   - *localhost* is the name of the Command Center application server.
   - *7003* is the port number of the Command Center application server.
2. On the Login Administrator page, enter the secure, preconfigured administrator ID and password provided by your Oracle representative. Click Submit.
3. At the Enter Administrator's User Information page, enter a new value for each of the following fields to create a new system administrator user:

   - Administrator ID (You cannot use the preconfigured administrator ID.)

- First Name
- Last Name
- Email Address
- Confirm Email Address

4 Confirm the email address, and click Submit.

Creating a Password for Logging In to the Command Center

An enrolled system administrator user must create a password to access the Command Center.

To create a password for logging in to the Command Center

1 Start your Internet browser, and enter the URL for the Command Center.

This URL was configured when Oracle Self-Service E-Billing was installed, for example

https://*localhost*:*7003*/eBilling

where:

- *localhost* is the name of the Command Center application server.
- *7003* is the port number of the Command Center application server.

2 On the Login Administrator screen, click the First Time Login link.

3 On the Validate Administrator's Information page, enter the administrator ID and email address that you entered for the enrollment, then click Submit.

4 On the Enter Administrator's User Information page, enter a contact phone number and a password. For the contact number, only numbers and dashes are allowed (the plus sign is not allowed).

By default, the password must contain a minimum of eight characters and the following:

- At least one uppercase character
- At least one lowercase character
- At least one number
- No spaces

The following special characters are not valid: |, $, +, (,), <, >, {, }, [,].

The password cannot be either of the following:

- The initial system administrator ID
- The previously entered or preconfigured password

5 Confirm the password, and click Submit.

Logging In to the Command Center

After you have enrolled as a system administrator user and have set a password, complete the following task to log in to the Command Center.

Logging in to the Command Center displays the Main Console, where you can configure jobs and use the Command Center to schedule and run production jobs, monitor live production activity, and perform other system administration activities.

To log in to the Command Center

1 Start your Internet browser, and enter the URL for the Command Center.

This URL was configured when Oracle Self-Service E-Billing was installed, for example

https://*localhost*:*7003*/eBilling

where:

- *localhost* is the name of the Command Center application server.
- *7003* is the port number of the Command Center application server.

2 On the Login Administrator page, enter the administrator ID and password, and click Submit.

The Command Center Main Console appears, and you can begin working. If your account password is older than the configured number of days, you must create a new password when prompted.

If you try to enter an incorrect administrator ID or password more than the configured number of times that is allowed, Oracle Self-Service E-Billing locks your account and you must create a new administrator user. For details on creating a new administrator user, see "Enrolling a System Administrator User" on page 11.

Changing an Administrator's Password or Personal Information

A system administrator can change their password for logging in to the Command Center at any time. An administrator's password will automatically expire after the configured number of days.

Administrators can also update their first and last name, contact number, and email address.

NOTE: You can use the PasswordExpNotify job to send an email to warn enrolled administrators that their password is set to expire. For information on the PasswordExpNotify job, see "Configuring a ThresholdExceedNotify Job" on page 54.

To change the administrator's password or personal information

1 On the Command Center Main Console, click Settings.

2 Click the Admin Login tab.

3 On the Enter Administrator's User Information page, edit the personal information as needed. For the contact number, only numbers and dashes are allowed (the plus sign is not allowed).

To change the password, enter the new password in the Password field. Enter the password again in the Confirm Password field.

4 Click Submit.

Deactivating and Reactivating the Secure Administrator ID

You can deactivate and reactivate the secure administrator ID when needed.

To disable or reactivate the secure Command Center administrator ID

1 Log on to the Oracle Self-Service E-Billing OLTP instance using SQL*Plus (not as SYSDBA):

OLTP schema username/*OLTP schema password*@*OLTP TNS name*

where:

- *OLTP schema username* is the name of the OLTP schema user.
- *OLTP schema password* is the password of the OLTP schema user.
- *OLTP TNS name* is the TNS name for the OLTP instance.

2 To deactivate the secure administrator ID, run the following command:

```
SQL>exec disable_cc_default_admin;
```

To reactivate the secure administrator ID, run the following command:

```
SQL>exec enable_cc_default_admin;
```

Logging Out of the Command Center

Always log out of the Command Center after completing a session. By logging out, you help to maintain the security of the production environment and minimize the risk that an application or job might be accidentally corrupted or destroyed.

To log out of the Command Center

- Click Logout on the Main Console.

3 Creating Applications and Jobs

This chapter describes how to create an application and jobs in the Command Center. The *Command Center* is the Oracle Self-Service E-Billing production application where you configure, schedule, and manage billing and payment jobs. It includes the following topics:

- Creating a New Application on page 15
- About Oracle Self-Service E-Billing Jobs on page 16
- Creating a New Job on page 16
- Changing a Job Configuration on page 17
- Deleting a Job on page 17
- About Data Protection on page 17

Creating a New Application

You must create an application in the Command Center for each online Billing and Payment application designed for your Oracle Self-Service E-Billing implementation.

About Mapping Your Application to a Data Source EJB

The Data Source Name is the real, global JNDI name as opposed to the local JNDI name (java:comp/env/…). The data source EJB exists in a separate presentation EAR file. To successfully create the application, the JNDI name must exist and the EJB must be properly deployed and available to application. The Command Center validates the JNDI name before the mapping is persisted.

You must specify a data source EJB for each application (DDN) you create in the Command Center. When creating an application in the Command Center, a data source refers to an EJB in your application (EAR file) that specifies summary information and location of your document data. For information on developing a custom data source EJB, consult your Oracle Professional Services representative.

Specifying the data source EJB at the DDN level allows you to set the JNDI mapping without modifying deployment descriptors, repackaging, and redeploying your Web application. It also enables you to retrieve, for example, live data from an external database or archival data from offline storage. In some cases, customizing the data source can also improve performance and save disk space.

To create a new application

1 At the Command Center, click Create New Application.

2 Enter the name of the application. The first character in the name must be an alpha, and the rest of the name can contain alphanumeric characters and underscores, but no spaces, such as testApp.

3 Enter the JNDI name of the data source EJB to use for this application, such as `edx/ejb/EdocsDataSource`.

4 Ignore the Index Partition Count. Click Create Application and Continue.

5 At the Create New Job screen, proceed to configure jobs for your application.

About Oracle Self-Service E-Billing Jobs

There are two categories of Oracle Self-Service E-Billing jobs:

- **Command Center production jobs**. You must create, run, and manage various production (batch) jobs for your Oracle Self-Service E-Billing application using the Command Center. You must determine which production jobs you need for your implementation. For a list of available production jobs, see "Billing Job Types" on page 19. For a list of available Payment jobs, see "Payment Job Types" on page 30.

  NOTE: Command Center does not automatically schedule jobs to run. You must manually specify job schedules for production.

- **Extract Transform Loading (ETL) jobs**. You must run and manage Extract Transform Loading (ETL) jobs using Oracle Workflow Manager (OWF) and the Design Center in Oracle Warehouse Builder. For instructions on setting up and running ETL jobs, see Chapter 12, "Running the ETL Jobs Using Oracle Workflow Manager."

Creating a New Job

You must create and configure each job your particular application requires.

To configure a new job

1 On the Command Center Main Console, click on the application name link.

2 Click Add a New Job.

3 Provide a descriptive name for the job.

4 From Job Type menu select the type of job you want to create.

5 Click Configure Job and then Continue.

6 Specify the task configuration parameters for the particular job type (if any). For details on the parameters for each job type, see "Configuring Jobs" on page 19.

7 Click Submit Changes and Schedule. Click OK.

You can schedule when to run the job or you can click Run Now to run the job immediately.

Changing a Job Configuration

You can change a job configuration any time, except while the job is processing.

CAUTION: If you try to save configuration changes while a job is running in the production queue (job status is Processing), the new job configuration parameters are ignored.

To change a job configuration

1. On the Main Console, click the name of the job you want to reconfigure.

   Command Center displays the job configuration screen.

2. Enter your changes. If you want to clear all current job parameters, click Reset.
3. Click Submit Changes and Schedule. Click OK.

   Command Center displays the Schedule screen where you can edit the job schedule, if needed.

Deleting a Job

You can delete a job you no longer need in an application. Deleting a job removes the job configuration and schedule in the Command Center.

Deleting a job does not remove data associated with the job that is already in the database. (Use Purge Logs jobs to purge data.)

Make sure you really want to delete the job; you can always cancel a job, or change its configuration or schedule.

To delete a job

1. On the Main Console, click the name of the application. Command Center displays the Edit Application screen, which lists the application jobs.
2. Click the box in the Delete? column for the job.
3. Click Delete Marked Jobs. Click OK when asked if you are sure you want to delete the marked job.

   Oracle Self-Service E-Billing removes the job from the application.

About Data Protection

Oracle Self-Service E-Billing works with Oracle Data Guard to provide solutions for data protection and recovery from hardware crash or data corruption.

Oracle Self-Service E-Billing uses two database instances for data storage, one for storing historical data for analytics (OLAP) and another for transactional data (OLTP). Certain transactional data is created from the OLTP side of the database and is synchronized with OLAP. The synchronization allows real-time analysis based on the changes made in OLTP, of which hierarchy structure is one example. Hierarchy can be created by online users. The newly created hierarchy is stored in the OLTP database and then pushed to OLAP. The analytic reports then can be generated based on the newly updated hierarchy structure.

It is possible to set up a primary database and a standby database for both the OLAP instance and the OLTP instance. Each database works independently with Oracle Data Guard. The integrity of the data is guaranteed in Oracle Self-Service E-Billing through the design of the ETL process as well as by using distributed database transaction management.

Data created from the ETL process is first loaded to the OLAP database. After the data is committed and validated in the OLAP database, some key data is populated into the OLTP exchange tables. From there onward, the ETL process continues to populate the OLTP production tables. In cases where the OLAP or OLTP databases go down, the counterpart of each database keeps the data in its entirety.

For data created by online users, the distributed database transactions are used to guarantee data integrity. If one of the databases goes down, no new data is written to either of the databases.

4 Configuring Jobs

This chapter describes the Command Center production jobs and configuration parameters in Oracle Self-Service E-Billing. It includes the following topics:

- Billing Job Types on page 19
- Payment Job Types on page 30

Billing Job Types

Table 5 lists the billing production jobs available with Oracle Self-Service E-Billing.

NOTE: If the configuration parameters change for a job, you must recreate and reconfigure the job.

Table 5. Available Production Jobs

Job Name	Description
StatementReady	Sends email to enrolled users who are linked to accounts that have new statements available in Oracle Self-Service E-Billing. For details on configuring a StatementReady job, see "Configuring a StatementReady Job" on page 20.
Notifier	Sends the emails stored in the MESSENGER_GROUP_TABLE (and waiting to be sent) to users. You run the Notifier job after the Payment Reminder, Payment Due Notification, Statement Ready, and pmtCCExpiration jobs. For details on configuring a Notifier job, see "Configuring a Notifier Job" on page 20.
BatchReportScheduler	Processes batch reports as requested by the reporting server. Running the BatchReportScheduler job starts processing all scheduled batch reports, making them available online. For details on configuring a BatchReportScheduler job, see "Configuring a BatchReportScheduler Job" on page 25.
PasswordExpNotify	Generates a warning email to system administrators whose password is set to expire within the (next) configured number of days. For details on configuring a PasswordExpNotify job, see "Configuring a PasswordExpNotify Job" on page 25.
Purge Logs	Removes historical data from the Log table. Running the Purge Logs job periodically removes data for all applications you have, freeing up space on your database server. For details on configuring a Purge Logs job, see "Configuring a Purge Logs Job" on page 26.

Table 5. Available Production Jobs

Job Name	Description
ReportCleanUp	Deletes batch report files and related records from the database. For details on configuring a ReportCleanUp job, see "Configuring a ReportCleanUp Job" on page 26.
ShellJob	Provides a way to run a custom shell command (shell script or external script), executable, or other program that was written to perform a task specific to your requirements using the command line. You can process the output files from other tasks within the ShellJob job. For details on configuring a ShellJob, see "Configuring a ShellJob" on page 27.
HierarchyImporter	Reads XML files that define a hierarchy, builds that hierarchy in the online transactional processing (OLTP) database, and synchronizes with the online analytical processing (OLAP) database. For details on configuring a HierarchyImporter job, see "Configuring a HierarchyImporter Job" on page 27.
HierarchyCopy	Replicates all published hierarchies for the periods up to the current periods. For details on configuring a HierarchyCopy job, see "Configuring a HierarchyCopy Job" on page 28.
HierarchyCleanUp	Cleans up closed accounts and unsubscribed services, and removes them from hierarchies. For details on configuring a HierarchyCleanUp job, see "Configuring a HierarchyCleanUp Job" on page 29.
HierarchyPurge	Cleans up closed accounts and unsubscribed services, and removes them from hierarchies. For details on configuring a HierarchyPurge job, see "Configuring a HierarchyPurge Job" on page 30.

Configuring a StatementReady Job

The StatementReady job sends email to enrolled users whose accounts have new statements that are ready in Oracle Self-Service E-Billing. When this jobs runs, it reads information passed from the ETL process and finds all the users whose account statement has recently been loaded and processed, and sends email to the corresponding users. Email messages are sent only after running the Notifier job.

There are no configuration parameters for the StatementReady job.

Configuring a Notifier Job

The Notifier job extracts the email address registered for each bill-ready account and composes a single, consolidated email for the address. The Notifier job then sends the composed email to the recipients.

To send queued email messages, you must run Notifier after any job that was configured to generate email notifications. You can configure Notifier to run for all notification types or for one particular type of message only, such as:

- Payment reminders
- A payment is due
- A statement is ready
- A user's credit card expires
- User enrolls in an account
- User updates a personal profile
- User creates, updates or deletes a payment account
- User makes an instant payment
- User schedules a one-time or recurring payment

The types of notifications you can send are listed in the Notifier task parameters.

The Notifier job sends email based on entries that are stored in the MESSENGER_GROUP_TABLE table in the OLTP database, composes the correct email message based on email templates, and sends it to the users to whom the messages are addressed.

The Notifier job consists of two tasks:

- NotificationComposer
- NotificationDispatch

Configuring the NotificationComposer Task

The NotifierComposer task extracts the email address registered for each bill-ready account and composes a single, consolidated email for the address. It writes the email message to the location specified and stores the name of the email file and the email address in MESSENGER_QUEUE_TABLE.

This task references the MESSENGER_GROUP_TABLE table. For each entry with a value in MESSAGE_TYPE matching the Message type parameter value, the task composes the corresponding email message and stores the composed message in the local message repository. The repository file directory is specified in the *EDX_HOME*/xma/config/com/edocs/common/notification/notification.xma.xml file, in the GlobalConfigurationBean section with the bean configuration property value of mailQueueStorageDirectory. In the path, *EDX_HOME* is the directory where you installed Oracle Self-Service E-Billing. The location of the message file is in the MESSENGER_QUEUE_TABLE table.

To send and receive email, you must configure the Notification.xma.xml file for the email address and password (mail.user and mail.password values).

To configure the Notification.xma.xml file

- Edit the notification.xma.xml file, found in the *EDX_HOME*/xma/config/com/edocs/common/notification directory. Specify the mail.user and mail.password values:

```
<prop key="mail.host">stbeehive.yourcompanydomain.com</prop>
              <prop key="mail.transport.protocol">SMTP</prop>
                  <!-- For requiring authentication mail server -->
                  <prop key="mail.smtp.auth">true</prop>
              <prop key="mail.user">eBillingAdmin_WW@yourcompanydomain</prop>
                  <prop key="mail.password">eBilling603</prop>
                  <!-- For SSL connection mail server-->
                  <prop
key="mail.smtp.socketFactory.class">javax.net.ssl.SSLSocketFactory</prop>
                  <prop key="mail.smtp.socketFactory.port">465</prop>
           </props>
```

Parameters for Configuring the NotificationComposer Task

Table 6 describes the configuration parameters for the NotificationComposer task.

Table 6. Parameters for Configuring the NotificationComposer Task

Parameter	What to Enter or Select
Skip task	When set to the default value N, the task is not skipped when running the Notifier job. Otherwise, the job is not executed.
Message type	This value specifies the types of messages that the Notifier job picks up when it runs. (This list comes from the notification.xma.xml configuration file, in the section of GlobalConfigurationBean with the bean configuration property value of notifierMessageTypes. The notification.xma.xml file is located in the *EDX_HOME*\xma\config\com\edocs\common\notification, directory, where *EDX_HOME* is the directory where you installed Oracle Self-Service E-Billing.) You must run the appropriate notification jobs before running Notifier.

Table 7 Describes the valid message types you can specify in the Message type field for the NotificationComposer task.

Table 7. Message Types

Message Type	Description
AllNotifications	All email notification types.
BillNotification	Statement-available notifications only.
CreditCardExpiryNotification	For pmtCreditCardExpNotify job notifications only.
BatchReportReadyNotification	This message is populated in the MESSENGER_GROUP_TABLE after running the batch report job successfully. Run the Notifier job to send the actual email to the recipients. Users who created the batch report receive the message.
PaymentDueNotification	For PaymentDueNotification job notifications only.
PaymentScheduledNotification	Notifications for scheduled payments from the pmtPaymentRecurringPayment job only.
PaymentSuccessNotification	Notifications for successful payments from the pmtPaymentReminder job only.
PaymentFailureNotification	Notifications from the pmtPaymentReminder job for failed payments only.
PaymentThresholdNotification	Notifications from the pmtPaymentRecurringPayment job for scheduled payments that have reached the user's threshold only.

Table 7. Message Types

Message Type	Description
RecurringPaymentNotification	This message is populated in the MESSENGER_GROUP_TABLE after a recurring payment has been set up from Oracle Self-Service E-Billing. Run the Notifier job to send the actual email message to the recipients. Users who have access to the billing account through billing hierarchy receive this email.
RecurringPaymentUpdateNotification	This message is populated in the MESSENGER_GROUP_TABLE after a recurring payment is updated. Run the Notifier job to send the actual email message to the recipients. Users who have access to the billing account through billing hierarchy receive this email.
RecurringPaymentDeleteNotification	This message is populated in the MESSENGER_GROUP_TABLE when a recurring payment is deleted. Run the Notifier job to send the actual email message to the recipients. Users who have access to the billing account through billing hierarchy receive this email.
QuickPayment Notification	This message is populated in the MESSENGER_GROUP_TABLE when a one-time payment has been made from Oracle Self-Service E-Billing. Run the Notifier job to send the actual email message to the recipients. Users who have access to the billing account through billing hierarchy receive this email.
EnrollmentNotification	This message is populated in the MESSENGER_GROUP_TABLE when a new user is enrolled in Oracle Self-Service E-Billing. Run the Notifier job to send the actual email message to the recipients.
NotificationComposer	Option to skip the NotificationComposer task when the Notifier job runs. To skip the task, specify Y, otherwise leave as default (N).
PasswordExpiredNotification	This message goes to system administrator users to warn them that their Command Center password is set to expire. Run the PasswordExpNotifier job before running the Notifier job to send the actual email message to the recipients.

Parameters for Configuring the NotificationDispatcher Task

The NotificationDispatcher task sends email messages that are prepared by the NotificationComposer task to the corresponding end users. The detail implementation logic can vary depending upon the implementation Java class provided by the input parameter value.

The default implementation class is com.edocs.common.notification.queue.QueueDispatcher. In this implementation it looks for entries that are stored in MESSENGER_QUEUE_TABLE and sends email to the user for each entry. If email sending fails, then the message entry stays in the MESSENGER_QUEUE_TABLE until it is sent in the next attempt. Based on the task parameter value, the entry can be removed from the table and from the message repository after the message is sent successfully.

Table 8 describes the configuration parameters for the NotificationDispatcher task.

Table 8. Parameters for Configuring the NotificationDispatcher Task

Parameter	What to Enter or Select
Implementation of Interface IDispatcher	The class performs the dispatch function. The default class is com.edocs.common.notification.queue.QueueDispather. If you need to alter the logic to send email, provide your own implementation class and enter the full class path in this field.
Skip task	When set to the default value of N, the task runs (not skipped).
Cleanup messages after successful delivery	When set to the default value of Y, the task deletes the message from the MESSENGER_QUEUE_TABLE table and from the message repository.

Configuring a BatchReportScheduler Job

The BatchReportScheduler job processes batch reports as requested by the reporting server. Running the BatchReportScheduler job starts processing all scheduled batch reports, making them available online.

Schedule the Report job to run after the ETL load (Reporting feature users only).

There are no configuration parameters for the BatchReportScheduler job.

Configuring a PasswordExpNotify Job

The PasswordExpNotify job gives you the option of sending an email to system administrator users to warn them that their Command Center password is set to expire.

The PasswordExpNotify job sends just one email to a system administrator whose password is set to expire. Running this job every day (or close to it) to provide sufficient time for system administrator users to log in and reset their passwords.

After running the PasswordExpNotify job, you must run the Notifier job with message type PasswordExpiredNotification (or AllNotifications).

The PasswordExpNotify job consists of the PasswordExpNotify Task.

Configuring Parameters for the PasswordExpNotify Task

The only parameter you configure with the PasswordExpNotify Task is: Number of days to send notification before a password is expired.

Configuring a Purge Logs Job

You can create and configure the Purge Logs job to periodically remove historical information from the log table in the database. Purge Logs removes data for all applications you have. Running this job frees space on your database server.

When you configure a Purge Logs job, you specify settings for the Purge Logs production task. You do not need to publish files for a Purge Logs job.

Purge Logs removes data globally (for all applications you have). After you run a Purge Log job, the deleted data is no longer available for inclusion in View Log reports.

Parameters for Configuring the PurgeLogs Task

Table 9 describes the parameters for configuring the PurgeLogs task.

Table 9. Parameters for Configuring the PurgeLogs Task

Parameter	What to Enter or Select
Purge Prior To (Number of Days)	The age, in days, of logs to purge. The PurgeLogs task purges all log files older than this number of days. The default is 90.

Configuring a ReportCleanUp Job

The ReportCleanUp job deletes batch report files and related records from the database.

Parameters for Configuring the ReportCleanUpTask Task

Table 10 describes the configuration parameters for the ReportCleanUp task.

Table 10. Parameters for Configuring the ReportCleanUpTask Task

Parameter	What to Enter or Select
Number of Days to Keep Batch Files	The number of days after which Oracle Self-Service E-Billing deletes report files.
Report DB JNDI	Use `edx.report.databasePool`.

Configuring a ShellJob

You can run create a custom production job that contains a shell script, executable, or other program written to perform a task specific to your requirements. The ShellJob job type enables you to run custom scripts or programs, and include preprocessors or postprocessors as part of a custom job. You can use a SQL script to add the new job to the Oracle Self-Service E-Billing database, making it available to configure, schedule, and run using the Command Center.

The com.edocs.tasks.shellcmd task provides the ShellCmdTask class that executes an external shell command, for example to create custom Command Center jobs.

The following command enables the debug key for the shell command task:

```
-Dcom.edocs.tasks.shellcmd.debug=true Shell commands
```

NOTE: When a shell command job runs, it is possible for it to run successfully but generate nonstandard output. In this case, the return code (0) displays No Operation even though the job ran to completion. To clarify this situation for the user, add a command to the shell script to display a message telling users to check the output.

Parameters for Configuring the ShellCmdTask Task

Table 11 displays the configuration parameters for the ShellCmdTask task.

Table 11. Parameters for Configuring the ShellCmdTask Task

Parameter	What to Enter or Select
Shell Command	Your shell command.
Environment Vars	Your environment variables, separated by semicolons.

Configuring a HierarchyImporter Job

You use the HierarchyImporter job to load data and set up the hierarchy. Run the HierarchyCopy job to get multiple periods of data.

Hierarchy Importer is an optional job for running ETL process. The HierarchyImporter job imports the hierarchies specified in the file as is. System administrators can import the hierarchies for any companies.

Hierarchy XML schema files are in the directory *EDX_HOME*/config/xml, where *EDX_HOME* is the directory where you installed Oracle Self-Service E-Billing.

Parameters for Configuring the Scanner Task

Table 12 describes the configuration parameters for the Scanner task.

Table 12. Parameters for Configuring the Scanner Task

Parameter	What to Enter or Select
Input File Path	The directory where the XML file is that defines the hierarchy to import. Check that the sub directories that are specified in the path exist.
Input File Name	The name of the XML file that defines the hierarchy.
Output File Path	The directory where the processed XML file is stored for the next step. Check that the sub directories that are specified in the path exist.

Parameters for Configuring the HierarchyImporter Task

Table 13 describes the configuration parameters for the HierarchyImporter task.

Table 13. Parameters for Configuring the HierarchyImporter Task

Parameter	What to Enter or Select
XML File	Use the default (Use the output file from Scanner task).
Publish Type	Select whether you want the hierarchies to be published. Suggested value is Published, especially for billing hierarchies. If the file contains a hierarchy that exists already in Oracle Self-Service E-Billing and has been published, importing the hierarchy as Unpublished will cause an error during the Job execution.
Start Period	The start of the period (inclusive) in which the imported hierarchies are to be published.
End Period	The end of the period (inclusive) in which the imported hierarchies are to be published.

Configuring a HierarchyCopy Job

The HierarchyCopy job replicates all published hierarchies for the periods up to the current period.

After Oracle Self-Service E-Billing is in production, schedule the HierarchyCopy job to run when the new billing and reporting period begins. The Period parameter must be the current period. During the initial production data loading, the user has the option to replicate data period by period all the way to the current period.

For example, if you have 12 months of billing data starting in September 2007 that must be populated before your implementation goes live, you can run ETLOLTPProductionLoaderJob for September 2007, then run Hierarchy Copy job for October 2007, (to replicate September 2007's hierarchy to October). After the October data has been replicated, October will not show up in the list. You can then run ETLOLTPProductionLoader again to load billing data for October 2007. The ETLOLTPProductionLoader job will then add additional data to the billing structures that are replicated from September 2007.

Unless the Hierarchy copy job has been run for all periods up to the current period, you must prevent any online activities that involve hierarchy management, to include creating a new hierarchy and publishing to the periods that are later than the period for which you have run the Hierarchy Copy job.

Parameters for Configuring the HierarchyCopy Task

Table 14 describes the configuration parameters for the HierarchyCopy task.

Table 14. Parameters for Configuring the HierarchyCopy Task

Parameter	What to Enter or Select
Replicate All Hierarchies To Period	The period to which you want all hierarchies replicated. The list shows only periods to which hierarchies have not been replicated. After the Copy job has successfully run for the period, the past periods will no longer show up in the list. Note that the current period always shows up in the list even though the Copy job has been run for the current period.

Configuring a HierarchyCleanUp Job

The HierarchyCleanup job removes closed accounts and unsubscribed services from hierarchies. You normally schedule this job to run after all billing cycles have processed for a given billing period.

It usually takes several runs of the ETL process to update all accounts and services in Oracle Self-Service E-Billing for a new billing period. The additional runs are needed to handle multiple billing sources or multiple billing cycles within one billing source. Therefore, accounts and services that no longer exist in the current billing files are never removed by the ETL process.

The job first finds if the HierarchyCleanUp job has been run for the period and any other period prior to the one that has passed. If any period prior to the selected period has not been run with the clean up job, that period will run behind the scenes. For each period being run, the job checks all accounts and services that have no activities for the given period, and sets the last active period for those objects to the given period.

For any accounts and services that have a gap between last active period and the period being run that exceeds the threshold, all references to those objects are removed from hierarchies. For example, if a customer has a business rule stating that accounts and services must be removed if there are no activities for three consecutive billing periods, the Number of Periods is set to 3. When the gap exceeds that number, any references to these objects are removed from the billing hierarchies as well as all business hierarchies. Each account and service has its own counter.

After the HierarchyCleanUp job runs, qualified accounts and services do not appear in any hierarchies or in any reports starting on the period when the cleanup job is run. Historical data for the accounts and services are available in Oracle Self-Service E-Billing for viewing.

The HierarchyCleanUp job consists of only one task, called HierarchyCleanUp.

Configuring a HierarchyPurge Job

When a user deletes a hierarchy, the hierarchy is marked as deleted, but the records of the hierarchy are still in the database. You can schedule the HierarchyPurge job to run periodically (every day, every week, and so on) to remove records marked as deleted from database.

OLTP removes the following data:

- All services and accounts and their subordinated objects, such as service charges and service plans, that are marked as deleted.
- All hierarchies that are marked as deleted.
- All hierarchies that have expired for a specified number of periods. When set a hierarchy to Expire a hierarchy, the data remains in the database until a predefined number of periods pass.

OLAP removes the following data:

- Records of a deleted hierarchy in all related link target workspace tables.
- In all link target workspace tables, records of hierarchy that have expired for a certain number of periods (the threshold value).
- Records for the previously mentioned deleted or expired hierarchies.

Parameters for Configuring the HierarchyPurge Task

Table 15 describes the configuration parameters for the HierarchyPurge task.

Table 15. Parameters for Configuring the HierarchyPurge Task

Parameter	What to Enter or Select
Number of Periods	The number of periods that hierarchy nodes or hierarchies have been expired.

Payment Job Types

You must configure the following payment jobs to enable payment processing. Payment jobs transfer information between a biller and a payment gateway, and perform associated maintenance tasks.

Table 16 lists the available Payment Jobs.

Table 16. List of Payment Jobs

Job Name	Description
pmtRecurringPayment	Schedules payments on a recurring basis, as defined by the user. For details on configuring a pmtRecurringPayment job, see "Configuring a pmtCreditCardSubmit Job" on page 48.
pmtSubmitEnroll	Submits enrollment information to a payment gateway. Currently this field applies to the ACH payment gateway only. For details on configuring a pmtSubmitEnroll job, see "Configuring a pmtCreditCardSubmit Job" on page 48.
pmtConfirmEnroll	Activates pending payment accounts after three days (by default), as long as there has been no error returned. Applies to ACH only. For details on configuring a pmtConfirmEnroll job, see "Configuring a pmtConfirmEnroll Job" on page 36.
pmtPaymentReminder	Sends payment reminder email notifications to customers. For details on configuring a pmtPaymentReminder job, see "Configuring a pmtCreditCardSubmit Job" on page 48.
pmtCheckSubmit	Submits scheduled check and debit payment transaction requests to an Automated Clearing House (ACH) payment gateway. For details on configuring a pmtCheckSubmit job, see "Configuring a pmtCreditCardSubmit Job" on page 48.
pmtCheckUpdate	Updates a check's status according to the response from a payment gateway. For ACH it also processes check returns, prenote returns and NOC returns. For details on configuring a pmtCheckUpdate job, see "Configuring a pmtCheckUpdate Job" on page 41.
pmtARIntegrator	Creates a file in a variety of formats that can be read by Accounts Receivable software. For details on configuring a pmtARIntegrator job, see "For the PaymentIntegratorTask task, you can configure the following parameters described in Table 28." on page 45.
pmtAllCheckTasks	Runs a sequence of Payment jobs automatically, in the required order. For details on configuring a pmtAllCheckTasks, see "Configuring a pmtAllCheckTasks Job" on page 46.
PaymentDueNotification	Sends email to the account holder notifying them of the current account balance due before a configured number of days. For details on configuring a PaymentDueNotification job, see "Configuring a PaymentDueNotification Job" on page 47.
pmtCreditCardSubmit	Submits scheduled credit card payments to a credit card gateway. For details on configuring a pmtCreditCardSubmit job, see "Configuring a pmtARIntegrator Job" on page 45.

Table 16. List of Payment Jobs

Job Name	Description
pmtPaymentRefund	Processes the refund payment after a CSR user or an administrator authorizes a refund. For details on configuring a pmtPaymentRefund job, see "Configuring a pmtPaymentRefund Job" on page 49.
ThresholdExceedNotify	Sends a notification at the time bills are loaded into Oracle Self-Service E-Billing through the ETL process indicating that the user has an amount due that exceeds the configured payment threshold amount. This notification is intended to precede the notification sent by the Recurring Payment job to provide advance notice that the user has a bill amount due that exceeds the threshold amount configured on the Recurring Payment Setup screen. For details on configuring a ThresholdExceedNotify job, see "Configuring a ThresholdExceedNotify Job" on page 54.
pmtCreditCardExpNotify	Sends emails to users whose use a credit card for payments to warn them that their credit card is about to expire. For details on configuring a pmtCreditCardExpNotify job, see "Configuring a pmtPaymentRefund Job" on page 49.
pmtCustom	Use to create a custom job. For details on configuring a pmtCustom job, see "Configuring a pmtCustom Job" on page 55.

Configuring a pmtRecurringPayment Job

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

The pmtRecurringPayment job looks for recurring payments where the amount due is greater than zero, and the date to be scheduled is equal to or greater than a configured number days before the current date (the configurable Number of Days Before Pay Date to Schedule the Payment field). If the number of payments specified in the effective period on the customer interface has been met, this job sets the recurring payment to inactive.

If the customer selects a payment to be made based on the number of days before the due date, or selects the amount to be based on the due amount, the pmtRecurringPayment job must query Oracle Self-Service E-Billing to determine when to schedule the payment or how much to pay, or both. For that reason, you must run the pmtRecurringPayment job after the Oracle Self-Service E-Billing ETL load runs.

If the customer selects a payment for a fixed amount on a fixed date, then the pmtRecurringPayment job does not need to query Oracle Self-Service E-Billing to schedule the payment.

A pmtRecurringPayment job consists of the following tasks:

- RecurPaymentSynchronizerTask

- RecurPaymentSchedulerTask

Configuring Parameters for the RecurPaymentSynchronizerTask

Table 17 describes the configuration parameters for the RecurPaymentSynchronizer Task.

Table 17. Configuration parameters for the RecurPaymentSynchronizer task.

Configuration Parameter	What to Enter or Select
Implementation of interface IRecurringPaymentPlugIn.	The name of the Java class that is called before the pmtRecurringPayment job schedules a payment. It currently does nothing, but you can replace this class with one of your own to process additional business logic to and possibly modify how the payment is scheduled, or cancel it completely. You could use this class to copy selected fields from an index table into a payment table, or to deny a recurring payment. For information about implementing this class, contact Oracle Professional Services. Override the default with com.edocs.common.services.payment.plugin.CustomRecurringPaymentPlugI. The action of this plug-in class is to populate the statement invoice details to the payment_invoices table for the scheduled payments.
When to synchronize recurring payment with Oracle Self-Service E-Billing	By default, the Payment module uses the latest available bill when submitting the payment to the payment gateway. You can configure each payment gateway to only synchronize one time, which reduces processing. The setting whenever job runs can be changed to Only after the current bill is scheduled, which causes the Payment module to synchronize only one time; when the bill is scheduled.
Skip Synchronization	N (No, default) enables synchronization. To ignore synchronization and start scheduling immediately, change the value to Y.

Configuring Parameters for the RecurPaymentSchedulerTask

For the RecurPaymentScheduler task, you configure the following parameters:

Table 18 describes the configuration parameters for the RecurPaymentScheduler task.

Table 18. Parameters for Configuring the RecurPaymentScheduler Task

Parameter	What to Enter or Select
Number of days before pay date to schedule the payment	The number of days before the pay date that the check payment will be scheduled by the pmtRecurringPayment job. The number of days before the due date that email notification will be sent, if Send email notification when the payment is scheduled is set to Y, giving the customer this number of days (less one, the day it is paid) to modify or cancel the scheduled payment. Modifying this field might require modifications to the JSP that checks for valid entries when a customer schedules a check.
Implementation of interface IRecurringPaymentPlugIn	Represents the name of the Java class that is called before the pmtRecurringPayment job schedules a payment. It currently does nothing, but you can replace this class with one of your own to process additional business logic to and possibly modify how the payment is scheduled, or cancel it completely. You could use this class to copy selected fields from an index table into a payment table, or to deny a recurring payment. For information about implementing this class, contact Oracle Professional Services. The default value for this plug-in is: `com.edocs.common.services.payment.plugin.CustomRecurringPaymentPlugIn` The action of this plug-in class is to populate the statement invoice details to the payment_invoices table for the scheduled payments.
Send email when payment is scheduled?	Determines whether email notification is active for recurring payments.
Send email if scheduled payment amount is zero?	Determines whether email notification occurs when the scheduled payment amount is zero.
Send email when recurring payment is expired?	Determines whether email notification is sent when a recurring payment effective period has ended.
Cancel recurring payment if payment account is canceled?	Specify whether the recurring payment must be canceled if the account has been cancelled. This condition is treated as recurring payment expiration and will send an email to the user. Normally, this parameter is set to Y. Use N to have the pmtCheckSubmit job handle this condition, or if the plug-in is going to take actions based on this condition.

Table 18. Parameters for Configuring the RecurPaymentScheduler Task

Parameter	What to Enter or Select
Cancel recurring payment if payment account is invalid?	The account information (contained in a prenote for ACH) sent to the ODFI by the pmtConfirmEnroll job was returned to the Payment module as having incorrect account information. The user is enrolled, but the account is not valid. If a credit card account was used for enrollment, the account information is not checked until a payment is made. If a credit card payment is sent to the payment processor with invalid account information, the account will be marked invalid. Because the customer's enrollment failed, they will be sent an email. The customer must resubmit the information for that account, which must be verified before this account can be used to make a payment.
Send email when recurring payment is canceled?	Specify whether to send email to notify the user that his or her recurring payment was cancelled.

Configuring a pmtSubmitEnroll Job

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

The pmtSubmitEnroll job submits enrollment information to a payment gateway (for ACH only). The job finds all accounts in the payment_accounts table whose account_status field is pnd_active and writes them into an ACH file. The txn_date field is set to the current date, and account_status field is changed to pnd_wait. For information about using PayPal Payflow Pro threads, see "Using PayPal Payflow Pro Threads" on page 49.

The pmtSubmitEnroll job has only one task, called pmtSubmitEnroll.

Configuring Parameters for the SubmitEnrollTask

Table 19 describes the configuration parameters for the SubmitEnroll Task.

Table 19. Parameters for Configuring the SubmitEnroll Task

Parameter	What to Enter or Select
Skip Holidays	Determines whether to send the ACH payment batch file to the ACH payment gateway even when the bank is closed because of a holiday. The default is N, which means send the file even if it is a holiday.

Configuring a pmtConfirmEnroll Job

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

The number of days to wait is specified by the Days to Activate Pending Subscribers parameter in the Payment Settings for the ACH payment gateway. There are no configuration parameters for the pmtConfirmEnroll job.

Configuring a pmtPaymentReminder Job

The pmtPaymentReminder job populates the email notifications that will be sent to customers. This job sends payment reminder email notifications to the following types of customers:

- Who have configured payment reminders.
- When the status of a check changes to processed, failed, canceled, or returned.
- When the status of a credit card changes to paid, failed, canceled, or returned.

The operations that you can perform for the pmtPaymentReminder Job are as follows:

- **Payment Reminders**. Email notifications are sent out for customers who have set up payment reminders. pmtPaymentReminder sends out reminder email for reminders in the payment_reminders table whose next_reminder_date is today or later, and whose Reminded field is N. After sending the email, the Reminded field is updated to Y, and the next_reminder_date field is updated as specified by the reminder_interval field.
- **Check Payment Notification**. pmtPaymentReminder sends out email for checks as configured in the job. Email notifications can be sent for rows in the check_payments table where the Status field is Returned, Failed, or Processed (sent for processing). After sending the email, the Reminded field is updated to Y.

  The check status values are as follows.

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

- **Credit Card Payments**. pmtPaymentReminder sends email for the scheduled credit cards in the creditcard_payments table whose status field is Settled or Failed to authorize, and whose reminded field is N. After sending email, pmtPaymentReminder sets the reminded field to Y. pmtCreditCardSubmit sets the reminded field back to N when it makes a payment for that scheduled credit card.
- **Email Templates**. The email address is retrieved from the payer_email_addr field in the payment_reminders table. Email content is created using the email template file configured for this job. The default template file is paymentReminder.txt, which is in the *EDX_HOME*/payment/lib/payment_resources/ directory.

  The template used for the email sent with a configured payment reminder is payment_reminder_fixed.xml.vm.

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

  (The pmtPaymentReminder job also uses the notifyEnroll.txt and payment_account_status.xml.vm templates.)

NOTE: The Notifier job is responsible for sending the email notification to the customer from the MESSENGER_GROUP Table.

The pmtPaymentReminder job consists of the PaymentReminderTask.

Parameters for Configuring the PaymentReminderTask

Table 20 describes the parameters you configure for the PaymentReminder task.

Table 20. Parameters for Configuring PaymentReminder Task

Parameter	What to Enter or Select
Implementation of Interface IPaymentReminderPlugin	Optional class that modifies the IPaymentReminderPlugin plug-in. You can modify whether a payment reminder is sent and other actions based on the type of reminder. For information about implementing this class, contact Oracle Professional Services. The default is com.edocs.payment.tasks.reminder.PaymentReminderPlugin.
Notify if a check is sent for processing?	Indicates whether notification is to be sent for checks that were sent for processing.
Notify if a check is cleared?	Indicates whether notification is to be sent for checks that have been paid (cleared).
Notify if an ACH transaction fails, is returned or is canceled?	Indicates whether notification is to be sent for checks that were returned by the payment gateway as canceled, returned or failed settlement.

Table 20. Parameters for Configuring PaymentReminder Task

Parameter	What to Enter or Select
Notify if a credit card is settled	Indicates whether email must be sent for credit card payments that are settled successfully.
Notify if a credit card transaction is not authorized or is canceled	Indicates whether email must be sent for credit card payments that failed to authorize or were canceled by the Payment module.

Configuring a pmtCheckSubmit Job

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

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

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

A zero dollar amount check (a prenote) will not be submitted, and this job changes the check's status to Processed.

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

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

The format of the ACH file can be modified by editing the XML files in the *EDX_HOME*/payment/lib/payment_resources/ach/template/ directory (or the *EDX_HOME*\payment\lib\payment_resources\ach\template\ directory in Windows). In the path, *EDX_HOME* is the directory where you installed Oracle Self-Service E-Billing. See Oracle Professional Services for help modifying ACH batch format.

After a check is submitted, its status in the database changes from Scheduled to Processed. If an error occurs during the check submission process, the status of the check changes to Failed.

Table 21 shows the fields that the pmtCheckSubmit job updates in the check_payments table.

Table 21. Columns Updated in the Check Payments Table by the pmtCheckSubmit Job

Column	Description
last_modify_time	Current time
status	7
action_code	For ACH: 27 for checking and 37 for saving.
txn_number	For ACH: trace number.
reminded	N
log_id	ID of the summary report in the payment_log table.
gateway_payment_id	Populated only if you are using the gateway payment ID to match a check returned from ACH to a check in the Payment database. For more information, contact Oracle Professional Services.

Scheduling and Holidays

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

If the holiday is on:

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

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

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

- If the pay date is today or earlier, then the batch entry effective date is set to tomorrow. If not, it is set to the pay date.
- If the batch entry effective date is on a bank holiday, then it is moved to the next availed bank business date.

If you do not want to skip holidays when calculating the batch entry effective date, modify the Payment Settings.

The CheckSubmit job consists of the CheckSubmit task.

Configuring Parameters for the CheckSubmitTask

Table 22 describes the configuration parameters for the CheckSubmitTask.

Table 22. Parameters for Configuring the CheckSubmit Task

Parameter	What to Enter or Select
Number of days before a check's pay date for it to be submitted	When a check payment is scheduled, a date must be specified when the check is going to be cleared. By default, a check payment will not be submitted to a payment gateway until one day before the scheduled pay date. The submission date can be changed by specifying a different value. The value of this parameter can be zero. Modifying this field might require modifications to the JSP that checks for valid entries when a customer schedules a check. For example, if the value is one and the job runs today, all the checks whose pay dates are tomorrow or earlier will be selected to send to the ACH payment gateway. Payments made after the pmtCheckSubmit job runs will not necessarily be paid on the same day. It is recommended that you run this job at 11:59 P.M. to ensure that payments will be processed on the same day as they were made. If the job runs early in the morning each day, for example, at 2:00 A.M., then the job will not process payments scheduled during normal business hours on the same day, since it already ran that day.
Cancel payment if check account is canceled?	Specifies whether the scheduled payment must be canceled if the check account has been cancelled. Normally, the value is Y. Use N if the plug-in is going to take actions based on this condition.
Cancel payment if account information is invalid?	The account information (contained in a prenote for ACH) sent to the ODFI by the pmtConfirmEnroll job was returned as having incorrect account information. The user is enrolled, but the account is not valid. Because the customer's enrollment failed, they will be sent an email. The customer must resubmit the information for that account, which must be verified before this account can be used to make a payment.
Submit payment if check account is pending?	When the customer adds a new checking account, it is in a pending state until the period specified by Days to Activate Pending Subscribers has expired. The value Y means submit the payment even if the account is pending. If the value is N, the task does not submit the payment when the account is pending.
Submit checks for additional DDNs	List additional DDNs checks that the pmtCheckSubmit will submit to the payment gateway for processing, separated by semicolons.
Skip Holidays	Determines whether to send the ACH payment batch file to the ACH payment gateway even when the bank is closed because of a holiday. The default is N, which means send the file even if it is a holiday. The federal holidays are listed on page 65.

Configuring a pmtCheckUpdate Job

The pmtCheckUpdate job updates a check's status according to the response from the payment gateway. All files that match the necessary criteria are processed and moved to a history directory under the input directory. This job examine the check_payments table changes the status to Paid of any checks that meet the following criteria:

- Status is Processed
- Pay date is five or more days old
- Not returned

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

There are no configuration parameters for the pmtCheckUpdate job.

Automated Clearing House (ACH) Logic

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

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

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

For each successful batch header record match, the pmtCheckUpdate job updates the status according to Table 23.

Table 23. Status Changes With the pmtCheckUpdate Job

Status Change	Addenda Type	Amount Field in the First Detail Record	Return
Prenote returned (prenote_returned)	99	0	Prenote error
NOC returned (noc_returned)	98	0	NOC
Processed (processed)	99	A value other than zero (0)	Check error

The pmtCheckUpdate job also updates the status to Paid if there is no return from ACH after the configured number of days. In the configuration settings for an ACH payment gateway, you can change the number of days to wait after the pay date.

TIP: If pmtCheckUpdate will be processing ACH return files containing multiple DDNs, then the Payment Settings for each payment gateway must have the same immediate origin and immediate destination. Also, each payment gateway used by those DDNs must use the same ACH template files (ACH Template Directory parameter).

ACH Return Types

One return file can have three kinds of returns:

- Check returns
- NOC returns
- Prenote returns

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

Table 24 lists the columns that are updated in the check_payments table after a check return is processed.

Table 24. Columns Updated in the ACH Check Returns Table

Column	Value
last_modify_time	Time that the table was modified.
status	-4
reminded	N
log_id	ID of the exception report in the payment_log table
txn_err_msg	ACH return code

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

Table 25 lists the columns that are updated in the payment_accounts table after a prenote is processed.

Table 25. Columns Updated in the ACH Prenote Returns Parameters Table

Column	Value
txn_message	return ACH code
account_status	bad_active
notify_status	N

NOC Returns

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

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

NOC_CODE::*NEW_ADDENDA_INFO*::*OLD_ADDENDA_INFO*

where:

- *NOC_CODE* is the returned NOC code.
- *NEW_ADDENDA_INFO* is the correct NOC information returned from ACH (content from position 36 to 64 of addenda record, with white spaces trimmed).
- *OLD_ADDENDA_INFO* is the existing or incorrect addenda information with the same format as new_addenda_info and is calculated on current payment account information.

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

Table 26 lists the columns that are updated in the payment_accounts table after a NOC is processed.

Table 26. Columns Updated in the ACH NOC Returns Parameters Table

Column	Value
txn_message	*NOC_CODE*::*NEW_ADDENDA_INFO*::*OLD_ADDENDA_INFO*
account_number	Changed for C01, C03, C06, C07
routing_transit	Changed for C02, C03, C07
account_type	Changed for C05, C06, C07

Table 26. Columns Updated in the ACH NOC Returns Parameters Table

Column	Value
txn_date	current time
notify_status	N

ACH Return File Format

The ACH return file must not include new lines at the end of each record. To ensure that the ACH return files process correctly, the information listed in Table 27 must return correctly.

Table 27. ACH Return File Format

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

Table 27. ACH Return File Format

Column Name	Record	Description
totalDebitAmount	fileTrailer	Total debit amount in the file.
totalCreditAmount	fileTrailer	Total credit amount in the file.

Configuring a pmtARIntegrator Job

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

Parameters for Configuring PaymentIntegratorTask Task

For the PaymentIntegratorTask task, you can configure the following parameters described in Table 28.

Table 28. Parameters for Configuring the PaymentIntegratorTask

Parameter	What to Enter or Select
Full Path Name of XML Query File	Enter the path to the general query template file. The default path is *EDX_HOME*/payment/lib/payment_resources/ar/arQuery.xml, which you can modify to fit your needs. In the path, *EDX_HOME* is the directory where you installed Oracle Self-Service E-Billing. Refer to the sample arQuery.xml file, which shows a check query and a credit card query.
Check Query Name in XML Query File	Enter the value of the name attribute of the query element in arQuery.xml, which is used for the query against the check_payments table. This query only works for the check_payments and check_payments_history tables. You can modify the values, or add new query elements.
Credit Card Query Name in XML Query File	Enter the value of the name attribute of the query element in the arQuery.xml file. This query currently only works for the creditcard_payments and creditcard_payments_history tables. You can modify the values, or add new query elements.
Implementation of IARPaymentIntegrator	Specifies the implementation class for IPaymentIntegrator. The default parameter is com.edocs.payment.tasks.ar.SampleARPaymentIntegrator. The IPaymentIntegrator interface defines a method to use the Payment module to generate accounts receivable files in a specific format.
Full Path of AR Template File	The complete path to the payment source template file. The default path is *EDX_HOME*/payment/lib/payment_resources/ar/arFlat_template.txt. The default file is a sample flat text template file that shows how to format output, which you can edit to meet your requirements. The other template file provided with the ARIntegrator job is arXML_template.xml.

Table 28. Parameters for Configuring the PaymentIntegratorTask

Parameter	What to Enter or Select
Directory for Output AR File	Specify a directory to put the output file (*EDX_HOME*/payment/Output/AR or other location).
Transform Output AR File to Another Format?	Flag, Y or N. This parameter is ignored unless an XML template is chosen. If the value is Y, the pmtARIntegrator job takes the generated XML file in the output directory as XML input for the XSLT processor, reads the XSLT template, and transforms the data into a different file format. Do not set this field to Y if the XSLT file used to transform the AR file to a different format is in TXT format. Only enter Y if the XSLT template file format is well formed XML.
Full Pathname of XSLT File Used for Transform.	The XSLT template file. You can create your own XSLT template file in a different directory with a different name. The default template, arTransform.xsl, is a sample flat text file that shows how to format output, which you can edit to meet your requirements. You can find this file in *EDX_HOME*//payment/lib/payment_resources/ar. After specifying the preceding parameters and scheduling the job, the pmtARIntegrator job generates an output file in output directory based on your check and credit card query criteria as specified in the arQuery.xml file and the Java class ARPaymentIntegrator.java. You can modify the sample templates files and reimplement the IPaymentIntegrator interface to add features. The XSLT template file must be well-formed XML or the pmtARIntegrator job fails with the error: java.lang.exception: javax.xml.transform.TransformerException.
Directory for Transformed File.	The directory for the final transformed file. The default is *EDX_HOME*//payment/Output/ar, where *EDX_HOME* is the directory where you installed Oracle Self-Service E-Billing.
File Name Extension for Transformed File	The file extension of the final transformed file. The default is TXT.
Flexible Parameter 1 (for customization)	Optional parameter used in the AR query.
Flexible Parameter 2 (for customization)	Optional parameter used in the AR query.

Configuring a pmtAllCheckTasks Job

The pmtAllCheckTasks job runs the following Payment check payment jobs sequentially:

- pmtRecurringPayment
- pmtSubmitEnroll

- pmtConfirmEnroll
- pmtCheckUpdate
- pmtPaymentReminder
- pmtARIntegrator
- pmtCheckSubmit

You configure all of the task parameters associated with each individual Payment job for the pmtAllCheckTasks job. For descriptions of the configuration parameters for this job, see the following topics:

- "Configuring Parameters for the RecurPaymentSynchronizerTask" on page 33
- "Configuring Parameters for the RecurPaymentSchedulerTask" on page 33
- "Configuring Parameters for the SubmitEnrollTask" on page 35
- "Parameters for Configuring the PaymentReminderTask" on page 37
- "Parameters for Configuring PaymentIntegratorTask Task" on page 45
- "Configuring Parameters for the CheckSubmitTask" on page 40

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

Configuring a PaymentDueNotification Job

ThePaymentDueNotification job sends email to billing account holders, notifying them of their current account balance due. End users configure their own payment due notification through the Oracle Self-Service E-Billing application notification settings, which tell Oracle Self-Service E-Billing how many days prior to the payment due date that they would like to receive this notification. This job identifies all the users who are scheduled to receive payment due notification at the time the job runs. It then sends email to the corresponding users. For each message sent, this job keeps a record in Oracle Self-Service E-Billing for a specific number of configurable days.

Parameters for Configuring the PaymentDueNotification Task

Table 29 displays the configuration parameters for the PaymentDueNotification task.

Table 29. Parameters for Configuring the PaymentDueNotification Task

Parameter	What to Enter or Select
Number of days to keep records	Enter the number of days you want Oracle Self-Service E-Billing to save payment notification records.

Configuring a pmtCreditCardSubmit Job

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

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

Credit card account information is saved by the Payment module as part of the payment when the payment is scheduled. Whether this copy of the account information is used for submission depends on the contents of the PID field:

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

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

The Payment module supports the PayPal Payflow Pro and uses HTTP to communicate with it. If there is a network problem, the status of the payment stays scheduled, but the payment txn_err_msg field gets the error message, ensuring that the payment will be picked up by the next run of the pmtCreditCardSubmit job. If the payment is successful, the Payment module stores the confirmation number from PayPal Payflow Pro in the txn_number field of the creditcard_payments table.

Table 30 describes the columns in the CHECK_PAYMENTS table updated after a credit card is submitted.

Table 30. Columns Updated in the CHECK_PAYMENTS Table After a Credit Card is Submitted

Column	Description
last_modify_time	Current submit time.
reminded	Set to N if the status is Settled or Failed-authorize.
status	Set to Settled if the payment is accepted by card issuer. Set to Failed-authorize if the payment is rejected or returned as referral by card issuer. Stays Scheduled if there is a network error.
log_id	Contains the value in payment_log, which represents a report ID.

Parameters for Configuring the CreditCardSubmitTask

The configuration parameters for the CreditCardSubmitTask task are:

Table 31 describes the configuration parameters for the CreditCardSubmitTask.

Table 31. Parameters for Configuring the CreditCardSubmit Task

Parameter	What to Enter or Select
Number of Days Before a Credit Card's Pay Date for it to be Submitted	When a credit card payment is scheduled, a date must be specified when the credit card payment must be settled. By default, a credit card payment will not be submitted to a payment gateway until one day before the scheduled pay date. The submission date can be changed by specifying a different value. Modifying this field might require modifications to the JSP that checks for valid entries when a customer schedules a payment.
Cancel Payment if Credit Card Account has Expired?	If the credit card account used in a payment has expired, then cancel the payment (Y or N). If you specify N, the task tries to make the payment using the old account. (If the task fails to make the payment using the old account, the pmtPaymentReminder job sends email to the customer, if configured.)

Using PayPal Payflow Pro Threads

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

However, there is a bug with the PayPal Payflow Pro SDK, which causes a connection failure when the number of threads is too high. Connection failures can be significantly reduced by using multiple copies of the PayPal Payflow Pro certificates. By default, there is only one certificate.

Connection failures caused by the PayPal Payflow Pro bug are not fatal. The Payment module recognizes the failure and keeps the payment's status as scheduled so that the failed payments process the next time the pmtCreditCardSubmit job runs. If you increase the number of threads and find there are failures, schedule your job run twice, back to back.

Configuring a pmtPaymentRefund Job

An administrator can authorize refunds and the pmtPaymentRefund job processes the refund payment.

Oracle Self-Service E-Billing supports refunds for credit card transactions that have Settled or Paid status. There are three categories of refunds:

- **Cancellations.** A payment can be cancelled only when a payment has a status of Scheduled. If a payment is in another state, the payment cannot be cancelled.

- **Voids.** If a payment has a status of Authorized, it can be voided. An authorized payment must not be cancelled without performing an authorization reversal for the authorized amounts. Voids apply only to credit cards and not checks. The authorization reversal procedure is determined through the cassette implementation.
- **Refunds.** A payment can be refunded if it has Settled or Paid status. It is necessary to create a new payment transaction to make the refund.

The pmtPaymentRefund job consists of the following tasks:

- LoadRefunds
- SubmitRefund

Configuring the Batch Refunds XML

The pmtPaymentRefund job uses XML to retrieve the refund records. You must also configure the Batch Refunds XML for payment refunds as shown in the following example text.

Example XML for Batch Refunds

Process the batch refunds XML using the following format:

```
<?xml version="1.0" encoding="UTF-8"?>

<!--  The root element can be either doc or refunds both elements are supported. doc
is supported for the ddf conversion.   -->

<doc><!-- refunds -->

<refund>

<payeeId>BCBS</payeeId>

<userId>John</userId>

<paymentAccountNumber>4317345689007461</paymentAccountNumber>

<payerAccoutNumber>ACC1234567890</payerAccoutNumber>

<amount>200.00</amount>

<paymentType>ccard</paymentType>

<refundType>R</refundType><!-- R for Refunds D for Deposits -->

<billingSystemTransactionNumber></billingSystemTransactionNumber>

<originalTransactionNumber>12132312</originalTransactionNumber>

<sourceSystemPaymentInitiator>CSR2</sourceSystemPaymentInitiator>

<transactionDescription>Refund Desc</transactionDescription>

<notifyRequired>Y</notifyRequired>

<flexFieldOne>xxxxxx</flexFieldOne>
```

```
<flexFieldTwo>xxxxxx</flexFieldTwo>
<flexFieldThree>xxxxxx</flexFieldThree>
<flexFieldDate>mm-dd-yyyy</flexFieldDate>
</refund>
</doc><!-- refunds-->
```

Table 32 describes the configuration parameters you must specify in the XML file for batch refunds.

Table 32. Parameters for Configuring the Batch Refund XML File

Parameter	What to Enter or Select
Payee Id	DDN Name of the Biller. (Required)
User Id	The user's registration ID used to identify the user. (Required)
Payment Account Number	The user's credit card or checking account number where the refund must be made. (Required)
Payer Account Number	The account number that the user is paying for. (Mandatory field for refunds.)
Amount	The amount to be refunded. (Optional) If the Amount tag does not contain an amount, the job refunds the original amount of the payment.)
Payment Type	The refund payment mode for credit card. (Required)
Refund Type	Refund type: R–Refund (Required)
Billing System Transaction Number	The transaction number provided by Oracle Self-Service E-Billing (the payment ID). (Required)
Original Transaction Number	The transaction number provided to the Payment module by the authorizing gateway or clearing house. Optional field, but populating this field helps the gateway to process the refund more efficiently. (Optional)
Source System Payment Initiator	The ID of the refund payment initiator and is used for security. (Optional)
Transaction Description	The description for the refund justification. (Optional)
Notify Required	Whether a notification is required for the refund. Notification can be sent to the biller as well as the payer. (Optional)
Flex Field One	Flexible field for storing custom data. (Optional)
Flex Field Two	Flexible field for storing custom data. (Optional)
Flex Field Three	Flexible field for storing custom data. (Optional)
Flex Field Date	Flexible field for storing custom data. (Optional)

About the LoadsRefundsTask

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

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

If the refund type is R (Refund) then the task checks whether the required fields are set. To identify the required fields, see Table 32 on page 51.

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

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

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

Parameters for Configuring the LoadsRefunds Task

Table 33 describes the parameters you must configure for the LoadsRefunds task.

Table 33. Parameters for Configuring the LoadsRefunds Task

Parameter	What to Enter or Select
Payment selection criteria for refunds	Select the unit of time (days, weeks, or months) to use for payment refund criteria.
Selection criteria value (Greater than 0 value)	Enter a specific number (of days, weeks, or months specified in previous field) to use for payment refund criteria.
Refund batch file location directory	The input directory of the XML file.
Refund batch file name	The XML file name.
Refund batch file archive location directory	The output directory of the XML file.
Implementation of interface IRefundDepot	The implementation class of the IRefundDepot.
Implementation of interface ILoadRefundPlugin	The implementation class of the ILoadRefundPlugIn.
Skip Refund Loader Task	Default is N (No). If there are no batch refunds for the Biller, set this parameter to Y (Yes) to skip the task.

About the SubmitRefundTask

The SubmitRefundTask submits all the scheduled refunds for payment. This task accesses the payment_refunds table and queries all the scheduled refunds and processes them one at a time. the SubmitRefundTask processes refund types as follows:

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

  If a scheduled payment is cancelled, Oracle Self-Service E-Billing processes it. If the status of the payment is Scheduled, Oracle Self-Service E-Billing cancels it immediately and the refund status changes to Processed. A refund type of Cancellation is never scheduled, so the status changes to Processed because there is no gateway interaction. If the payment is in another state, it is processed as a void or a refund.

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

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

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

  If the original payment is not in the Authorized state and the status is Processed or Settled, Oracle Self-Service E-Billing processes it as a refund.

  You must only use a Void on a transaction that has not yet settled. To refund a customer's money for a settled transaction, you must submit a Refund (credit) transaction.

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

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

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

  If the plug-in accepts the new payment, then it schedules the new payment transaction in the payment tables and updates the status of the refund to Processed and records the new payment ID in the refund using the refund payment id field. If the plug-in rejects the payment, Oracle Self-Service E-Billing changes the refund status to Rejected.

Parameters for Configuring the SubmitRefund Task

Table 34 describes the parameters you must configure for the SubmitRefundTask.

Table 34. Parameters for Configuring the SubmitRefund Task

Parameter	What to Enter or Select
Implementation of Interface ISubmitRefundPlugin	The implementation class of the ISubmitRefundPlugin
Allow multiple refunds per payment	Y (Yes), or N (No). The option to allow multiple refunds for a payment.

Configuring a ThresholdExceedNotify Job

The ThresholdExceedNotify job sends a notification at the time bills are loaded into Oracle Self-Service E-Billing through the ETL process indicating that the user has an amount due that exceeds the configured payment threshold amount. This notification is intended to precede the notification sent by the Recurring Payment job to provide advance notice that the user has a bill amount due that exceeds the threshold amount configured on the Recurring Payment Setup screen.

The ThresholdExceedNotify job consists of the ThresholdExceedNotifyTask, which has no configuration parameters.

Configuring a pmtCreditCardExpNotify Job

When credit card accounts for payments are configured, the pmtCreditCardExpNotify job sends an email to users whose credit card will expire soon.

Parameters for Configuring the CreditCardExpNotifyTask Task

You can configure the following parameters for the CreditCardExpNotify

Table 35 describes the configuration parameters for the CreditCardExpNotifyTask task.

Table 35. Parameters for Configuring the CreditCardExpNotifyTask Task

Parameter	What to Enter or Select
Number of days before card expiration date to send email	Specifies the number of days before their credit card expires to send the expiration notice. The default is 30.
Implementation of Interface ICCExpNotificationPlugin	Represents the name of the Java class that is called before the pmtCreditCardExpNotify job emails notifications. It currently does nothing, but you can replace this class with one of your own to process additional business logic to and possibly modify credit card expiration email, or cancel it completely.

For information about implementing this class, contact Oracle Professional Services. The default value of the pmtCreditCardExpNotify job is:

```
com.edocs.payment.tasks.ccexpnotify.CCExpNotificationPlugIn
```

Configuring a pmtCustom Job

You can use the pmtCustom job type to create a custom Payment-related job to perform tasks that are not part of standard Payment jobs. By default, there is one task with five parameters, which you define.

5 Administering the Live Production Process

This chapter describes the management of the production process and the scheduling of jobs in Oracle Self-Service E-Billing. It includes the following topics:

- Administering Jobs on page 57
- Starting a Job Manually on page 63
- Monitoring Production Jobs on page 58
- Viewing Job Status on page 59
- Viewing and Verifying Task Status on page 60
- Canceling and Rerunning Failed Jobs on page 62

Administering Jobs

After you have set up and configured an application and its jobs, use the Command Center to schedule jobs, manage the production process on a daily basis, and to perform administrative activities related to your application.

The Command Center Main Console provides a high-level status of all activity related to jobs in the production environment, and is the first screen that appears when you log in to the Command Center.

Use the Main Console perform the following monitoring activities:

- Setting and changing job schedules. For details about scheduling jobs, see “Scheduling Jobs” on page 65.
- Continuous monitoring of job and task status. For details about monitoring jobs, see “Monitoring Production Jobs” on page 58.
- Starting a job (Run Now feature); for details see “Starting a Job Manually” on page 63.
- Monitoring services. For details on monitoring services, see “Monitoring Service Status” on page 119.

Perform the following regular maintenance activities to keep your applications running efficiently in an ongoing, live production environment:

- Daily application monitoring tasks:
  - Check the status of production jobs and tasks on a continuous basis using the Command Center.
  - Check the administrator email accounts for any administrator alert mail. Administrator email is generated if there's a problem passing email notifications to the SMTP host or if email notification is not working properly for some other reason. For details about sending administrator email, see “Creating Alert Groups” on page 70.

- Weekly (or more frequently): Check production message logs (Error, Information, Warning, and Debug logs). For details on viewing log reports, see "About Message Log Files" on page 118.
- General: Run job reports to view job activity; for details, see "About Job Reports" on page 117.

Monitoring Production Jobs

The Main Console of the Command Center shows the state of all production jobs for your applications.

Regularly check the status of jobs and tasks to track:

- Whether a job has completed successfully. For details on job status, see "Viewing Job Status" on page 59.
- Which tasks completed successfully. For details on task status, see "Viewing and Verifying Task Status" on page 60
- Why a job failed, and to restart or cancel failed jobs. For details on managing failed jobs, see "Canceling and Rerunning Failed Jobs" on page 62.

Table 36 describes each field on the Command Center Main Console.

Table 36. Fields on the Main Console

Field	Description
Application	Name of the application.
Job Name	Name of the job.
Job Type	The purpose of the batch job. For example, Email Notification, Purge Logs.
Last Run	Date and time the representative job instance ran.
Run Time	Elapsed time the representative job instance has been running in hours, minutes, and seconds.
Status	Current execution state of the representative job instance.
Next Run	Date and time the job is scheduled to run next. (This parameter applies only to the job and not a particular instance.)
Action	Displays an option that lets you take action on that job. The Run Now option lets you run the job one time immediately, overriding the scheduling parameters (except concurrency parameters). The Retry option lets you retry all failed instances of the job.

NOTE: The Main Console does not show any activity until you create one or more applications and jobs.

For each application, the Main Console lists each configured job type alphabetically. Although there can be multiple instances of an individual job for an application, the Main Console can display only one, so it chooses a representative job instance. The job instances are sorted first by status ranking and then by last run time in reverse chronological order. The top-most instance from that list is selected as the representative instance.

Listing Jobs for an Application

You can list jobs by application on the Command Center Main Console.

To list jobs for a particular application only

- On the Main Console, click the name of the application in the Application column. The Edit Application page appears, showing only those jobs defined for the selected application.

Sorting Jobs on the Main Console

You can sort jobs displayed on the Command Center Main Console.

To sort jobs listed on the Main Console by application

- Click Application in the column header.

To sort jobs on the Main Console by job name (alphabetically), job type, last run, run time, status, or next run

- Click the column header.

Displaying Current Job Status

Use the following procedure to display current information in the Command Center Main Console.

To display current job status on the Main Console

- Click Refresh.

Viewing Job Status

The status of each production job appears on the Command Center Main Console. Jobs can have one of the following status values, shown in Table 37 in the order used for ranking purposes.

Table 37. Job Status Values

Job Status	Description
Failed	Job failed
Processing	Job is currently executing

Table 37. Job Status Values

Job Status	Description
Reprocessing	Job is currently executing after a user manually selected it for reprocessing using Retry or Retry All.
Reprocess	A user has manually selected the job for reprocessing using Retry or Retry All, but the job has not yet begun.
No operation	Job or task did nothing as resources were not ready yet, for example, if the Scanner task found no file in the input directory.
Done	Job has completed successfully.
Cancelled	Job run failed and was cancelled.
Not yet started	Job has not begun executing.
Done, recurring	Job completed successfully and has been scheduled to run again, or the job has processed one data file and is looking at the input directory to determine whether there are any more data files to process in this run.
No operation, recurring	Previous job run resulted in a No operation status, but the job has been scheduled to run again.
Canceled, recurring	Job was canceled and is now looking at the input directory to determine whether there are any more data files to process in this run.

Viewing and Verifying Task Status

Each production job consists of several individual tasks that work together to generate job output. In addition to job status, each individual task is assigned a status when the job runs. You can closely monitor and manage the task status for a job instance using the Command Center Task Status page.

Every configured task must complete successfully before the application sets job status to Done on the Main Console.

If any of the production tasks is unable to complete, the job fails, and the status changes to Failed. All failed jobs display in red on the Main Console. If a job fails, you can run it again.

(In addition to checking the individual task status on the Task Status screen, you can check for individual task output to determine whether a task completed successfully.)

To view task status detail for a job

1. Click the status of the job in the Main Console Status column.

   The Task Status screen appears showing the status of each production task in the job.

2. To change the display order of tasks (processing order remains unchanged), click Task. To change the display order of information in the Last Run and Status columns, click Last Run or Status. Click the links again to restore each display to its original order.

3. Click Refresh to display an updated task status.

4 You have the option of rerunning or canceling a failed job. Click Retry Failed Job, or Cancel Failed Job.

5 The Task Status page displays each instance of a job started during the most recent scheduled run in reverse chronological order (youngest first), along with the status of each task in the instance.

The Task Status page identifies each job instance by a Job Instance ID, and displays the information described in Table 38.

Table 38. Command Center Task Status Fields

Field	Description
Job Instance ID	A number uniquely identifying each job instance.
Last Updated	The time the task status last updated.
Status	Current execution state of the task. Task Status can be: Processing, Failed, Reprocessing, Reprocess, No operation, Canceled, Not yet started, or Done.
Action	Displays an option that lets you take action on that job instance or on all instances. Retry lets you retry that instance; the Retry All option lets you retry all failed instances of the job. Cancel lets you cancel that instance; the Cancel All option lets you cancel all failed instances of the job.

Which Job Instances Appear on the Task Status Page

The Task Status page displays:

- Up to the last N job instances that have Done, Canceled, or No operation status (where N is the maximum number of instances allowed for the job), plus
- Any instances in Processing, Failed, Reprocessing, or Reprocess status

If you are not using concurrency (N=1), the Task Status page shows up to five rows of job instances in Done, Canceled, or No operation status, plus any instances in Processing, Failed, Reprocessing, or Reprocess status.

When a scheduled run completes, the completed rows remain in view on the Task Status page until a new schedule begins. At this point, the Task Status page begins displaying the instances generated by the new schedule instead. The only exception is that any instances from the previous schedule still in Processing, Failed, Reprocessing, or Reprocess states remain even if a new schedule has begun. Those instances are removed from the Task Status page once processing is complete, or in the case of a failed instance, once you cancel or retry it successfully.

Schedules can overlap if a second schedule begins before the current run completes. Another scheduled run can begin only if:

- The first run is not using the maximum number of instances (if enough resource is available). For example, if the first run has 3 instances in Processing and the maximum allowed is 10, the next run can start up to 7 new instances.

- No job instances in the first are in the Failed state.

  Overlapping schedules mean that instances from both schedules could appear on the Task Status page. You can tell from the Last Updated field to which schedule the instance belongs.

  The number of rows that appear on the Task Status page at any given time depends on the point of progress of the job and:

  - Whether you have enabled concurrency for the job (if the maximum number of instances specified in the schedule is greater than 1).
  - Represents the maximum number of Done, Canceled, or No operation jobs that can appear on the Task Status. The Task Status shows a maximum of 5 job instances in Done, Canceled, or No operation.
  - For jobs that scan for an input file, the number of input files placed in the input directory.
- For jobs that process multiple statements in parallel with the StatementScanner task, such as the Report job, the number of statements to process up to the maximum number of instances.
- Whether the job schedule overlaps due to a long lasting run.

Canceling and Rerunning Failed Jobs

You can use the Main Console to cancel or rerun a job, and the Task Status page to retry or cancel a failed instance of a job.

If one instance fails, other instances that have started continue to completion, but no new instances are started.

Retry running a failed job or job instance if you want to start it from the point where it failed. If you want to restart a job or instances of a job, cancel and run it again.

If the task has not been started, the Last Update field shows - and Status shows Not Yet Started.

Retrying a Failed Job Before Next Schedule Run

Use the following procedure to retry a failed job before its next scheduled run.

To retry a failed job before its next scheduled run time

- On the Main Console, click Retry for the failed job. Or, on the Task Status page, click Retry All, which retries all failed instances of the job.

  The failed job immediately restarts at the failed task, and changes the instance status from Failed to Reprocess.

Canceling All Instances of a Failed Job

Use the following procedure to cancel all instances of a failed job.

To cancel all instances of a failed job

- On the Task Status page, click Cancel All.

  All failed instances of the job are cancelled, and the job status changes to Canceled, and remains Canceled until the next time the job is scheduled to run again.

Canceling a Failed Job Instance

Use the following procedure to cancel a failed job instance.

To cancel a failed job instance

- On the Task Status page, click Cancel in the Action section next to the failed instance.

  The failed job instance is canceled, and the job instance status changes to Canceled, and remains Canceled until the next time the job is scheduled to run again.

Starting a Job Manually

You can run just one instance of the job immediately, overriding the scheduling parameters, except concurrency parameters. If you saved the schedule to run multiple occurrences of a job, running a job manually uses multiple occurrences instead of one.

To run an instance of a job immediately

- On the Command Center Main Console, click Run Now next to the job you want to run.

6 Scheduling Jobs

This chapter describes the tasks associated with scheduling production jobs, using blackout calendars to cease job processing, and using job alerts. It includes the following topics:

- Scheduling Jobs on page 65
- Process of Configuring a Blackout Calendar on page 67
- Creating a Blackout Calendar on page 68
- Applying a Blackout Calendar to a Job Schedule on page 69
- Process of Configuring Job Alerts on page 69
- Creating Alert Groups on page 70
- Creating Alert Profiles on page 71
- Applying Alerts to a Job Schedule on page 73

Scheduling Jobs

Jobs are not scheduled automatically; you must schedule jobs to run in a live production environment. You can change a job schedule anytime, except while the job is processing.

You can schedule a job to run on a daily, weekly, or monthly basis or establish a more complex timetable. The frequency with which you choose to run a job depends on both job type and your organization's needs. Consider all jobs and events in planning your schedule. You can also schedule a job to run just one time (not recurring). For information about when to schedule Payment jobs, see "Scheduling Payment Jobs" on page 77.

To use Scheduler, it must be configured and running. See *Installation Guide for Oracle Self-Service E-Billing* for details on configuring and starting Scheduler.

For each job, you can apply any of the following options:

- **A blackout calender.** Apply a calendar that specifies processing blackout days, such as holidays, shut-downs, and so on. For details on creating blackout calendars, see "Creating a Blackout Calendar" on page 68.
- **Job alerts.** Send an email notification when a job successfully completes processing, fails, or both. For details on using job alerts, see "Process of Configuring Job Alerts" on page 69.
- **Concurrency.** Multiple instances of a job can run at the same time.

To schedule a job

1. On the Command Center Main Console, verify that the job you want to schedule is not currently running, then click the value in the Next Run column for the particular application job.

   Alternately, you can specify the schedule after configuring the job; on the job configuration page, Click Submit Changes and Schedule, then click OK.

2. On the Job Schedule page, enter or select the date when you want the job schedule to begin:
   - To enter a date, use MM/DD/YYYY format.
   - To select a date, click Popup Calendar and choose a month and day.

3. In the Start Time field, select the time of day when you want the job to begin, in hours and minutes.

4. Some jobs require an resource, or input data, file. In these jobs, the scanner looks for a file one time only; if the file is not there, the job cannot run. If you are scheduling a job to run one time only (not recurring), you have the option to retry the job over a configurable time period until the resource file is available. For a non-recurring job, specify one of the following scan options:
   - **Try once.** To scan for a job resource file just one time. This option minimizes the amount of time the scanner process runs.
   - **Try every (#) minutes until (Time).** To scan for a resource file multiple times during a specified time period, select the number of times to try (scan) and the time to stop scan attempts.

   NOTE: You can also run a job immediately from the Job Schedule page; click Run Now.

5. Specify how frequently you want the job to run. Choose one of the following options:
   - **Do not repeat this event.** Runs the job one time only, on the date and time specified.
   - **Repeat (Every) (Day).** Runs the job once, on the basis you specify. To use this option, select a value from each of the following sets of options:
     - Every, Every other, Every third, Every fourth
     - Day, Monday, Tuesday, Wednesday, Thursday, Friday, Mon-Fri, Sat-Sun
   - **Repeat on day (#) of the month every (Month).** Runs the job one time only, on the basis of days or months you specify. To use this option, select a value from each of the following sets of options:
     - Day of the month (1-31)
     - Month, Other month, 3 months, 4 months, 6 months, 12 months
   - **Repeat Daily on every (#) hours:** Runs the job daily, at the hour intervals you specify. To use this option, select one of the following hourly intervals: 1, 2, 3, 4, 6, 8, 12. Also specify one of the following end periods for this option:
     - **Forever.** Use this schedule indefinitely.
     - **Until.** Use this schedule until a specified date. Enter or select the end date.

     If a job is still running when the next job process is scheduled to start, the next job process will be skipped, so only that one job process runs at a time.

6 Specify one of the following blackout processing options:

- **Always run (Ignore Blackout Dates).** Choose this option if you do not want to cease job processing on any dates.
- **Do not run in blackout dates defined in (Calendar name).** Choose this option if you want to cease job processing on the dates defined in a blackout calendar. Select a blackout calendar to apply. For details on creating a blackout calendar, see "Creating a Blackout Calendar" on page 68.

7 To use alerts with the job, make sure Activate Alerts is set, and choose one of the following options:

- **Use Global Alert Settings.** Use the alert groups and types configured for this job in the alert profiles (global alert settings). This is the default setting.
- **Use Alert Group with Alert Type.** Select an alert group and alert type (Success, Failure or both) to use for the job, overriding the global alert settings.
- **Use Contact Details with Alert Type.** Enter one or more email addresses, separated by a comma, and select an alert type to use for the job, overriding the global alert settings.

8 Specify whether you want to run instances of the job concurrently. Choose one of the following options:

- **Do not run multiple job instances, only one at a time.** By default, one instance of a job runs at a time.
- **Run maximum number of concurrent job instances.** If you choose this option, choose the maximum number of instances of this job to run concurrently: 5, 10, 15, or 20. By default, up to five instances of the job can run concurrently.

9 Click Save Schedule.

CAUTION: If you try to save schedule changes while a job is running in the production queue (job status indicates Processing), the new scheduling parameters are ignored.

Process of Configuring a Blackout Calendar

The blackout calendar feature lets you define days during which Oracle Self-Service E-Billing ceases processing one or more jobs. To black out dates on a job schedule, you create a calendar and specify which days are blackout dates, then associate the calendar with the job.

To configure a blackout calendar, perform the following tasks:

1 "Creating a Blackout Calendar" on page 68

2 "Applying a Blackout Calendar to a Job Schedule" on page 69

Creating a Blackout Calendar

To black out days for job processing, you must create a blackout calendar. You can apply one calendar to multiple jobs. You can also create multiple calendars, though only one calendar can be in effect for a particular application job at a time.

When creating a new calendar, you have the option to copy and modify an existing calendar.

This task is a step in "Process of Configuring a Blackout Calendar" on page 67.

To create a new blackout calendar

1. On the Command Center Main Console, click Settings.
2. Click the Calendar Settings tab.
3. Create a new calendar in one of the following ways:
   - **Create a new calendar.** Click the Create New Calendar tab.
   - **Copy and modify an existing calendar.** Click the Copy Calendar tab, then choose the calendar with the settings you want to copy.
4. Type a name for the new calendar and click OK.
5. Click a date to toggle it on or off as a blackout date. You can select Set Weekends as Blackout Dates to automatically select all weekend days for blackout job processing.

   Use Clear if you want to clear all your selections. To display months for the previous or next year, click the right and left arrows.
6. Click Save Calendar.

Related Topics

"Editing a Blackout Calendar" on page 68

"Deleting a Blackout Calendar" on page 69

Editing a Blackout Calendar

You can change the selected blackout dates on an existing calendar.

To edit a saved blackout calendar

1. Click Settings from the Command Center Main Console.
2. Click the Edit Calendar tab.
3. Choose a calendar to edit from the list and click OK.
4. Edit the calendar as needed and click Save Calendar.

Deleting a Blackout Calendar

You can delete blackout date job calendars from Oracle Self-Service E-Billing. Be sure to disassociate a calendar from any jobs using the Schedule Job screen. You cannot delete a calendar that is actively associated with a job schedule.

To delete a blackout date job calendar

1. Click Settings from the Command Center Main Console.
2. Click the Delete Calendar tab.
3. Make sure that the message No Associated Jobs appears below the name of the calendar you want to delete.
4. Click the Confirm Delete check box for the calendar you want to delete. You can check multiple calendars for deletion. Review your selections carefully.
5. Click Delete Selected Calendars.

Applying a Blackout Calendar to a Job Schedule

After creating a blackout calendar, you must apply the calendar to the schedule for the application job.

This task is a step in “Process of Configuring a Blackout Calendar” on page 67.

To apply a blackout date calendar to a job schedule

1. On the Command Center Main Console, click the value in the Next Run column for the particular application job.
2. Select the Do not run on Blackout Dates option.
3. Choose a blackout calendar from the list and click Save Schedule.

   Oracle Self-Service E-Billing ceases processing the selected job on the blackout dates in the selected calendar.

Process of Configuring Job Alerts

The alerts feature lets you send an email notification when a job successfully completes processing, fails processing, or both.

To implement email notification alerts for a job, perform the following tasks:

1. “Creating Alert Groups” on page 70 (Optional)
2. “Creating Alert Profiles” on page 71 (Optional)

3 "Applying Alerts to a Job Schedule" on page 73.

Creating Alert Groups

Alert groups let you define a list of email addresses (contacts) to receive email notification alerts. You can create multiple lists to use with different jobs, or just one alert group for many jobs. To create a new alert group you must provide the group a name and add one or more contact email addresses to notify.

This task is a step in "Process of Configuring Job Alerts" on page 69.

To create a new alert group

1 Click Settings from the Command Center Main Console.

2 Click the Alert Settings tab.

3 Click the Create Alert Groups tab. In the Group Name field, type a name for the new alert group.

4 Add the contact email address where you want Oracle Self-Service E-Billing to send the alert email message.

5 Click Add Contact. Oracle Self-Service E-Billing adds the contact to the list. (Note that you can edit or delete a contact in this table using the Edit and Delete options in this table.)

6 Continue adding contacts to the new alert group as needed. Click Save Alert Group.

7 You can now display the list of group contacts. Under Existing Alert Groups, select the group from the list and click View Group.

Related Topics

"Editing Alert Groups" on page 70

"Deleting Alert Groups" on page 71

Editing Alert Groups

Editing alert groups lets you rename a group, or view, add, edit or delete contacts from a group.

To edit an existing alert group

1 Click Settings from the Command Center Main Console.

2 Click the Create Alert Groups tab.

3 Click the Alert Settings tab.

4 Click the Edit Alert Groups tab. Choose an alert group to edit from the list. A list of current contacts for the selected group appears.

5 You can do any of the following:

- To change the group name, click Rename Alert Group, type a new name, and click OK.
- To add a contact to the group, type the email address in the Contact field and click Add Contact.
- Click Edit or Delete to edit or delete a contact.

6 Select Save Alert Group option to save your changes.

Deleting Alert Groups

You can delete alert groups from Oracle Self-Service E-Billing.

To delete an alert group

1 Click Settings from the Command Center Main Console.

2 Click the Alert Settings tab.

3 Click the Delete Alert Groups tab. Select the group you want to delete from the menu.

4 Click Delete Alert Group. At the confirmation prompt verify the group and click OK.

Creating Alert Profiles

The alert profile feature lets you associate a list of applications, jobs, and alert types with an alert group. One alert group can have multiple alert profiles associated. By default, Scheduler sends alerts to the groups configured in alert profiles when running job.

You can choose to base a profile on:

- **All Jobs.** Includes all jobs for all applications (DDNs) in the profile.
- **Selected Job Types.** Lets you select a list of specific job types (for all DDNs).
- **Selected DDNs.** Lets you select a list of DDNs (for all job types).
- **Selected Job Types in a Particular DDN.** Let you choose particular job types for a specific DDN.

For each profile you can send email notification to the associated alert group contacts when a job completes successfully, if a job fails, or both.

This task is a step in “Process of Configuring Job Alerts” on page 69.

To create a new alert profile for an alert group

1 Click Settings from the Command Center Main Console.

2 Click the Alert Settings tab.

3 Click the Configure Alert Profiles tab.

4 Under the Action column for the alert group, click Create.

5 Make sure the Alert Service Status is enabled.

6 Under Alert Category, choose one of the following options and follow any additional steps noted. Use Reset if you want to clear the screen selections at any point.

- **All Jobs**. To send alerts for all jobs and all application DDNs. This option is selected by default.
- **Selected Job Types**. To send alerts to specific job types for all applications, select this option, then click Configure.

  The Alert Category Configuration screen appears showing all job types in the Command Center. To select a job type, highlight it and click the double right-arrow. To deselect any job types, highlight the job type under Selected Job Types and click the double left-arrow. Click Save Job Types.
- **Selected DDNs**. To send alerts to specific applications, for all job types, select this option, then click Configure.

  The DDN Profile screen appears showing all DDNs defined in the Command Center. To select a DDN, highlight the name and click the double right-arrow. (To deselect a DDN, highlight the name under Selected DDNs and click the double left-arrow.) Click Save DDNs.
- **Selected Job Types in a Particular DDN**. To send alerts to specific application jobs only, select this option, then click Configure.

  The Alert Category Configuration screen appears. Choose a DDN from the list. To select a job type, highlight the name under All Job Types and click the double right-arrow. To deselect a job type, highlight the name under Selected Job Types and click the double left-arrow. Click Save Job Types.

7 Specify when to send alerts. Select one or both alert types:

- **Success.** When the job completes successfully.
- **Failure.** If the job fails.

8 In the Implementation of IAlertServicePlugin Interface field, enter:

```
com.edocs.common.notification.jobalerts.AlertServiceMessngerPlugIn.
```

Alternatively, if you have a custom alert service, you can enter the name of that custom service plug-in to use for this profile.

9 Click Submit to save the alert group profile configuration.

Related Topics

"Updating an Alert Profile" on page 72

"Deleting an Alert Profile" on page 73

Updating an Alert Profile

Use the following procedure to update a job alert profile.

To update a job alert profile

1. Click Settings from the Command Center Main Console.
2. Click the Alert Settings tab.
3. Click the Configure Alert Profiles tab. In the Action column for the job alert, click Update. The Alert Profile Configuration page appears.
4. Edit the alert profile as needed and click Submit.

Deleting an Alert Profile

Use the following procedure to delete a job alert profile.

To delete a job alert profile

1. Click Settings from the Command Center Main Console.
2. Click the Alert Settings tab.
3. Click the Configure Alert Profiles tab. In the Action column for the job alert, click Delete. Oracle Self-Service E-Billing removes the job alert profile.

Applying Alerts to a Job Schedule

By default, Scheduler is automatically set to enable alerts for all jobs and uses the alert groups and types defined in the alert profiles. You can override the alert profile configurations for a particular job at any time by changing the alert settings in Scheduler.

For a particular job you can:

- Turn the alert service off and on. By default, the alert service is on.
- Use the alert groups and alert types for the job as configured in the job alert profiles (also referred to as the *global* settings).
- Manually select a particular alert group and alert types to use, overriding the alert groups and alert types (success only, failure only, or both) configured for this job in the alert profiles.
- Manually enter a list of email contacts and alert types to use, overriding the alert groups and alert types (success only, failure only, or both) configured for this job in the alert profiles.
- Specify the name of a customized interface plug-in to use for the job, if you have created one.

This task is a step in “Process of Configuring Job Alerts” on page 69.

To apply alerts to a job in Scheduler

1. On the Command Center Main Console, click the link under the Next Run column for the particular application job.
2. Under the Alert Setting section, make sure Activate Alerts is selected.

3 Choose one of the following alert options to use for the job:

- **Use Global Alert Settings.** (Default) Use the alert groups and types configured for this job in the alert profiles (global alert settings).
- **Use Alert Group with Alert Type.** Select an alert group and alert type (Success, Failure or both) to use for the job, overriding the global alert settings.
- **Use Contact Details with Alert Type.** Enter one or more email addresses, separated by a comma, and select an alert type to use for the job, overriding the global alert settings.

4 In the Implementation of IAlertServicePlugin Interface field for this profile, enter:

```
com.edocs.common.notification.jobalerts.AlertServiceMessngerPlugIn
```

Alternatively, to use your own custom plug-in class or to override what is defined in the Job Alerts Profile configuration for this job, enter the full Java class path name.

5 Click Save Schedule.

The next time the job runs it will use these alerts.

7 Configuring the Payments Module

This chapter describes the features and procedures for setting up the Payment module for check and credit card transactions. It includes the following topics:

- About the Payment Module Features on page 75
- About the Transaction Manager and Payment Cartridges on page 76
- Scheduling Payment Jobs on page 77
- About Email Notifications on page 77
- Setting Up PayPal Payflow Pro Certification (IBM WebSphere Users Only) on page 78
- Configuring a Payment Gateway on page 79

About the Payment Module Features

The Payments module in Oracle Self-Service E-Billing is an electronic solution that decreases payment processing costs, accelerates receivables, and improves operational efficiency. It is complete payment scheduling and warehousing software with real-time and batch connections to payment gateways for Automated Clearing House (ACH) and credit card payments, and payments through various payment processing service providers.

The Payment module can act as a payment portal to host payments from different billers. For ACH, Payment can put checks from different billers (DDNs) into one ACH file, and can process a return file with checks from different DDNs. For information on configuring payment gateways, see "Configuring a Payment Gateway" on page 79.

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

- **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.

The Payment module performs authorization and settlement in one transaction using the credit card gateway for credit card payments.

The Payment module consists of the following features:

- Connections to payment networks:

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

- Advanced warehousing and scheduling:

  - Full payment warehousing to manage all of the scheduling, transaction, and business logic. Make one-time instant payments, schedule future payments, set up recurring and “auto-pay” payments, utilize threshold functionality, cancel, and change payments.
  - Supports ACH Notification of Changes (NOC), ACH addenda records, and multiple billers in one ACH file. Demand deposit account (DDA) verification before a payment is submitted through prenotes.
  - After the Oracle Self-Service E-Billing Payment module retrieves an invoice from Oracle Self-Service E-Billing, it keeps it in the Payment database. That allows customers to view invoices to make payments and view payment history.
- Integration with your existing infrastructure. Updates Accounts Receivables software with remittance information and supports reconciliation processes. Includes XML based API’s for integration into backend systems.
- Front-end GUIs:
  - Includes fully functional front-end Web pages, which can also be used as templates, enabling you to fully brand and customize your front-end interface.
  - Account history and access to details of past payments, providing an integrated view of all transactions, regardless of payment type or who initiated them
  - Payment reminders and a variety of customizable email templates available to the administrator as well as the end-user. Examples of email notification include enrollment status, recurring payment scheduling, and bill payment status.
- Administration tools, including:
  - Web-based configuration
  - Integration with the Oracle Self-Service E-Billing Command Center
  - Customer information management
  - Monitor activities and generate reports
- Database optimization for high-performance and scalability
- A rich SDK enables you to fully extend the solution, including API’s for two-way access and customizable front-end screens, jobs, and processes.

About the Transaction Manager and Payment Cartridges

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

A payment cartridge communicates with a payment gateway, to include ACH. The cartridge translates a payment object to a format understandable by the payment processors.

Payment uses the following payment cartridges:

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

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

- **Credit Cards.** Each credit card processor requires a specific credit card payment cartridge to process credit card payments. Payment supports PayPal Payflow Pro, which processes real-time online authorization and payment.

Scheduling Payment Jobs

You can schedule Payment jobs to run during periods of low customer activity, typically after midnight.

If two jobs access the same database table, schedule them to run at different times, as this could cause a database access error. Be sure to schedule the following jobs as follows:

- **Check jobs.** Run pmtRecurringPayment, pmtCheckSubmit, pmtCheckUpdate, pmtPaymentReminder, and pmtConfirmEnroll in that order.
- **Credit card jobs.** Run the pmtCreditCardSubmit job before running the pmtPaymentReminder job.
- **Enrollment jobs.** Run pmtSubmitEnroll before running the pmtConfirmEnroll job.

Use the pmtAllCheckTasks job or the job sequencing feature in Oracle Self-Service E-Billing to run all Payment module jobs sequentially (preventing specific jobs from running simultaneously). You can also customize your environment by editing the pmtAllCheckTasks job to not run specific jobs.

About Email Notifications

Email can be sent to customers to notify them of an action regarding a payment and about enrollment status. The customer email jobs are described in Table 39.

Table 39. Email Notification Jobs

pmtRecurringPayment	Sends email notification when a payment is scheduled by a recurring payment, and when the recurring payment expires.
pmtPaymentReminder	Reminds the customer when a payment is about to be made, when a payment has been successful, when a payment has failed, and when a payment has been returned.

Table 39. Email Notification Jobs

pmtCreditCardExpNotify	Notifies customers that their credit card is about to expire.
pmtAllCheckTasks	Email can also be sent to the payment administrator indicating whether payment jobs have finished successfully. Job status email is optional. It can be disabled for each payment gateway during payment gateway configuration.

The email message format and content is controlled by the email template files, the path to which is defined by the Email template status notification parameter in Payment Settings. The default templates are stored in the `$EDX_HOME/config` directory (or the `%EDX_HOME%\config` directory on Windows). The default email template files must be customized, which is usually done by Oracle Professional Services during installation. Contact Oracle Professional Services or your development team for more information. Email Template Notification describes the template files and describes the variables available for use in template files.

Setting Up PayPal Payflow Pro Certification (IBM WebSphere Users Only)

To use PayPal Payflow Pro as a gateway, IBM WebSphere users must follow this procedure to export the PayPal Payflow Pro digital certificate and link it in the IBM WebSphere console. (Oracle WebLogic users do not need to complete this task.)

To export and link the PayPal Payflow Pro certificate

1. Using Firefox 3.0 or higher, go to

   https://pilot-payflowpro.paypal.com/

2. Click the blue icon to the left of the URL. Click More Information.
3. In the Security window, click View Certificate.
4. Click the Details tab. Click Export to save the certificate.
5. Add the PayPal Payflow Pro certificate link in your IBM WebSphere console. Complete this step for both the Billing and Payment and the Command Center profiles:
   a. Log in to the IBM WebSphere console.
   b. Select Security, SSL certificate and key management, Key stores and certificates, NodeDefaultTrustStore, Signer certificates, and Add signer certificate.
   c. In the Alias field, type payflow.

d In the File name field, specify the full file path where the certificate you downloaded is located, such as $eBilling/pilot-payflowpro.paypal.com.

6 Save and restart the application server.

Configuring a Payment Gateway

You must configure at least one payment gateway for ACH or credit card payments for a BILLERID application (which you set up on the Main Console as a DDN). You can also update a payment gateway at any time to add or remove information about a particular payee.

The payment gateway or the biller's bank must provide information specific to a payment gateway. You must have this information in-hand before configuring.

You can specify global settings for applications (if needed, you can override some of these settings in certain job configurations for each application).

To configure a payment gateway and specify global configuration settings

1 From the Command Center menu, click Settings, then click the Payment Settings tab. Click Global Configuration. Specify the settings you want to use as global defaults. Select the payee DDN application and specify the global configuration parameters shown in Table 40 on page 80.

Select your BILLERID application in the Payee DDN section (selecting the same application under the External DDN section), and click Update.

2 Click Indexer DDN Configuration. For your BILLERID application, specify the parameters shown in Table 41 on page 82. Click Create.

3 Click Create next to the check gateway type. Select a payment gateway (such as ACH) from the list and click Add. Specify the payment gateway settings for checks in Table 42 on page 84, Table 43 on page 85, and Table 44 on page 88. Click Add.

If you are migrating, use the same settings as the old payment configuration that you wrote down before deleting the old configuration.

NOTE: Fields with an asterisk (*) are required.

4 For the same DDN, click Create next to the ccard (credit card) gateway type. Select paypal from the list, and click Add. Specify the credit card gateway settings in Table 42 on page 84, Table 46 on page 89, and Table 47 on page 91. Click Add.

5 Click Payee Account Configuration, then Click New. Select the BILLERID payee DDN. Specify bank account details for the application. You must configure at least one payee bank account for each payee DDN. For details on configuring parameters for a payee bank account, see "Parameters for Configuring a Payee Bank Account" on page 92.

6 Click Add to save the settings.

NOTE: It is possible to use multiple payee bank accounts for the BILLERID application. For help using the external DDN feature, contact your Oracle sales representative to request assistance from Oracle's Professional Services.

Parameters for Configuring Global Payment Settings

Table 40 describes the global parameters required when configuring the Payment module.

Table 40. Parameters for Global Payment Configuration

Parameter	What to Enter or Select
Payee DDNs	Choose one or more payment DDNs to configure.
Payment Notification Service Implementation	Always select com.edocs.common.notification.payment.ConsolidatedPaymentNotificationService' class.
Payment Notification Service Implementation	Always select com.edocs.common.notification.payment.ConsolidatedPaymentNotificationService' class.
Enable Payment Audit	Selecting 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. The value N disables auditing. The default is Y.
Enable Recurring Payment Audit	Selecting 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. The value N disables auditing. The default is Y.
Enable Payment Account Audit	Selecting 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. The value N disables auditing. The default is N.
Enable Payment Reminder Audit	Selecting Y causes any actions that affect the payment_reminders table to be audited. These actions can be from the Web application or from the Payment jobs. The value N disables auditing. The default is N.
Enable Bill Summary Audit	Selecting 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. The value N disables auditing. The default is Y.
Email Content Audit Length	Specifies how many characters of an email's content are 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	Selecting Y causes emails sent out by the recurring payment job to be audited. The value N disables email auditing (default).
Enable Payment Reminder Notification Audit	Payment reminders sends two types of email: remind-pay-bill emails and payment status notification emails. The value 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.

Table 40. Parameters for Global Payment Configuration

Parameter	What to Enter or Select
Enable Payment Status Notification Audit	Selecting Y causes payment status notification emails to be audited for both check and credit card payments. The value N disables payment status notification email auditing (default).
Enable Payment Account Enrollment Notification Audit	Selecting Y causes enrollment notification emails sent to be audited. The value N disables enrollment notification emails auditing (default).
Enable Credit Card Expiration Notification Audit	Selecting Y causes credit card expiration emails sent by the pmtCreditCardExpNotify job to be audited. The value N disables credit card expiration emails auditing (default).
Enable Payment Status Notification Audit	Selecting Y causes payment status notification emails to be audited for both check and credit card payments. The value N disables payment status notification email auditing (default).
Enable Payment Account Enrollment Notification Audit	Selecting Y causes enrollment notification emails sent to be audited. The value N disables enrollment notification emails auditing (default).
Enable Credit Card Expiration Notification Audit	Selecting Y causes credit card expiration emails sent by the pmtCreditCardExpNotify job to be audited. The value N disables credit card expiration emails auditing (default).

Parameters for Configuring an Indexer DDN

Table 41 describes the parameters you configure for an Indexer DDN.

Table 41. Parameters for Configuring an Indexer DDN

Parameter	What to Enter or Select
Implementation of com.edocs.payment.imported.IBillDepot	The parameter settings are: ■ **com.edocs.common.services.payment.BillDepot**. Retains the default handling for bill processing. ■ **com.edocs.payment.imported.eadirect.BillDepot**. This field is backward compatible. ■ **com.edocs.payment.imported.eadirect.SampleBillDepot**. This field is backward compatible. It adds the following features: ■ If the rebill has a zero amount, the due date is set to the previous bill's due date. A *rebill* is a bill that is reissued during the current billing cycle. A rebill occurs when a biller makes frequent adjustments to a bill before the due date. ■ If the due date is ON RECPT, the due date is changed to the current date.
The Name of Due Date in Statement Index Table (for recurring payment)	The name of the field you have defined to extract the Due Date from each statement. This field must be routinely indexed in the Oracle Self-Service E-Billing database, during data processing, for use in the Payment module. (Keep the default value; this field is backward compatible.) This field only applies to recurring payments. If you are not implementing recurring payments, you can leave this field empty.
Due Date Format	Selected from the 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. (Keep the default value; this field is backward compatible.)

Table 41. Parameters for Configuring an Indexer DDN

Parameter	What to Enter or Select
Name of Amount Due in Statement Index Table (for recurring payment)	The name of the field you have defined to extract the Amount Due from each statement. This field must be routinely indexed in the Oracle Self-Service E-Billing database, during data processing, for use in the Payment module. (Keep the default value; this field is backward compatible.) This field only applies to recurring payments. If you are not implementing recurring payments, you can leave this field empty.
Due Date Format	Selected from the 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. (Keep the default value; this field is backward compatible.)
Name of Minimum Amount Due in Statement Index Table (if applies)	Keep the default value; this field is backward compatible.

Parameters for Configuring Check and Credit Card Gateways

This topic describes the parameters you set to configure payment gateways for checks and credit cards for regular and external payee DDNs.

Note that some parameters are required for both checks (ACH) and credit card gateways (for both regular and external DDNs), and are described in "Check and Credit Card Gateway Configuration Parameters (Regular and External Payee DDNs)."

Check and Credit Card Gateway Configuration Parameters (Regular and External Payee DDNs)

Table 42 shows the parameters you configure for the credit card cartridge (if any) and check cartridges for both regular and external payee DDNs.

Table 42. Parameters for Configuring a Credit Card Gateway (Regular and External Payee DDNs)

Parameter	What to Enter or Select
Batch Size for Payment Reminder Table	Specifies the number of payment reminders to be read into memory from the Oracle Self-Service E-Billing database for the pmtReminder job. Note that specifying a batch size that is too small increases the number of times the database is accessed, and specifying a batch size value that is too large could result in an excessive amount of memory being used. A batch size of 100 is suggested for a medium-sized database, and a batch size of 1000 is suggested for a large database. You can enter a batch size of 0 (zero) to disable batched table reads, however, doing so requires a lot of memory. Entering zero means that one partition ID is created for all payments, and that all payments are processed at once instead of in batches. The resulting batch file does not have multiple batch records, which some banks prefer.
Send Email Notification when Payment Jobs are Done (with or without error)	The value 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-to addresses (separated by ; semicolon) for job status notification	One or more email addresses that must be sent job status notification, separated by semicolons.
JNDI name of IAccount	Implementation of IAccount, which must match the enrollment model (single or multiple DDNs for each user account) used by applications that use this payment gateway (DDN). Select the JNDI name edx/ejb/AdminAccount.
Implementation of IUserAccountAccessor	The name of the class that handles getting Oracle Self-Service E-Billing user information, which is determined by the type of enrollment supported. Select the class com.edocs.common.services.payment.plugin.DummyUserAccountAccessor.

Table 42. Parameters for Configuring a Credit Card Gateway (Regular and External Payee DDNs)

Parameter	What to Enter or Select
Implementation of IPaymentAccountAccessor	The name of the class that handles getting Payment user information. Select the class com.edocs.payment.payenroll.payacct.SSOPaymentAccountAccessor.
Make Authorize Reversal Payments	Y, Yes or N, No

Check Gateway Configuration Parameters (Regular Payee DDN Only)

Table 43 describes the parameters you must specify when configuring a check gateway (for a regular payee DDN only).

Table 43. Parameters for Configuring a Check Gateway (Regular Payee DDN Only)

Parameter	What to Enter or Select
Payment Type	Check
Gateway	ACH
Batch Size for Check Payment Table	The number of scheduled checks to read into memory from the payment database as a batch job. Note that specifying a batch size that is too small increases the number of database accesses, and specifying a batch size value that is too large might result in an excessive amount of memory being used. A batch size of 100 is suggested for a medium-sized database, and a batch size of 1000 is suggested for a large database. A batch size of 0 (zero) can be entered to disable batched table reads, but it is not recommended because it requires a lot of memory. Entering zero means that one partition ID is created for all payments, and that all payments are processed together, instead of in batches. The resulting ACH file will not have multiple batch records, which some banks prefer.
Number of Business Days After Pay Date to Clear the Check	The number of business days for a check to be marked as paid after submitting to the payment gateway without returns or failures. The default is five business days.
NDays to Activate Pending Subscribers	The number of business days to wait before approving a customer for online payment. If a prenote file is returned before this time, then the customer is rejected for the causes stated in the prenote file. The default is three days.
Batch File Name Prefix	The text used as a prefix to each batch file name.
Batch File Name Time Stamp	The format used for each batch file time stamp.

Table 43. Parameters for Configuring a Check Gateway (Regular Payee DDN Only)

Parameter	What to Enter or Select
Batch File Name Suffix	The text used as a suffix to each batch file name.
ACH Prenote File Name Prefix	The text used as a prefix to each ACH prenote file name.
ACH Prenote File Name Time Stamp	The format used for each ACH prenote file name
ACH Prenote File Name Suffix	The text used as a suffix to each ACH prenote file name
Update Payment Enrollment in Case of NOC	The value Y causes the pmtCheckUpdate job to update enrollment status when a NOC is returned. The value N means this value is updated by the customer through the user interface
Send Email Notification in Case of NOC	The value Y sends email to customers whose payment returns a NOC. The value N means the customer does not receive email when a NOC is returned.
Skip non-business days for batch entry effective dates	Whether batch effective dates skip U.S. Federal holidays and weekends.
Generate an empty ACH file if no checks to submit	Whether the pmtCheckSubmit job generates an empty ACH file if there are no checks to submit.
Immediate Destination	The routing number of the Biller (DDN). A routing number 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).A routing 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 might 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.

Table 43. Parameters for Configuring a Check Gateway (Regular Payee DDN Only)

Parameter	What to Enter or Select
Immediate Destination Name	The name of the Biller. This information is assigned to you by your bank, and follows a format specific to the needs of the ODFI. The pmtCheckSubmit job inserts this information into the ACH File Header record's Immediate Destination Name field. The pmtCheckUpdate job uses this value to validate entries in the ACH return file.
Immediate Origin Name	The name of the biller's bank (ODFI). This information is assigned to you by your bank, and follows a format specific to the needs of the ODFI. The pmtCheckSubmit job inserts this information into the ACH File Header record's Immediate Destination Name field. The pmtCheckUpdate job uses this value to validate entries in the ACH return file.
ACH File Output Directory	The directory where ACH files are created to be sent to the originating bank. Payment does not create this directory.
ACH File Input Directory	The directory where the originating bank sends ACH return files. Payment does not create this directory.
ACH Template File Directory	The directory where the ACH XML files are stored. The path for default templates is `$EDX_HOME/payment/lib/payment_resources/ach/template`.
Implementation of IAchCheckSubmitPlugIn	The plug-in allows modification of whether a check payment is submitted, plus other actions based on a check selected for payment. For example, to generate a remittance file with a format different from the standard ACH file specification. For information about implementing this class, contact Oracle Professional Services. The default is *com.edocs.payment.cassette.ach.AchCheckSubmitPlugIn*.
Flexible Field 1 and 2	Specify up to two fields for the IAchCheckSubmitPlugin plug in.

Check Gateway Parameters (External Payee DDN Only)

Table 44 describes the parameters you specify when configuring a check gateway for an external DDN only.

Table 44. Parameters for Configuring a Check Gateway (External Payee DDN Only)

Parameter	What to Enter or Select
Batch Size for Check Payment Table.	Specifies the number of scheduled checks to read into memory from the payment database as a batch job. Note that specifying a batch size that is too small increases the number of database accesses, and specifying a batch size value that is too large might result in an excessive amount of memory being used. A batch size of 100 is suggested for a medium-sized database, and a batch size of 1000 is suggested for a large database. A batch size of 0 (zero) can be entered to disable batched table reads, but it is not recommended because it requires a lot of memory. Entering zero means that one partition ID is created for all payments, and that all payments are processed at once, instead of in batches. The resulting ACH file will not have multiple batch records, which some banks prefer.
Number of Business Days After Pay Date to Clear the Check.	The number of business days for a check to be marked as paid after submitting to the payment gateway without returns or failures. The default is five business days.
Days to Activate Pending Subscribers.	The number of business days to wait before approving a customer for online payment. If a prenote file is returned before this time, then the customer is rejected for the causes stated in the prenote file. The default is three days.
Batch File Name Prefix	The text used as a prefix to each batch file name.
Batch File Name Time Stamp	The format used for each batch file time stamp.
Batch File Name Suffix	The text used as a suffix to each batch file name.
Batch File Output Directory	The name of the directory to place the batch output file.

ACH Federal Holidays

You can configure an ACH check payment gateway to skip non-business days for the batch-effective entry date. The Skip non-business days for batch effective entry date field lets you determine when a payment must be made.

NOTE: 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.

Non-business days in the Payment module include the U.S. Federal holidays listed in Table 45.

Table 45. U.S. Federal Holidays

Holiday	Description
New Year's Day	January first.
Martin Luther King's Birthday	The third Monday in January.
Presidents' Day	The third Monday in February.
Memorial Day	The last Monday in May.
Independence Day	The 4th of July.
Labor Day	The first Monday in September.
Columbus Day	The second Monday in October.
Veterans' Day	The 11th of November.
Thanksgiving	Fourth Thursday in November.
Christmas Day	The 25th of December.

Credit Card Gateway Parameters (Regular Payee DDN Only)

Table 46 describes the parameters you must specify when configuring a credit card gateway for a regular payee DDN only.

Table 46. Parameters for Configuring a Credit Card Gateway (Regular Payee DDN Only)

Parameter	What to Enter or Select
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 increases the number of times the database is accessed, and specifying a batch size value that is too large might result in an excessive amount of memory being used. A batch size of 100 is suggested for a medium-sized database, and a batch size of 1000 is suggested for a large database. A batch size of 0 (zero) can be entered to disable batched table reads, but is not recommended because it requires a lot of memory. Entering zero means that one partition ID is created for all payments, and that all payments are processed together, instead of in batches. The resulting batch file will not have multiple batch records, which some banks prefer.
PayPal Host Name	The URL to the PayPal Payflow Pro host that processes credit card transactions for this payment gateway: pilot-payflowpro.paypal.com.
PayPal Host Port	The TCP port number to be used when contacting the PayPal Payflow Pro host. The default is 443.

Table 46. Parameters for Configuring a Credit Card Gateway (Regular Payee DDN Only)

Parameter	What to Enter or Select
PayPal Timeout Period for Transaction	The number of seconds the pmtCreditCardSubmit job will wait for a transaction to complete with the PayPal Payflow Pro host before timing out.
Proxy Address	The address of the proxy server: www-proxy.us.yourcompanydomain.com. Optional field; if you are using the proxy network connection, enter your proxy setting.
Proxy Port	The port number of the proxy server: 80. Optional field; if you are using the proxy network connection, enter your proxy setting.
PayPal User	The case-sensitive vendor name from PayPal Payflow Pro. See your PayPal representative for this value. (This parameter must match the PayPal Vendor.)
PayPal Vendor	The case-sensitive vendor name assigned by registering with PayPal Payflow Pro. See your PayPal representative for this value.
PayPal Partner	The partner name used by PayPal: PayPal.
PayPal Password	The case-sensitive password. See your PayPal representative for this password.
URL StreamHandler class of application server	The name of the class used for the application server StreamHandler: ■ **Oracle WebLogic.** Use sun.net.www.protocol.https.Handler. ■ **IBM WebSphere 6.1.** Use com.ibm.net.ssl.www2.protocol.https.Handler.
PayPal Certificate Path	The path to the SSL server certificate purchased from PayPal Payflow Pro and installed on the application server.
Number of Threads	Specifies the number of connections to open with the PayPal Payflow Pro payment gateway at one time. More threads consume more resources (including 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 PayPal address verification service	The value Y (Default) enables address verification for credit card payments. AVS support must also be set up with PayPal Payflow Pro.

Table 46. Parameters for Configuring a Credit Card Gateway (Regular Payee DDN Only)

Parameter	What to Enter or Select
Implementation of IPaypalCreditCardSubmitPlugIn	The plug-in allows modification of whether a credit card payment is submitted, plus other actions based on the payments selected for settlement. For example, to deny a credit card payment based on additional business rules. For information about implementing this class, contact Oracle Professional Services. The default is: com.edocs.payment.cassette.paypal.PaypalCreditCardSubmitPlugIn
Flexible Field 1 and 2	Specify up to two parameters for a plug-in.

Credit Card Gateway Parameters (External Payee DDN Only)

Table 47 describes the parameters you must configure for a credit card payment gateway using PayPal Payflow Pro (external payee DDN only).

Table 47. Parameters to Configure for a Credit Card Gateway (External Payee DDN Only)

Parameter	What to Enter or Select
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 increases the number of times the database is accessed, and specifying a batch size value that is too large might result in an excessive amount of memory being used. A batch size of 100 is suggested for a medium-sized database, and a batch size of 1000 is suggested for a large database. A batch size of 0 (zero) can be entered to disable batched table reads, but it is not recommended because it requires a lot of memory. Entering zero means that one partition ID is created for all payments, and that all payments are processed together, instead of in batches. The resulting batch file will not have multiple batch records, which some banks prefer.
Batch File Name Prefix	The text used as a prefix to each batch file name.
Batch File Name Time Stamp	The format used for each batch file time stamp.
Batch File Name Suffix	The text used as a suffix to each batch file name.
Batch File Output Directory	The name of the directory to place the batch output file.

Parameters for Configuring a Payee Bank Account

Table 48 shows the parameters you configure when adding a new payee bank account.

Table 48. Parameters for Configuring a New Payee Bank Account

Payee	The name of the payee DDN. The payee bank account will be assigned to this payment DDN.
Company ID	This information is assigned to you by your bank, and follows a format specific to the needs of the ODFI. The pmtCheckSubmit job inserts this information into the ACH Batch Header record's Company ID field. The pmtCheckUpdate job uses this value to identify the biller in the ACH return file.
Company Name	The name given to the Biller by the Biller's bank (ODFI). This information is assigned to you by your bank, and follows a format specific to the needs of the ODFI. The pmtCheckSubmit job inserts this information into the ACH File Header record's Company Name field. The pmtCheckUpdate job uses this value to identify the biller in the ACH return file.
Company Entry Description	Describes the purpose of the detail records, and is dependent on the type of details records. For example, this could be Gas Bill if the ACH Detail records following this Batch Header record are of type PPD. This value is 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 is provided by the payment gateway. The pmtCheckSubmit job inserts this information into the ACH Batch Header record's Originating DFI ID field.
Business Unit	The name of the business unit associated with this account.
Vertical Field 1, Vertical Field 2, Vertical Field 3	Fields that can be customized for use with your application.
Flexible Field 1, Flexible Field 2, Flexible Field 3	Fields that can be customized for use with your application.

8 Payment Transactions

This chapter describes the types of payment transactions and how to view information about them. It includes the following topics:

- About Check Payment Transactions on page 93
- About Credit Card Payment Transactions on page 98
- Viewing Payment Reports on page 102

About Check Payment Transactions

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

The Payment module in Oracle Self-Service E-Billing supports check payments through the Automated Clearing House (ACH) payment gateway. A user can schedule a check payment using any of the unlimited number of accounts for each user that are available for the ACH check payment gateway.

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

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

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

Adding a New Customer Account for Check Payment Services

The following actions describe the process of enrolling a new user who specifies a checking account at enrollment:

1. A new customer enrolls for check payment services by completing an check account enrollment form, which saves the check account information in the payment_accounts table with a status of active.
2. The customer can optionally receive an email about enrollment status.

The Cycle of an ACH Check Payment Transaction

Figure 1 shows the entities in an ACH payment transaction.

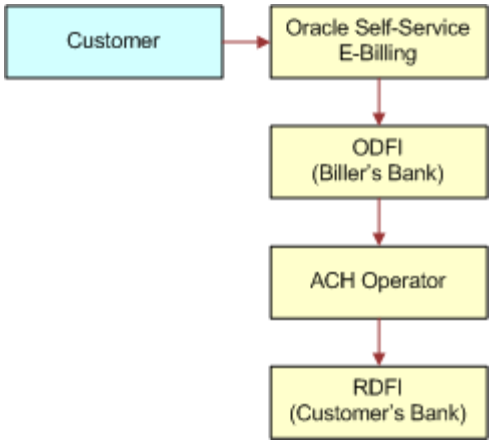

Figure 1. Check Payment Transactions

A typical ACH check payment transaction cycle (excluding transfers between the ODFI, ACH operator and RDFI) is as follows:

- A customer logs in and schedules a new payment from the list of defined checking accounts. The Payment module in Oracle Self-Service E-Billing 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.
- 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.
- 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 the Payment module.
- The pmtCheckUpdate job runs. If there is no return code, and five business days (default) have passed, pmtCheckUpdate changes the status of the check from processed to paid.
- If the payment gateway returns the check, the pmtCheckUpdate job updates the check's status to returned, and saves the reason code in the txn_err_msg field of the check_payments table. An exception report is generated to summarize the information in the returned file, which can be viewed from Command Center.
- If there is an error other than returned, pmtCheckUpdate changes the check status to failed.

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

Supported SEC Codes

The following SEC Codes (Standard Entry Class Codes) are supported for ACH:

- **Web.** Internet Initiated Entry (default for the Payment module).

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

- **PPD.** Prearranged 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.** Represents 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

Table 49 lists some of the ACH change codes (also known as *NOC codes*) that can appear in the returns file after running the pmtCheckUpdate job if previously valid payment information is now incorrect or out-of-date.

Table 49. ACH Change Codes

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

Table 50 lists some of the ACH return codes that can appear in the returns file after running the pmtCheckUpdate job.

Table 50. ACH Return Codes

ACH Return Code	Description
R01	Insufficient Funds
R02	Account Closed
R03	No Account or Unable to Locate Account
R04	Invalid Account Number
R05	Reserved
R06	Returned for each ODFI's Request
R07	Authorization Revoked by Customer (adjustment entries)
R08	Payment Stopped or Stop Payment on Item
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*

where:

- *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. The Payment module checks the scheduled payment date to determine 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.

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

The Payment module 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 customizing is required to support this feature. Theoretically, you can put any information into an addenda record, for example, the invoices of a payment. To add addenda records, you must write a plug-in for the pmtCheckSubmit job. Contact Oracle Professional Services or your development team for more information about supporting ACH addenda records.

Multiple DDNs in ACH Files

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

About Check Payment Status Flow

Figure 2 shows the possible states of a check payment and the jobs that can change the state.

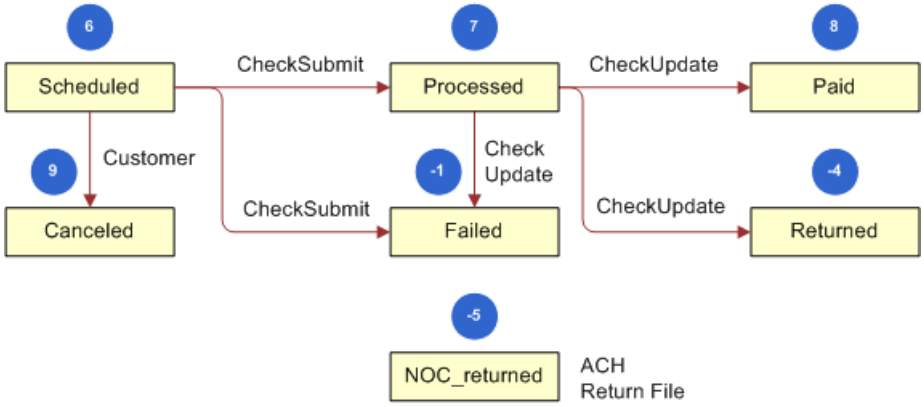

Figure 2. Check Payment Status Flow

Table 51 lists the status values that can occur during a check payment transaction cycle. The numeric values shown are the actual values saved in the Payment module database.

Table 51. Status Values for Check Payment Transactions

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

About Credit Card Payment Transactions

The user interface to the Payment module can offer a variety of credit card payment options. Some of those options require you to configure fields in Payment Settings for a credit card payment gateway. Oracle Self-Service E-Billing supports credit reversals.

Credit card processing usually occurs in the following order:

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

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

3 The merchant issues a settlement request to the issuing bank so that the money can be transferred. The merchant usually does this request 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 so that a credit card payment is settled at the same time it is authorized.

Because credit card processing is in real-time and not batch-based, the life cycle for credit cards is not as complex as check processing.

Figure 3 shows the entities involved in a credit card payment transaction.

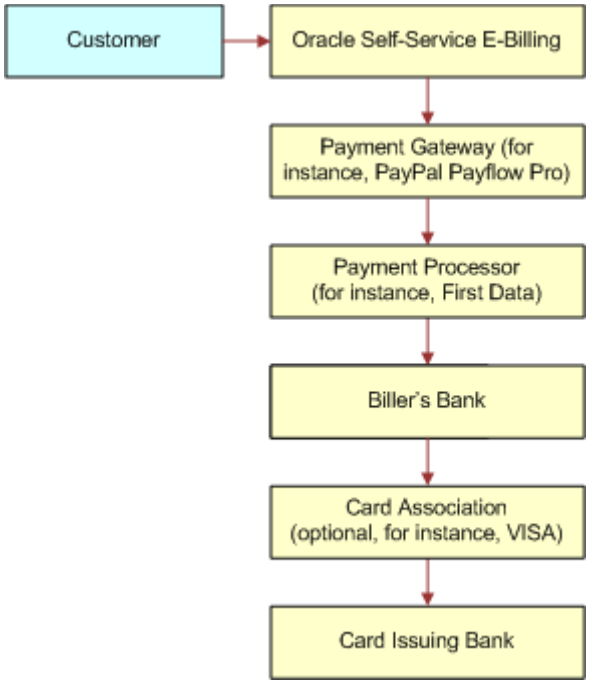

Figure 3. Credit Card Payment Transactions

About Instant Credit Card Payments

The states for an instant credit card payment can be settled, failed-authorize, or failed. For instant payments, there is no scheduled state.

Instant credit card payments are processed in the following order:

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

2 The Payment module sends the payment to a credit card cartridge in real time.

3 The Payment module sets the state of the credit card transaction to one of the following:

- If the card is authorized and settled, it sets the state to settled.

- If the card failed to authorize, it sets the state to failed_authorize.
- If there is a network problem, it sets the state to failed.

4 The Payment module inserts the payment into the creditcard_payments table.

5 The Payment module presents the result of the transaction to the user.

6 The pmtPaymentReminder job runs and (optionally) sends email to users who have made an instant payment.

About Scheduled Credit Card Payments

Scheduled credit card payments process in the following order:

1 The customer schedules a credit card payment from the UI, and the Payments module marks the payment as scheduled in the creditcard_payments table.

2 Before the pmtCreditCardSubmit job process the scheduled credit card payment, a user can modify or cancel it.

3 When the pmtCreditCardSubmit job runs, it selects all credit card payments that are scheduled to be paid at the time the job runs, opens a connection to the credit card payment gateway, and starts making payments. The Number of days before a credit card's pay date for it to be submitted parameter on the pmtCreditCardSubmit job determines how many days ahead to look when selecting payments to be made.

   If the IPaypalCreditCardSubmitPlugIn 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 plug-in are called before and after credit card payment processing. For information about configuring job plug-ins, contact Oracle Professional Services.

4 The credit card gateway sends the transactions to the credit card processor. The credit card processor either authorizes and settles the credit card payment, or rejects it. The results are returned to the credit card gateway, which forwards the results to the pmtCreditCardSubmit job.

5 The pmtCreditCardSubmit job changes the status of the credit card payment in the database depending on the transaction status returned by the credit card processor, and optionally sends email to the customer about the status of the payment.

- If the card is authorized and settled, it sets the state to settled.
- If the card fails to authorize, it sets the state to failed_authorize.
- If there is a network problem, the state remains scheduled, so it can process the next time pmtCreditCardSubmit runs.

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

About Credit Card Payment Status

Table 52 lists the status values that can occur during a credit card payment transaction cycle. The values in parentheses () are the actual values saved in the Payment module database.

Table 52. Credit Card Payment Transaction Cycle

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

About the 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 might 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, PayPal Payflow Pro) for address comparison. If AVS is turned on by an Oracle Self-Service E-Billing administrator, address information passes into PayPal Payflow Pro as part of the PayPal Payflow Pro request. PayPal Payflow Pro 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 PayPal Payflow Pro. The credit card issuing bank then replies back to PayPal Payflow Pro whether information matched (address and zip code are checked during AVS). The value Y means yes, N means no, and X means a match cannot be determined. PayPal Payflow Pro then accepts or rejects (voids) the transaction based on the filter set through the Payment module (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.

The Payment module does not handle these by default, but the pmtCreditCardSubmit job has a plug-in to allow custom code pass in the AVS values. For more information about AVS functionality, go to

https://www.paypal.com/payflow-support

If the Payment module does not send the address information to PayPal Payflow Pro, or if the Oracle Self-Service E-Billing administrator did not turn on AVS and the AVS check level is set to Full, the transaction fails. If the card issuer address is sent to the payment gateway, but the address does not match the information on the gateway, then the gateway can send an AVS code. If an AVS code is received, the Payment module logs the AVS code in the audit tables.

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

Viewing Payment Reports

The Payment module keeps payment history for auditing purposes. Checks go through a list of states before clearing. For insert, update and insert, and update operations, the Payment module keeps a copy in the check_payments_history table. The Payment module records when the check was created, when the check was updated or cancelled, when the check is processed, and other check actions.

The Payment module also logs additional important information, warnings, and errors. Based on the format of the logging messages, the Payment module can connect with other monitoring software, for example, EMC patrol. Credit card gateways report only credit card transactions.

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

- Daily Summary
- Daily Exceptions

To view Payment reports

1. In the Command Center, click Reporting.
2. Click the Payment Reports icon.

   The Search Payment Report page appears.
3. Select a payee and a report type.
4. Enter the date on which you want to the search to run. You can use the Popup Calendar to help you determine the date.
5. Click Search.

   Oracle Self-Service E-Billing generates the report.

9 Recurring Payments

This chapter explains recurring payment processing. It includes the following topics:

- About Recurring Payments on page 103
- About the Multiple Recurring Payment Feature on page 104
- About the Recurring Payment Transaction Cycle on page 104
- Database Tables Affected by Recurring Payments on page 106
- Examples of Recurring Payment on page 107

About Recurring Payments

For checks and credit cards, the Payment module provides two types of recurring payments:

- 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 National Automated Clearing House Association (NACHA) 2009 specification. NACHA 2009 defines a payment as recurring when the account manager keeps the account information in a database.

A user can modify or cancel recurring payments at any time before the payment is scheduled. Recurring payment allows a customer to make payments automatically, based on the amount and pay date. To configure recurring payments, see “Configuring a pmtCreditCardSubmit Job” on page 48.

There are several kinds of recurring payments:

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

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

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

For monthly payments, if day 29, 30, or 31 is selected, and that day does not exist for a particular month, Oracle Self-Service E-Billing uses the last day of that month by default. For example, specifying day 31 of each month ensures that payments are made on the last day of each month.

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

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

After an end-customer creates a recurring payment, that customer is not permitted to change the payment amount from fixed to (minimum) amount due, or to change the pay date from fixed to before due date, or before due date to fixed. 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.

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

About the Multiple Recurring Payment Feature

The multiple recurrent payment feature lets a customer, or payee, select multiple biller accounts to pay with a single recurring payment setup. It is also possible to set up one recurring payment for a collection of bills from different accounts that come from different indexer DDNs. Payment entities like recurring payments, bank accounts, and payments made are visible to and can be modified by all authorized persons.

About the Recurring Payment Transaction Cycle

Oracle Self-Service E-Billing saves recurring payment information in the recurring_payments table. The pmtRecurringPayment job retrieves bills from Oracle Self-Service E-Billing, makes payments by either check or credit card, and sends email notifications for recurring payments. The pmtRecurringPayment job performs two functions:

- Contacts Oracle Self-Service E-Billing to get the latest bill for a recurring payment that a customer set up. This process is called *synchronization*. A recurring payment can only be synchronized with Oracle Self-Service E-Billing if it is 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 will not be synchronized with Oracle Self-Service E-Billing, which means there is no bill information associated with this recurring payment.
- Schedules payments (inserts a payment with status of scheduled in the check_payments or creditcard_payments table so that the payments are processed). This process is called *scheduling*. A payment is scheduled three days before the pay date (by default). The number of days can be changed by changing the Number of days before pay date to schedule the payment field in the job configuration. This delay allows the customer to modify or cancel this payment before the payment is processed by the pmtCheckSubmit or pmtCreditCardSubmit jobs.

Table 53 shows the columns updated in the recurring_payments table by the pmtRecurringPayment job.

Table 53. Recurring Payment Transaction Cycles

Column Updated in the recurring_payments Table	Description
bill_scheduled	Y or N. This field determines whether the current bill associated with the recurring payment has been scheduled (inserted) into the check_payments or creditcard_payments columns. It is always N for a fixed amount and fixed pay date.
Status	Active or Inactive. This status is calculated internally. The value 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 the last processed time and the current job running time (inclusive) are synchronized. By default, the value in the last_process_time column 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 the start date will not be synchronized.
last_pay_date	The pay date of last payment made. This value 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 the values in the start_date, last_pay_date and pay_interval columns.
bill_id	A foreign key reference to a row in the payment_bill_summaries table. Use the bill ID to retrieve the latest bill information paid by the recurring payment. It could be null if there is no such bill.
curr_num_payments	The current number of payments made.

TIP: There is no payment inserted into check_payments or creditcard_payments table when a recurring payment is created by the user. Payments are inserted by the pmtRecurringPayment job.

Database 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 as a result of the following actions:

- 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 retrieved from the Oracle Self-Service E-Billing 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. Payment information is scheduled into the check_payments (for check) or creditcard_payments (for credit card) tables.

Examples of Recurring Payment

This topic describes four examples of recurring payment with additional details about the relevant database interactions.

Example of Amount Due and Before Due Date

This example shows how Oracle Self-Service E-Billing processes a recurring payment using the amount due before the due date.

1. On date 04/09/2009, 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 04/10/2009, and the end date is 06/10/2009. The following table gives an example of a recurring payment created with Amount Due and Before Due Date.

Column Name	Value
payer_account_number	acct1111
bill_scheduled	Y
status	Active
last_process_time	04/10/2009. This value is the same as start date
last_pay_date	01/01/1970. This value means the bill has not been paid yet.
next_pay_date	01/01/3000. This future date ensures there is no due date available yet.
bill_id	Null.
max_num_payments	2147483647. This large number means the recurring payment is only deactivated when the pay date is after the end date

2 The Oracle Self-Service E-Billing ETL load runs and indexes one bill (the doc ID is bill1, in this example) on 03/10/2009. On 04/10/2009, the ETL load runs again and indexes two more bills: bill2 and bill3. The following table gives an example of ETL load producing bills (bill1, bill2, and bill3); it is a combination of the edx_rpt_account_dim,edx_rpt_account_fact, and edx_rpt_statement_fact tables from the OLAP database.

ACCOUNT_NUM	STATEMENT_NUMBER	STATEMENT_LOAD_DATE	AmountDue	DueDate
acct1111	bill1	03/10/2009	100.01	04/15/2009
acct1111	bill2	04/10/2009	50.00	04/25/2009
acct1111	bill3	04/10/2009	100.00	05/15/2009

3 The pmtRecurringPayment job runs on 04/10/2009 23:59:00PM, after the ETL load. The job searches the recurring_payments table to find all recurring payments whose bill_scheduled is Y and status is active. The job finds the example recurring payment and then asks Oracle Self-Service E-Billing to return all bills whose account number is acct1111 and whose STATEMENT_LOAD_DATE is between 04/10/2009 (last_process_time) and 04/10/2009 23:59:00PM (job run time). Two bills, bill2 and bill3, are returned. The pmtRecurringPayment job then finds the bill with latest due date bill3 and bill2 is ignored because only the latest bill is paid.

4 After finding the latest bill from Oracle Self-Service E-Billing, the pmtRecurringPayment job 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, it indicates that it is an old bill and must not be paid. In this case, since there is no last payment, the bill, bill3 is paid.

5 Bill3 is inserted into the payment_bill_summaries table and the recurring_payment table is recalculated as shown in the following table.

Column Name	Value
payer_account_number	acct1111
bill_scheduled	N. This bill has not been paid or scheduled.
status	Active. The next pay date is within the effective period.
last_process_time	04/10/2009 23:59:00PM. This value changes to the job run time.
last_pay_date	01/01/1970, unchanged
next_pay_date	05/14/2009. This value is one day before the due date, 05/15/2009.
bill_id	bill3

6 If the pmtRecurringPayment job runs between 04/11/2009 and 05/10/2009, nothing happens to this recurring payment because synchronization and scheduling does not happen. The table remains unchanged.

7 On 05/11/2009 11:59:00PM, three days before next_pay_date, pmtRecurringPayment runs again. The recurring payment mentioned previously is not synchronized, because its bill_scheduled is N. However, it will be scheduled. pmtRecurringPayment finds all recurring payments whose bill_scheduled is N, status is active and next_pay_date is equal to or before 05/14/2009 (05/11/2009 + 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 05/14/2009. After this, the recurring payment table is changed, as described in the following table.

Column Name	Value
payer_account_number	acct1111
bill_scheduled	Y. The bill has been paid.
status	Active. The next pay date is within the effective period.
last_process_time	04/10/2009 23:59:00PM. This value is unchanged since there was no synchronization.
last_pay_date	05/14/2009. This date changes to the check pay date.
next_pay_date	05/14/2009. This value is unchanged.
bill_id	bill3
payment_id	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 and update or cancel the scheduled payment.

8 On 05/12/2009 23:59:00PM, the pmtRecurringPayment job runs again and finds bills with a doc date between 04/10/2009 11:59:00PM and 05/12/2009 23:59:00PM. No bills exist, and the last process time is updated to 05/12/2009 23:59:00PM. Everything else remains the same.

9 On 05/13/2009, the ETL load occurs again and inserts a new bill, bill4, as shown in the following table. This bill is a combination of the edx_rpt_account_dim, edx_rpt_account_fact, and edx_rpt_statement_fact tables from the OLAP database.

Value in the ACCOUNT_NUM Column	Value in the STATEMENT_NUMBER Column	Value in the STATEMENT_LOAD_DATE Column	Value in the AmountDue Column	Value in the DueDate Column
acct1111	bill1	03/10/2009	100.01	04/15/2009
acct1111	bill2	04/10/2009	50.00	04/25/2009
acct1111	bill3	04/10/2009	100.00	05/15/2009
acct1111	bill4	05/13/2009	80.00	06/15/2009

10 On 05/13/2009 23:59:00PM, the pmtRecurringPayment job runs again, and retrieves bills with dates between 05/12/2009 23:59:00PM and 05/13/2009 23:59:00PM. In this case, the job retrieves bill4 and updates the recurring_payments table, as shown in the following table.

Column Name	Value
payer_account_number	acct1111
bill_scheduled	N. The bill has not been paid.
status	Inactive, because the next pay date is beyond the effective period.
last_process_time	05/15/2009 23:59:00PM. The value changes to the job run time.
last_pay_date	05/14/2009. This value is unchanged.
next_pay_date	06/14/2009. This value is one day before the due date, 06/15/2009.
bill_id	bill4

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

Example of Amount Due and Fixed Pay Date

This example shows how Oracle Self-Service E-Billing processes a recurring payment using the amount due on a fixed pay date.

1 On 04/09/2009, a customer with account number acct1111 creates a recurring payment. The amount is the amount due, the pay date is day 31 of each month, the start date is 04/10/2009, and the recurring payment stops after 10 payments, as shown in the following table.

Column Name	Value
payer_account_number	acct1111
bill_scheduled	Y
status	Active
last_process_time	04/10/2009
last_pay_date	01/01/1970
next_pay_date	4/30/2009. This is the first available pay date after 04/10/2009; there is no 31st of April.
bill_id	Null
end_date	01/01/3000. The end date is so far into the future that the recurring payment will only be deactivated when the number of payments reaches the maximum allowed.
curr_num_payments	0. No payments have been made yet.

The following table shows an example of an ETL load that produces three bills (bill1, bill2, and bill3). This table is a combination of the edx_rpt_account_dim, edx_rpt_account_fact, and edx_rpt_statement_fact tables from the OLAP database.

Value in the ACCOUNT_NUM Column	Value in the STATEMENT_NUMBER Column	Value in the STATEMENT_LOAD_DATE Column	Value in the AmountDue Column	Value in the DueDate Column
acct1111	bill1	03/10/2009	100.01	04/15/2009
acct1111	bill2	04/10/2009	50.00	04/25/2009
acct1111	bill3	04/10/2009	100.00	05/15/2009

Although the pay date is not related to the due date, the value in the DueDate column is indexed because it is used to determine which bill is the latest.

2 The pmtRecurringPayment job runs on 04/10/2009 23:59:00PM, after the ETL load. The load finds bill3 from OLAP database tables edx_rpt_account_dim, edx_rpt_account_fact, and edx_rpt_statement_fact and inserted in the payment_bill_summaries table.

The recurring_payments table is recalculated as shown in the following table.

Column Name	Value
payer_account_number	acct1111
bill_scheduled	N. This bill has not been paid.

Column Name	Value
status	Active. The value in the curr_num_payments column is less than the value in the max_num_payments column.
last_process_time	04/10/2009 23:59:00PM. This column changed to the job run time.
last_pay_date	01/01/1970. This value is unchanged.
next_pay_date	04/30/2009. There is no 31st of April.
bill_id	bill3
curr_num_payments	0

3 On 04/27/2001, three days before the next pay date, the pmtRecurringPayment job 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 the pay date is 04/30/2001. Changes to the recurring payment table are shown here.

Column Name	Value
payer_account_number	acct1111
bill_scheduled	Y. The bill has been paid.
status	Active. The value in the curr_num_payments column is less than the value in the max_num_payments column.
last_process_time	04/10/2009 23:59:00PM. This value is unchanged since there has been no synchronization.
payer_account_number	acct1111
bill_scheduled	Y. The bill has been paid.
status	Active. The value in the curr_num_payments column is less than the value in the max_num_payments column.
last_process_time	04/10/2009 23:59:00PM. This value is unchanged since there has been no synchronization.

4 Steps 2, 3, and 4 repeat until the value of the curr_num_payments column reaches 10. At Step 4 of the tenth payment, the status changes to inactive.

If no bills arrive for a month, then the next pay date automatically moves to the next month. For example, if there is no bill for April, then the next pay date automatically moves from 04/30/2009 to 05/31/2009 when the current job run time is May First.

Example of Fixed Amount and Before Due Date

This example shows how Oracle Self-Service E-Billing processes a recurring payment using a fixed amount before the due date.

1 On 04/09/2009, a customer with account number as acct1111 creates a recurring payment. The amount is $50, the pay date is one day before the due date, the start date is 04/10/2009 and the recurring payment stops after ten payments, as shown in the following table.

Column Name	Value
payer_account_number	acct1111
bill_scheduled	Y
status	Active
last_process_time	04/10/2009
last_pay_date	01/01/1970
next_pay_date	01/01/3000
bill_id	Null
end_date	01/01/3000. The end date is so far into the future that the recurring payment will only be deactivated when the number of payments reaches the maximum allowed.
curr_num_payments	0. No payments have been made yet.

The following table shows the Index table entries, which are a combination of the edx_rpt_account_dim, edx_rpt_account_fact, and edx_rpt_statement_fact tables from the OLAP database.

Value in the ACCOUNT_NUM Column	Value in the STATEMENT_NUMBER Column	Value in the STATEMENT_LOAD_DATE Column	Value in the AmountDue Column
acct1111	bill1	03/10/2009	04/15/2009
acct1111	bill2	04/10/2009	04/25/2009
acct1111	bill3	04/10/2009	05/15/2009

Amount due is not required for this case.

2 The pmtRecurringPayment job runs on 04/10/2009 23:59:00PM, after the ETL load. In this case, bill3 is found in the index table and inserted into the payment_bill_summaries table. The values in the recurring payments table is recalculated as listed in the following table.

Column Name	Value
payer_account_number	acct1111
bill_scheduled	N. The bill has not been paid.
status	Active. The value in the curr_num_payments is less than the value in the max_num_payments column.

Column Name	Value
last_process_time	04/10/2009 23:59:00PM. This value changes to the job run time.
last_pay_date	01/01/1970. The value is unchanged.
next_pay_date	05/14/2009. This value is one day before the due date, 05/15/2009.
bill_id	bill3
curr_num_payments	0

3 On 05/11/2009, three days before the next pay date, the pmtRecurringPayment job 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 05/14/2009. The recurring_payments table changes as shown in the following table.

Column Name	Value
payer_account_number	acct1111
bill_scheduled	Y. The bill has been paid.
status	Active. The value in the next_pay_date column is not after the value in the end_date column.
last_process_time	04/10/2009 23:59:00PM. This value is unchanged since there was no synchronization.
last_pay_date	05/11/2009. This value changes to the value in the next_pay_date column.
next_pay_date	05/11/2009. This value is unchanged, the next bill is not known.
bill_id	bill3
payment_id	The new payment ID inserted into the check_payments or creditcard_payments table.
curr_num_payments	1

4 Steps 2, 3, and 4 repeat until next_pay_date is after end_date, when status changes to inactive.

Example of Fixed Amount and Fixed Pay Date

This example shows how Oracle Self-Service E-Billing processes a recurring payment using a fixed amount and a fixed pay date.

1 On 04/09/2009, 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 04/10/2009 and ends at 06/10/2009. The columns in the recurring_payments table are updated, as shown in the following table.

Column Name	Value
payer_account_number	acct1111
bill_scheduled	N
status	active
last_process_time	04/10/2009
last_pay_date	01/01/1970
next_pay_date	05/01/2009
bill_id	Null
end_date	06/10/2009
curr_num_payments	0. No payment has been made yet.

2 On 04/28/2009, three days before the next pay date, the pmtRecurringPayment job runs again. There is no synchronization because the value in the bill_scheduled column is always N, however 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 05/01/2009.

The columns in the recurring_payments table are updated, as shown in the following table.

Column Name	Value
payer_account_number	acct1111
bill_scheduled	N. This bill has been paid.
status	Active. The value in the next_pay_date column is not after the value in the end_date column.
last_process_time	04/10/2009. This value is unchanged since there was no synchronization.
last_pay_date	05/01/2009. This value changes to the next pay date.
next_pay_date	06/01/2009. This value changes to the next available pay date.
bill_id	Null
payment_id	The payment ID inserted into the check_payments or creditcard_payments table.
curr_num_payments	1

3 This step repeats until the next pay date is after the end date, when the status changes to inactive.

10 Reviewing Production Activity

This chapter describes the tasks that you can perform to review production activity. It includes the following topics:

- About Job Reports on page 117
- About Message Log Files on page 118
- Monitoring Service Status on page 119
- About Administrator Activity Auditing on page 120

About Job Reports

You can use the Command Center to create and view job reports showing history and statistical information about each instance when a particular job ran, in one or more applications, over a particular time period. You can generate a report for a specific job or for all jobs. You can also generate a report about jobs that ran against one or more specific data files.

Table 54 describes the columns on the Job Report for Email Notification.

Table 54. Columns on the Email Notification Job Report

Column	Description
Job Name	Name of the job
Application	Name of the application
Start Time	Time the job started
End Time	Time the job ended
Time Elapsed	Total running time
Total Email Count	Total number of emails generated
Total Emails Sent	Total number of emails sent successfully
Total Emails Unsent	Total number of emails not sent; messages can be unsent if the job is still processing or due to an error or exception
Total Emails Unresolved	Total number of emails with unresolved status, for example, due to missing email address.
Total Emails Failed	Total number of emails with failed status; an email fails if all retries are unsuccessful
Address Error	Number of times an error occurred with the end rolling email address
Server Error	Number of times a mail server error occurred

Table 54. Columns on the Email Notification Job Report

Column	Description
Job Status	The current status of the job
Data File	Name of the data file the job used
Percent Complete	The percentage of the job that is currently complete.

To view a job report

1. Click Reporting on the Command Center menu. The Reporting screen displays.
2. Click the Job Reports icon to display the Job Reports screen.
3. Select one or more applications from the menu and then click Submit Query.
4. To generate a report for a particular job, enter the job name; to get a report for all jobs during a given time frame, leave the job name field blank.

   CAUTION: Not entering a job name can return a very large data set that can take a long time to load. For heavily trafficked implementations, setting a small date and time range is recommended.
5. Enter a start and end date, and a start and end time range. Click Popup Calendar to select dates quickly.
6. To generate a report for one or more data files instead of by job or date range, select the files.

   TIP: To select or deselect a data file, press Ctrl+Left mouse click.

   CAUTION: If you select a data file, all other fields for this query are ignored.
7. Click Submit Query to start the search.

   A Job Report for the selected search criteria displays.
8. Click Search Again to perform another search for job data.

About Message Log Files

Oracle Self-Service E-Billing maintains log files of all activities that occur and messages that are generated during production. Review these log files on a regular, ongoing basis to monitor jobs in your production environment.

You can create and view a report showing any of the following types of log messages generated over a select time period:

- **Error.** Logs errors.
- **Information.** Logs information about production activity.
- **Warning.** Logs warning messages.
- **Debug.** Troubleshooting log.

Log reports display the information shown in Table 55.

Table 55. Columns on the Log Report

Column	Description
Timestamp	The date and time the message was created in the log.
SourceHost	Name of the server that generated the error message or where the production activity occurred.
Message ID	A code identifying the task where the error occurred and the level of error.
Message	The message text.

To view production log messages

1. Click Reporting on the Command Center menu.
2. Click the View Logs tab to display the View Logs screen.
3. Select the type of message log to view.
4. Enter a start date and end date range to search. Click Popup Calendar to select dates quickly.
5. Enter a start and end time to search and then click Submit Query.

   Oracle Self-Service E-Billing displays the error messages for the selected log type, and date and time range.
6. To view different log information, click Reselect Log View, and select new criteria.

Monitoring Service Status

You can check on the status of services using the Command Center.

To view the status of services

1. Click Service Status on the Command Center menu.

   The Service Status page appears, showing whether all services are running or which, if any, are missing.
2. If services are missing, complete the following steps:
   a. Close Command Center.
   b. Shut down and restart the application server.
   c. Display the Service Status again to verify that the problem has been corrected. If services are still missing, see *Installation Guide for Oracle Self-Service E-Billing*.

About Administrator Activity Auditing

Oracle Self-Service E-Billing maintains an audit record of activities performed by each system administrator in the ADMIN_ACTIVITY table.

Audited Command Center activities include:

- Creating, deleting, and editing job configurations.
- Logging in and logging out of the Command Center.

To purge activity history from the ADMIN_ACTIVITY table, see "Purging Administrator Activity" on page 140.

11 Administering the Database

This chapter describes the tasks associated with administering the Oracle Self-Service E-Billing database. It includes the following topics:

- Managing the Payment Module Database on page 121
- Setting Enrollment Properties on page 126
- Deleting an Application on page 126
- About Purging Data on page 127
- Purging Payment Account and Transactional Data on page 128
- Purging Hierarchies and Hierarchy Assignments on page 134
- Purging User Data on page 136
- Purging Statement and Invoice Fact Data on page 138
- Purging Validation Codes on page 139
- Purging Administrator Activity on page 140
- Purging Locked Administrators in Command Center on page 140
- Purging Data in Batch on page 141
- Auditing Purged Data on page 142
- Process of Purging Sample Data on page 142
- Running the Master Key Update on page 143

Managing the Payment Module Database

Running an application in a live production environment can generate a large volume of historical data in an application's database. You are responsible for monitoring and maintaining your own database server.

It is recommended that you monitor your database server on a weekly or other regular basis to perform the following tasks:

- **Check database utilization.** To periodically eliminate older application data and free up space on your database server, create, configure, and run the Purge Logs job.
- **Check memory utilization.** Check memory utilization of the SQL server swap (paging) file. When the peak number of Commit Charge reaches 10% of the limit, then it is advisable to increase the size of paging file, or install more RAM.
- **Back up your database.** Create and implement a regular database backup plan.

There are specific tasks required in Oracle Self-Service E-Billing to administer the database of the Payment module.

Preventing Multiple Payments

By default, the Payment module allows a bill to be paid more than one time. To make sure that a bill can only be paid one time, you need to add a unique key constraint on the bill_id field of the check_payments table. Run the set_unique_bill_id.sql script, found in the *EDX_HOME/db/DB_NAME* directory (or in the *EDX_HOME/db/DB_NAME/migration/to42* directory), to set the unique constraint. In the path, *EDX_HOME* is the directory where you installed Oracle Self-Service E-Billing. The bill_id in the Payment module is the same as the doc ID in Oracle Self-Service E-Billing.

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

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

The unique key constraint only informs a customer that the bill has been paid when he or she tries to pay a bill that has already been paid. If you want to provide additional features, for example, disabling the payment option when the bill has already been paid, you must query the database to get that information. Use caution when adding extra functions because performing additional database queries can deteriorate Payment module performance. Make sure to create the proper index if you plan to create a new query.

UI Actions and Database Changes

Table 56 lists user actions and describes their impact on the Payment module database.

Table 56. UI Actions and Database Changes

UI Action	Payment Module Database
Create payment setting (Command Center)	Payment setting information is saved in the payment_profile table. The param_name user_account_accessor must point to the right IUserAccountAccessor implementation and payment_account_accessor must point to the right IPaymentAccountAccessor implementation.
User enrolls	User information is inserted into CDA and payment accounts are inserted into the payment_accounts table.
Run pmtSubmitEnroll	Finds all payment accounts whose account_status is pnd_active, txn_date is null, and sends to the ACH payment gateway. After that, it sets txn_date to the current date (yyyyMMddHHmm).

Table 56. UI Actions and Database Changes

UI Action	Payment Module Database
Run pmtConfirmEnroll	Changes account_status to bad_active if returned or to active if there is no return after three days. Updates notify_status to N.
User logs in	The user's ID and password is checked against the user_id and hash in the enrollment table.
User makes a check payment	The payment is saved into the check_payments table. The status of the check is Scheduled (6).
User clicks on Future Payments	Displays a list of scheduled payments for this user. The user can use the list to cancel or update scheduled payments.
User cancels or updates a check	When a user cancels a check, the status of that check is set to Canceled (9). The check is not deleted from the database. When a user updates a check, the same entry in check_payments will be updated. A check can only be cancelled or updated when the status of that check is Scheduled (6).
Run pmtCheckSubmit	Finds all checks due by tomorrow (or before), and sends them to the ACH payment gateway. The status of the check changed to processed (7). The txn_number field now holds the trace number of the check, and the reminded field is set to N. A batch report file is written to the payment_log table, whose type is summary. You can view this report from the Command Center.
User clicks on Payment History	Displays processed check payments as well as paid, returned, failed and cancelled checks.

Payment Table Sizing

The size to which the Payment tables grow depends on the number of enrolled users.

Table 57 describes the Payment tables and other Oracle Self-Service E-Billing tables that are related to enrollment. Differences are noted for Microsoft SQL Server. The numbers in the table assume that there are 100,000 registered users.

Table 57. Payment Table Sizing Information

Payment Table	Projected Row Count	Notes
check_payments	1.2 million based on 100,000 users kept for one year	Customer dependent. Assuming one user makes one check payment each month, and check history is kept in the database for one year, then the total number of rows is approximately 12 times the number of users.
check_payments_ history	Approximately 3 times the size of check_ payments table	Tracks the state of a check payment. Usually a check passes through three states before it is cleared or returned.
check_payments_ status	10	This table is a reference to the meanings of check payment status. It is not used by the Payment module.
credit_card_ payments	1.2 million based on 100,000 users a year	Customer dependent. Assuming one user makes one credit card payment a month, and credit card payments are kept in the database for one year, then the total number of rows will be approximately 12 times the number of users.
payment_accounts	200,000	The estimated row size is approximately 1,400 for each user, or 280MB for 100,000 users.
payment_bill_ summaries	Approximately 33,000 times 12, which equals 400,000, assuming one-third of the register with recurring payment	Saves bill information related to recurring payments.
payment_invoices	12 million based on each payment averaging 10 invoices	Typically used for business customers. Some billers might choose not to use this feature.
payment_log	Less than 10,000 a year	Approximately 20 rows will be inserted when a pmtCheckSubmit batch job is run.
payment_profile	Less than 100	There are approximately 20 rows for each DDN and payment type.

Table 57. Payment Table Sizing Information

Payment Table	Projected Row Count	Notes
payment_reminders	Less than 100,000 based on 100,000 users	Customer dependent. Each customer can set up one reminder (each reminder creates one row in the table), but not all customers will use reminders.
recurring_payments	Approximately 33,000, assuming one third of the register with recurring payment	If recurring payment is turned off, then this table will be empty.

Payment Table Maintenance

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

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

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

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

Backup and Recovery

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

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

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

Backing up Tables

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

Backing up Stored Procedures

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

Setting Enrollment Properties

You can modify the following properties in effect when system administrators enroll and log in to the Command Center:

- Administrator minimum length
- Password minimum length
- Number of times before re-entry of password accepted (number of times you must change your password before you can you can reuse a password)
- Account will become locked after X amount of invalid attempts
- Password expired days (the number of days before a password expires)

To set enrollment properties

1. On the Command Center Main Console, click Settings.
2. Click the Enrollment tab.
3. On the Enrollment Configuration page, specify the parameters you want to modify, and click Update.

Deleting an Application

If you have an old or unusable application, you can use the Command Center to delete it.

If you are testing in a quality assurance or development environment, you might want to remove an unneeded application.

You would not normally need to delete an application from a production environment, unless you incorrectly configured an application or set up the training application by mistake.

Note that deleting an application only removes it from use; it does not delete any associated data in the database for jobs that were run. To delete data from the database, run the Purge Logs job.

To delete an old application

1. Delete all jobs associated with an application. You cannot delete an application until you have deleted all the related jobs.
2. At the Main Console, click the name of the application you want to delete. The Edit Application screen displays.
3. If any jobs still exist, click the box in the Delete column for each job, then click Delete Marked Jobs. Click OK when asked if you are sure you want to delete the marked jobs.

4 Click Remove Application (which only appears when no jobs are listed).

5 When asked if you are sure you want to delete the application, click OK.

About Purging Data

Oracle Self-Service E-Billing provides a script for purging data from the databases (found in the *EDX_HOME*/bin directory (or the *EDX_HOME*\bin directory on Windows):

- **UNIX.** purge_data.sh
- **Windows**. purge_data.bat

You run the purge script from the shell command using the specific input for the type of data you want to remove, as described in the following procedures:

- "Purging Payment Account and Transactional Data" on page 128
- "Purging Hierarchies and Hierarchy Assignments" on page 134
- "Purging User Data" on page 136
- "Purging Statement and Invoice Fact Data" on page 138
- "Purging Validation Codes" on page 139
- "Purging Administrator Activity" on page 140
- "Purging Locked Administrators in Command Center" on page 140
- "Process of Purging Sample Data" on page 142

You can also purge all related transactional data in batch. For information about purging data in batch mode, see "Purging Data in Batch" on page 141.

Each time you run the purge script, Oracle Self-Service E-Billing records audit information about the purge. For information about viewing purge audit information, see "Auditing Purged Data" on page 142.

NOTE: Avoid running the purge script during busy traffic periods. Always consult your DBA before purging any data from the database.

Viewing Help for Running the Purge Script

Complete the following task to view help with general usage information for the purge script.

To get help for using the Oracle Self-Service E-Billing purge script

- Run the following shell script (found in the *EDX_HOME*/bin directory or the *EDX_HOME*\bin directory on Windows), where *EDX_HOME* is the directory where you installed Oracle Self-Service E-Billing:

  UNIX:

```
./purge_data.sh -help
```

Windows:

```
purge_data.bat -help
```

Purging Payment Account and Transactional Data

The payment purge function lets you remove the following payment account and transactional information from the Oracle Self-Service E-Billing database:

- Pending payments
- Payment accounts
- Payment history
- Recurring payments
- Payment invoices
- Payment notifications
- Payment audits

Purging Pending Payments

You can purge pending payment accounts for:

- All pending payments
- Pending payments for a specific company ID
- Pending payments for a specific user ID
- Pending payments for a specific period (with pay_date between specific to and from dates)
- Pending payments for all Consumer Edition (B2C) users

Purge pending payments removes payments scheduled in advance from the following OLTP tables:

- EDX_PMT_CHK_ACCT_ONETIME
- CHECK_PAYMENTS
- CREDITCARD_PAYMENTS

Complete the following task to purge pending payments.

To purge pending payments

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

```
./purge_data.sh -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
```

Windows:

```
purge_data.bat -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
```

where:

- *OLTP schema username* is the name of the OLTP schema user.
- *OLTP schema password* is the password of the OLTP schema user.
- *OLTP TNS name* is the TNS name for the OLTP instance.
- *Input* is one of the following:
  - All pending payments: `-payment pending -company -all`
  - Pending payments for a specific company ID: `-payment pending -company "Company ID"`
  - Pending payments for a specific user ID: `-payment pending -user User ID`
  - Pending payments for a specific period: `-payment pending -date MM-DD-YYYY MM-DD-YYYY`
  - Pending payments for all B2C users: `-payment pending -b2c`

Purging Payment Accounts

Purging payment accounts removes the payment accounts where the delete_date column is not null and also not referenced by any payment transaction. The delete_date_column is not null when a user has deleted the account using the interface.

You can purge payment accounts for:

- All payment accounts
- A specific company ID
- A specific user ID
- A specific time period
- All Consumer Edition (B2C) users

Purging payment accounts removes data from the PAYMENT_ACCOUNTS OLTP table.

Complete the following task to purge payment accounts.

To purge payment accounts

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

```
./purge_data.sh -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
```

Windows:

```
purge_data.bat -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
```

where *Input* is one of the following:

- All payment accounts: `-payment account -company -all`
- Accounts for a specific company ID: `-payment account -company "Company ID"`
- Accounts for a specific user ID: `-payment account -user User ID`
- Accounts for a specific period: `-payment account -date MM-DD-YYYY MM-DD-YYYY`
- Accounts for all B2C users: `-payment account -b2c`

Purging Payment History

You can purge payment history for:

- All payment history
- A specific company ID
- A specific company ID and period
- A specific user ID
- A specific user ID and period
- A specific period
- All Consumer Edition (B2C) users
- All Consumer Edition (B2C) users for a specific period

Purging payment history removes transactional history data from the following OLTP tables:

- EDX_PMT_CHK_ACCT_ONETIME
- CHECK_PAYMENTS
- CREDITCARD_PAYMENTS

Check payment history and credit card payment history purge by pay date. The Recurring payment history table purges by the timestamp.

Complete the following task to purge payment history.

This task is a step in "Process of Purging Sample Data" on page 142.

To purge payment history

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

  ```
  ./purge_data.sh -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
  ```

  Windows:

  ```
  purge_data.bat -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
  ```

  where *Input* is one of the following:

  - All payment history: `-payment history -company -all`
  - Payment history for a specific company ID: `-payment history -company "Company ID"`
  - Payment history for a specific company ID and period: `-payment history -company "Company ID" -date MM-DD-YYYY MM-DD-YYYY`
  - Payment history for a specific user ID: `-payment history -user User ID`
  - Payment history for a specific user ID and period: `-payment history -user User ID -date MM-DD-YYYY MM-DD-YYYY`
  - Payment history for a specific period: `-payment history -date MM-DD-YYYY MM-DD-YYYY`
  - Payment history for all B2C users: `-payment history -b2c`
  - Payment history for all B2C users for a specific period: `-payment history -b2c -date MM-DD-YYYY MM-DD-YYYY`

Purging Recurring Payments

You can purge recurring payments for:

- All recurring payments
- A specific company ID
- A specific user ID
- All Consumer Edition (B2C) users

Purging recurring payments removes data from the following OLTP tables:

- PAYMENT_BILL_SUMMARIES
- RECURRING_PAYMENTS

Complete the following task to purge recurring payments.

To purge recurring payments

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

  ```
  ./purge_data.sh -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
  ```

  Windows:

  ```
  purge_data.bat -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
  ```

  where *Input* is one of the following:

  - All recurring payments: `-payment recurring -company -all`
  - Recurring payments by specific company ID: `-payment recurring -company "Company ID"`
  - Recurring payments by specific user ID: `-payment recurring -user User ID`
  - Recurring payments for all B2C users: `-payment recurring -b2c`

Purging Payment Invoices

You can purge payment invoices for:

- All payment invoices
- A specific company ID
- A specific user ID
- A specific period
- All Consumer Edition (B2C) users

Purging payment invoices removes data from the PAYMENT_INVOICES table.

Complete the following task to purge payment invoices

To purge payment invoices

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

  ```
  ./purge_data.sh -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
  ```

  Windows:

  ```
  purge_data.bat -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
  ```

  where *Input* is one of the following:

- All payment invoices: `-payment invoice -company -all`
- Payment invoices for a specific company ID: `-payment invoice -company "`*`Company ID`*`"`
- Payment invoices for a specific user ID: `-payment invoice -user` *`User ID`*
- Payment invoices for a specific period: `-payment invoice -date` *`MM-DD-YYYY MM-DD-YYYY`*
- Payment invoices for all B2C users: `-payment invoice -b2c`

Purging Payment Notifications

You can purge payment notifications for:

- All payment notifications
- A specific company ID
- A specific user ID
- A specific period
- All Consumer Edition (B2C) users

Purging payment notifications removes data from the PAYMENT_DUE_NOTIFICATION_ACCTS table.

Purging payment notifications for a specific time period purges the PAYMENT_DUE_NOTIFICATION_ACCTS table by last_processed.

To purge payment notifications

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

  ```
  ./purge_data.sh -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
  ```

  Windows:

  ```
  purge_data.bat -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
  ```

  where *Input* is one of the following:

  - All payment notifications: `-payment notification -company -all`
  - Payment notifications for a specific company ID: `-payment notification -company "`*`Company ID`*`"`
  - Payment notifications for a specific user ID: `-payment notification -user "`*`User ID`*`"`
  - Payment notifications for a specific period: `-payment notification -date` *`MM-DD-YYYY MM-DD-YYYY`*
  - Payment notifications for all B2C users: `-payment notification -b2c`

Purging Hierarchies and Hierarchy Assignments

This topic describes how to purge the following hierarchy data from the OLAP and OLTP databases:

- Billing hierarchies
- Business hierarchies
- B2B user hierarchy assignments
- Billing accounts and services (from the OLTP database only)

NOTE: Removing information from the OLTP hierarchy table displays a message on screen. Information removed from OLAP hierarchy tables does not display a message because dbms_output does not work across the database link. Check the EDX_PURGE_LOG table for OLAP information.

Purging Billing Hierarchies

You can purge billing hierarchy information for a specific company ID only.

Purging billing hierarchies removes data from the following tables:

- OLTP:
  - EDX_HIER_NODE_USER
  - EDX_HIER_NODE_PERIOD
  - EDX_HIER_NODE_ATTRIBUTE
  - EDX_HIER_HNODE
  - EDX_HIER_HIERARCHY
- OLAP:
  - EDX_RPT_CC_CHARGE_WSPACE
  - EDX_RPT_ACCOUNT_WSPACE
  - EDX_RPT_HIERARCHY_NODE_PERIOD
  - EDX_RPT_HIERARCHY_XREF_DIM
  - EDX_RPT_HIERARCHY_TREE_DIM

To purge the billing hierarchy for specific company ID

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

  ```
  ./purge_data.sh -connstr OLTP schema username/OLTP schema password@OLTP TNS name
  -hierarchy billing -company "Company ID"
  ```

  Windows:

```
purge_data.bat -connstr OLTP schema username/OLTP schema password@OLTP TNS name
-hierarchy billing -company "Company ID"
```

Purging Business Hierarchies

You can purge the business hierarchy for a specific company ID only.

Purging business hierarchies purges data from the following tables:

- OLTP
  - EDX_HIER_NODE_USER
  - EDX_HIER_NODE_PERIOD
  - EDX_HIER_NODE_ATTRIBUTE
  - EDX_HIER_HNODE
  - EDX_HIER_HIERARCHY
- OLAP
  - EDX_RPT_CC_CHARGE_WSPACE
  - EDX_RPT_ACCOUNT_WSPACE
  - EDX_RPT_HIERARCHY_NODE_PERIOD
  - EDX_RPT_HIERARCHY_XREF_DIM
  - EDX_RPT_HIERARCHY_TREE_DIM

Follow these instructions to purge the business hierarchy for a particular company.

To purge the business hierarchy for specific company ID

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

  ```
  ./purge_data.sh -connstr OLTP schema username/OLTP schema password@OLTP TNS name
  -hierarchy business -company "Company ID"
  ```

  Windows:

  ```
  purge_data.bat -connstr OLTP schema username/OLTP schema password@OLTP TNS name
  -hierarchy business -company "Company ID"
  ```

Purge B2B User Hierarchy Assignments

You can purge B2B user hierarchy assignment information for a specific user ID only.

Purging B2B user hierarchy assignments removes data from the EDX_HIER_NODE_USER OLTP table.

To purge the B2B user hierarchy assignment for specific user ID

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

  ```
  ./purge_data.sh -connstr OLTP schema username/OLTP schema password@OLTP TNS name
  -hierarchy assign -user User ID
  ```

  Windows:

  ```
  purge_data.bat -connstr OLTP schema username/OLTP schema password@OLTP TNS name
  -hierarchy assign -user User ID
  ```

Purging Billing Accounts and Services

You can purge billing accounts and services data for a specific company ID only.

Purging billing accounts and services removes data from the following OLTP tables:

- EDX_OMF_SERVICECHARGE
- EDX_OMF_SRVICE_PLAN
- USER_SERVICE_AGREEMENT
- EDX_OMF_SERVICEAGREEMENT
- EDX_BSL_ACCT_ATTRIBS
- EDX_BSL_AMF_BACCOUNT

To purge billing accounts and service data for a specific company ID

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

  ```
  ./purge_data.sh -connstr OLTP schema username/OLTP schema password@OLTP TNS name
  -hierarchy service -company "Company ID"
  ```

  Windows:

  ```
  purge_data.bat -connstr OLTP schema username/OLTP schema password@OLTP TNS name
  -hierarchy service -company "Company ID"
  ```

Purging User Data

You can purge user data for:

- A specific company ID
- A specific user ID
- All Consumer Edition (B2C) users

- Inactive users for a specific company ID
- Inactive Consumer Edition (B2C) users

NOTE: When you purge a user, all scheduled recurring payments and one-time payments for this user cancel.

Purging user data removes information from the following OLTP tables:

- EDX_BSL_SEC_PROF_ATTRIBS
- EDX_BSL_SEC_PROF_ROLES_LINK
- EDX_BSL_USER_PROF_ATTRIBS
- EDX_UMF_SEC_ATTEMPTS
- EDX_UMF_SEC_QUESTION
- USER_SERVICE_AGREEMENT
- EDX_UMF_USER_ACCT_LINK
- ADDRESS_BOOK_PERSONAL
- EDX_UMF_BULK_ENROLL_LOG
- EDX_BSL_UMF_USER
- EDX_UMF_SEC_PWD_HISTORY
- EDX_BSL_AUTH_SECPROFILE

Purging user data also removes information from the EDX_UMF_USER_ACCT_LINK OLAP table.

To purge user data

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

  ```
  ./purge_data.sh -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
  ```

  Windows:

  ```
  purge_data.bat -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
  ```

  where *Input* is one of the following:

  - Purge user data for a specific company ID: `-profile -company "Company ID"`
  - Purge user data for a specific user ID: `-profile -user User ID`
  - Purge all B2C user data: `-profile -b2c`
  - Purge inactive user data for a specific company ID: `-profile inactive -company "Company ID"`
  - Purge inactive B2C user data: `-profile inactive -b2c`

Purging Statement and Invoice Fact Data

You can purge statement and invoice fact data for:

- A specific company ID
- A specific company ID and period
- A specific account number
- A specific account number and period
- All Consumer Edition (B2C) users
- All Consumer Edition (B2C) users for a specific period

Purging statements and invoices removes data from the following OLAP tables:

- EDX_RPT_STATEMENT_FACT
- EDX_RPT_STATEMENT_PAYMENT_FACT
- EDX_RPT_STATEMENT_ADJUST_FACT
- EDX_RPT_ACCOUNT_FACT
- EDX_RPT_ACCOUNT_CHARGE_FACT
- EDX_RPT_SERVICE_FACT
- EDX_RPT_SERVICE_CHARGE_FACT
- EDX_RPT_SERVICE_PRODUCT_FACT
- EDX_RPT_SERVICE_USAGE_FACT
- EDX_RPT_SERVICE_TARIFF_FACT
- EDX_RPT_SERVICE_DETAIL_FACT
- EDX_RPT_TOP_100_EXPENSIVE_CALL
- EDX_RPT_ACCOUNT_DEST_SUMM
- EDX_RPT_TOP_100_CALLED_NUM
- EDX_RPT_TOP_100_LONG_CALL

When purging by specific account number, the following tables purge only if the statement has only one account:

- EDX_RPT_STATEMENT_FACT
- EDX_RPT_STATEMENT_PAYMENT_FACT
- EDX_RPT_STATEMENT_ADJUST_FACT

To purge statement and invoice fact data

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

  ```
  ./purge_data.sh -connstr OLAP schema username/OLAP schema password@OLAP TNS name Input
  ```

  Windows:

  ```
  purge_data.bat -connstr OLAP schema username/OLAP schema password@OLAP TNS name Input
  ```

  where *Input* is one of the following:

  - Fact data by specific company ID: `-statement fact -company "Company ID"`
  - Fact data by specific company ID and period: `-statement fact -company "Company ID" -period Start period Mon-YYYY End period Mon-YYYY`
  - Fact data by specific account number: `-statement fact -account Account number`
  - Fact data by specific account number and period: `-statement fact -account Account number -period Start period Mon-YYYY End period Mon-YYYY`
  - All B2C fact data: `-statement fact -b2c`
  - All B2C Fact data for a specific period: `-statement fact -b2c -period Start period Mon-YYYY End period Mon-YYYY`

Purging Validation Codes

You can purge the validation codes automatically generated by Oracle Self-Service E-Billing when a new users enroll.

You can specify the number of days before which the code must have been generated. Purging validation codes removes that information from the EDX_UMF_SEC_VALIDATIONCODE table.

To purge validation codes

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

  ```
  ./purge_data.sh -connstr OLTP schema username/OLTP schema password@OLTP TNS name -validationcode -keepdays Number of days
  ```

  Windows:

  ```
  purge_data.bat -connstr OLTP schema username/OLTP schema password@OLTP TNS name -validationcode -keepdays Number of days
  ```

Purging Administrator Activity

You can purge the audit history of the administrator activity from the ADMIN_ACTIVITY table.

You can purge the administrator activity for:

- A specific administrator
- All administrators

To purge administrator activities

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

  ```
  ./purge_data.sh -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
  ```

  Windows:

  ```
  purge_data.bat -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
  ```

  where *Input* is one of the following:

  - Purge administrator activity for a specific user ID: `-adminactivity -user User ID`
  - Purge administrator activity for all user IDs: `-adminactivity -user -all`

Purging Locked Administrators in Command Center

You can unlock Command Center administrators by purging the lock data in the database.

You can unlock (purge data for):

- A specific administrator
- All administrators

Purging locked administrators removes data from the CDA_NODES, CDA_ATTRIBUTES, and USR_PASSWORD_ENTRIES table.

To unlock (purge data for) administrators

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

```
./purge_data.sh -connstr OLTP schema username/OLTP schema password@OLTP TNS name
Input
```

Windows:

```
purge_data.bat -connstr OLTP schema username/OLTP schema password@OLTP TNS name
Input
```

where *Input* is one of the following:

- Purge lock data for a specific user ID: `-lockedadmin -user` *User ID*
- Purge lock data for all user IDs: `-lockedadmin -user -all`

Purging Data in Batch

You can purge all related transactional data in batch for:

- A specific company ID
- A specific user ID
- All Consumer Edition (B2C) users

NOTE: Payment history data is excluded, as some clients have restrictions on deleting any payment for x number of months. The scheduled payments will be cancelled as users are removed from database.

Purging data for a specific company ID removes the following information:

- Payment accounts
- Payment invoice
- Payment notification
- Hierarchies
- Statements and invoice
- Accounts and services
- Scheduled payments for the users in this company (cancelled)
- User profiles
- Company itself
- Recurring payment

Purging data for a specific user ID or for all B2C users removes the following information:

- Payment accounts
- Payment invoice
- Payment notification
- Hierarchy user assignment

- Cancel scheduled payments for this user
- User profile
- Recurring payment

Complete the following task to purge data in batch.

This task is a step in "Process of Purging Sample Data" on page 142.

To purge data in batch

- Run the following command from the *EDX_HOME*/bin directory (the *EDX_HOME*\bin directory on Windows):

  UNIX:

  ```
  ./purge_data.sh -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
  ```

  Windows:

  ```
  purge_data.bat -connstr OLTP schema username/OLTP schema password@OLTP TNS name Input
  ```

  where *Input* is one of the following:

  - Purge data in batch for a specific company ID: `-company "Company ID"`
  - Purge data in batch for a specific user ID: `-user User ID`
  - Purge data in batch for all B2C users: `-user -b2c`

Process of Purging Sample Data

You must purge sample data from the Oracle Self-Service E-Billing database after performing any necessary installation test runs. Use the batch purge to remove all sample B2B companies and related transactional data one at a time. Also use the batch purge command to remove all sample B2C data.

To purge sample data, perform the following tasks:

1. "Purging Data in Batch" on page 141.
2. "Purging Payment History" on page 130.

Auditing Purged Data

When you purge data using the purge script, a list of messages appears in the format:

Number of entries removed from table *name*.

Each time you run the purge script, Oracle Self-Service E-Billing records audit information about the purge in the EDX_PURGE_LOG table in the OLTP instance.

The data recorded in the EDX_PURGE_LOG table includes the following information:

- When purges occurred
- Which database tables were purged
- How much data in the table was removed

NOTE: Sometimes you cannot see the full purged information on the console screen but data did purge in the tables, that's because the output function does not work through database link. Please refer to EDX_PURGE_LOG for the full removing information.

Running the Master Key Update

You must run the script to update the master key once a year as required to comply with the Payment Card Industry Data Security Standard (PCI DSS). This script updates the master key as well as related subkeys and validation code in the Oracle Self-Service E-Billing database.

You must also update the master key after installing Oracle Self-Service E-Billing and setting up the OLTP and OLAP databases, where the master key is used.

To run the master key update

1. Repackage the GNU Lesser Public License on your Oracle Self-Service E-Billing files. For instructions on the process of repackaging, see *Installation Guide for Oracle Self-Service E-Billing*.
2. Shut down the application server to ensure that all data is in a consistent status.
3. It is strongly recommended to back up the Oracle Self-Service E-Billing OLTP database.
4. Back up the master keystore folder, *EDX_HOME*\keystore.
5. Back up the persistence.xma.xml file, found in the *EDX_HOME*\xma\config\modules directory. You must modify the file, then restore it after you run the master key update script. Make a backup of the persistence.xma.xml file to use for the restore.

   Modify the persistence.xma.xml file for the myDataSource and TransactionManager beans, required to support OLTP database operation when the application server is shut down. It is suggested to use c3p0 to connect to myDataSource, and Spring Framework transaction support for the TransactionManager bean.

   Remove or comment out the existing configurations and uncomment the myDataSource and TransactionManager bean sections, using the code and settings shown here.

   Consult your database administrator for the appropriate database connection parameters to use in the jdbcUrl property, which points to the OLTP database.

   a. Modify the myDataSource bean section to use the following code:

```
<bean id="myDataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
      destroy-method="close">
    <property name="driverClass">
```

```
            <value>oracle.jdbc.OracleDriver</value>
        </property>
        <property name="jdbcUrl">
            <value>jdbc:oracle:thin:@ebillingsrv:1521:oltp
        </value>
        </property>
        <property name="user">
            <value>oltp</value>
        </property>
        <property name="password">
            <value>oltp</value>
        </property>
    </bean>
```

b Modify the TransactionManager bean section to use the following code:

```
    <bean id="TransactionManager"
      class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
        <property name="dataSource" ref="myDataSource" />
    </bean>
```

6 Place the following JAR files in the *EDX_HOME*\bin\keymgmt\lib directory:

- (Oracle Database 11*g* only) The ojdbc5.jar file, found on your database server in the $ORACLE_HOME\jdbc\lib directory.
- (Oracle Database 10*g* only) The ojdbc14.jar file, found on your database server in the $ORACLE_HOME\jdbc\lib directory.
- (All versions) The jta.jar file, which can be found on your database server in the $ORACLE_HOME\jlib) directory.

7 Set the log file path. The master key update uses Log4j. The configuration file, log4j_keymgmt.xml, is located in the *EDX_HOME*\config directory. For instructions on how to use Log4j, see:

http://logging.apache.org/log4j/1.2/index.html

The master key update generates a default log file, keymgmt.log, in the working directory, *EDX_HOME*\bin\keymgmt. Make sure you have write permission in this directory before running the script.

8 Set the correct JAVA_HOME and PATH for Java and then run the master key update script, found in the *EDX_HOME*\bin\keymgmt directory. For example:

- UNIX:

```
export JAVA_HOME=/opt/java1.5

export PATH=$JAVA_HOME/bin:$PATH

./update_master_key.sh
```

- Windows:

```
set JAVA_HOME=c:\java1.5

set PATH=%JAVA_HOME%\bin;%PATH%

update_master_key.bat
```

9 Determine whether the update was successful:

a Check the log file. If the script was successful, the following message appears in the log: Master Key Update process is completed.

b Check the MasterKeyStore.properties file, found in the *EDX_HOME*\keystore\AES128 (or *EDX_HOME*\keystore\blowfish) directory. The script generates a new master key in this file with a timestamp. The first record contains the timestamp, followed by the previous master key with the same timestamp, and the current master key. The script saves the old master key file calledMasterKeyStore.properties.*timestamp*.

Note: Distribute new MasterKeyStore.properties file to all application servers to guarantee master key consistency.

c Check the following database tables:

- The SECURE_SUBKEY column in the EDX_SECURE_SUBKEY table will be updated.
- Code in the EDX_UMF_SEC_VALIDATIONCODE table will be updated.
- The KEY_UPDATE_FLAG column in the EDX_SECURE_SUBKEY table will be NULL.

10 If the master key update script ran successfully, restore the original persistence.xma.xml file to the *EDX_HOME*\xma\config\modules directory, and restart application server, and log in with your ID and password.

If the master key update script did not run successfully, follow these steps to troubleshoot the problem:

a If the script did not generate the new master key in the MasterKeyStore.properties file, verify the configuration of the persistence.xma.xml file and run the script again.

b If the script successfully generated the new master key in the MasterKeyStore.properties file, check the sub key flag in the KEY_UPDATE_FLAG column in the EDX_SECURE_SUBKEY table:

- If the column contains all NULL values, then you must restore the master key. Replace the MasterKeyStore.properties file in the *EDX_HOME*\keystore\AES128 (or *EDX_HOME*\keystore\blowfish) directory with the backup file.
- If the column contains any value other than NULL, you can run the script again.

c If it is not possible to run the master key update script successfully, restore the contents of the *EDX_HOME*\keystore directory and perform a full restore of the OLTP database.

12 Running the ETL Jobs Using Oracle Workflow Manager

This chapter describes the procedures for running Extract Transform Loading (ETL) jobs in Oracle Workflow Manager (OWF) and the Design Center in Oracle Warehouse Builder. It includes the following topics:

- Running the ETL_SUPER_PF_2 (ETL Loader) Job on page 147
- Running the OLAP_ACCEPT_REJECT Job on page 147
- Retrying the OLTP_LD_2 Job Process on page 150

Running the ETL_SUPER_PF_2 (ETL Loader) Job

To load data into Oracle Self-Service E-Billing, you use the ETL loader job, called ETL_SUPER_PF_2. This job appears in Oracle Workflow Builder under the process flow package ETL_SUP.

The ETL_SUPER_PF_2 job loads the ETL data into the OLAP database, then by default, automatically accepts the OLAP load and proceeds to update the OLTP database using the OLTP_LD_2 process.

If you run the ETL_SUPER_PF_2 job using the manual acceptance option, you must run the OLAP_ACCEPT_REJECT and OLTP_LD_2 jobs manually on the data file.

You run the ETL loader job from the Design Center in Oracle Workflow Builder. Oracle Warehouse Builder is a single, comprehensive tool for all aspects of data management. It provides data quality, data auditing, fully integrated relational and dimensional modeling, and full life cycle management of data and metadata.

To run the ETL_SUPER_PF_2 job

1. Log in to the Oracle Workflow Builder Design Center as the repository owner.
2. Launch the Control Center from the Tools menu.
3. Expand the EBILLING_ETL_LOCATION, ETL_PF_MODULE, and ETL_SUP nodes.
4. Right click the ETL_SUPER_PF_2 node (the ETL Loader job) and click Start.
5. A dialog box might appear stating that the object must be deployed before execution. If this dialog appears, click OK.

Running the OLAP_ACCEPT_REJECT Job

By default, the ETL_SUPER_PF_2 job uses the automatic acceptance option, which automatically accepts the OLAP load and updates the OLTP database. The OLAP_ACCEPT_REJECT job is available to run in the following situations:

- **To manually accept the file.** If you ran the ETL_SUPER_PF_2 job using the manual acceptance option (ran ETL_SUPER_PF_2 job with the IN_05_OLTP_FLAG parameter set to M), then you must complete the load process by running the OLAP_ACCEPT_REJECT job to accept the file, which initiates the OLTP_LD_2 job process to update the OLTP database and the post-ETL process to perform split billing task.
- **To manually reject the file.** If the ETL_SUPER_PF_2 job fails, run the OLAP_ACCEPT_REJECT job to reject the file and remove the data that was partially loaded into the OLAP database.

To manually accept or reject the ETL load using the OLAP_ACCEPT_REJECT job

1. Log in to the Oracle Workflow Builder Design Center as the repository owner.
2. Launch the Control Center from the Tools menu.
3. Go to the `EBILLING_ETL_LOCATION/ACPT_REJ/OLAP_ACCEPT_REJECT` directory.
4. Click Start to run the job.

5 Specify or verify the following input parameters.

Parameter	What to Enter or Select
P_FILE_NAME	Enter the full name of Oracle Self-Service E-Billing data file, for example, EBILLING_B2B-DATA-FILE-20111210.DAT
P_02_ACCEPT_REJECT	To accept the file, type Y; to reject the file, type N. The default value is N.
P_03_OLAP_FLAG	Verify that the value is 1, to require a manual acceptance for the OLTP update. (If the value is 0, the ETL load was run using the default setting to automatically accept the file.)
P_04_OLTP_FLAG	Verify that the value is M, to require manual acceptance for the OLTP update. (If the value is null, the ETL load was run using default setting to automatically accept the file.)
P_05_STATEMENT_KEY	Enter the statement key, which can be found in the statement fact table. (The default value is null; the default value setting is for rejecting a file, and data is removed from the OLAP side only.)

If you manually accept the file, the OLAP_ACCEPT_REJECT job completes the ETL load process by initiating the OLTP_LD_2 process, which reads data from the exchange table (EDX_RPT_ETL_XCHANGE) in the OLTP database to build accounts, service, and hierarchy tables in the OLTP database by invoking the following sub loaders.

Subloader	Description
Period loader	Inserts rows into the edx_omf_period table.
Account loader	Inserts rows into the edx_bsl_amf_baccount table (using select distinct account ID's).
Service loader	Inserts rows into the edx_omf_serviceagreement table.
OMF loader	Not applicable.

Subloader	Description
Company loader	Inserts rows into the company table, edx_bsl_cmf_company (selects distinct company ID's).
Hierarchy loader	The hierarchy loader performs the following tasks in both the OLAP and OLTP databases: ■ Reads data from the exchange table in the OLTP database and synchronizes hierarchy tables. ■ Creates hierarchies for companies in the company table that do not have hierarchies. ■ Puts any new accounts under these companies in their hierarchies. ■ Puts any new service agreements under these accounts in their hierarchies.

If you manually reject the ETL load file, the OLAP_ACCEPT_REJECT job removes the data from the OLAP database.

Retrying the OLTP_LD_2 Job Process

The OLTP_LD_2 process runs automatically as part of the following jobs:

- **ETL_SUPER_PF_2.** When run with the default automatic acceptance, the ETL loader job completes the acceptance and OLTP load automatically.
- **OLAP_ACCEPT_REJECT.** When you accept a file manually using the OLAP_ACCEPT_REJECT job, the job runs this process to complete the ETL load by updating the OLTP database. (The ETL loader job, ETL_SUPER_PF_2, ran using manual acceptance in this situation.)

If a failure occurs during the OLTP load in either job, follow these instructions to retry the OLTP_LD_2 job process independently.

To retry the OLTP_LD_2 job

1. Log in to the Oracle Workflow Builder Design Center as the repository owner.
2. Go to the Control Center from the Tools menu.
3. Go to the `ebilling_etl_location/OLTP_LD2/OLTP_LD_2` directory.
4. Click Start to run the OLTP_LD_2 job.
5. Specify the following input parameters.

Parameter	What to Enter or Select
IN_05_FEEDTYPE	B2B or B2C
IN_06_REGULAR_LOAD	TRUE or FALSE (Default is TRUE).

Parameter	What to Enter or Select
IN_07_FILENAME	The full name of the data file, for example, EBILLING_B2B-DATA-FILE-20110710.DAT
IN_08_FROM_ADDRESS	Valid email address
IN_09_TO_ADDRESS	Valid email address
IN_10_CC_address	Valid email address
IN_11_SMTP_PORT	Port number (default value is 25)
IN_12_SMTP_SERVER	SMTP server name

13 Error Messages

This appendix provides a listing of the job error messages that can appear in Oracle Self-Service E-Billing and the action required to correct the errors. It includes the following topics:

- Command Center Job Error Messages on page 153
- Payment Error Messages on page 159

Command Center Job Error Messages

The Command Center jobs can generate various types of error messages, listed in this section. For help with an error, create a service request (SR) on My Oracle Support. Alternatively, you can phone Oracle Global Customer Support directly to create a service request or get a status update on your current SR. Support phone numbers are listed on My Oracle Support.

You can view error log files using the Reporting menu option in the Command Center. For details on viewing error logs, see "About Message Log Files" on page 118.

Application Error Messages

The letters *APP* prefix all error message IDs that relate to applications.

Table 58 describes the application error messages.

Table 58. Application Error Messages

Message ID	Severity	Message Text
APP0001	Error	Error initializing application stored procedure call strings
APP0002	Error	Unable to interpret the docid: {0}
APP0003	Exception	Exception caught: {0}

Mailer Error Messages

The letters *MAI* prefix all error message IDs that relate to the mailer purge.

Table 59 describes the mailer error messages.

Table 59. Mailer Error Messages

Message ID	Severity	Message Text
MAI0001	Error	Error initializing stored procedure call strings for mailer purge.

Services Merger Error Messages

The letters *MGR* prefix all error message IDs that relate to the services merger.

Table 60 describes the service merger error messages.

Table 60. Services Merger Error Messages

Message ID	Severity	Message Text
MGR0001	Error	InvalidAppException occurred while trying to access the application path: {0}
MGR0002	Exception Error	{0} occurred while trying to access the application path: {1}
MGR0003	Exception	Exception occurred while trying to access the input stream: {0}
MGR0004	Exception	Exception occurred while reading versioning information: {0}
MGR0008	Exception Error	Unable to de-serialize the parameter properties object: {0} Unable to interpret the docid: {0}
MGR0009	Exception	Exception occurred while application information: {0}
MGR0010	Exception	Exception occurred while reading versioning information: {0}
MGR0011	Exception	C++ merger code threw an exception: {0}
MGR0013	Error	UnsatisfiedLinkError, Check LD_LIBRARY_PATH
MGR0014	Error	Unable to interpret the docid: {0}
MGR0015	Error	Exception occurred while preparing to retrieve the document: {0}

Mail Server Error Messages

The letters *MNS* prefix all error message IDs that relate to the mail server.

Table 61 describes the messages.

Table 61. Mail Server Error Messages

Message ID	Severity	Message Text
MNS0001	Information	Started
MNS0002	Exception	Exception caught: {0}
MNS0003	Information	Finished Processing
MNS0004	Error	Something is very wrong. Mail servers might not be working.
MNS0005	Error	Total Accounts = {0}, Total Tried = {1}, Total Emails Sent = {2}
MNS0006	Exception	Exception caught: {0}

Table 61. Mail Server Error Messages

Message ID	Severity	Message Text
MNS0007	Exception	Exception caught: {0}
MNS0008	Exception	Exception caught: {0}
MNS0009	Exception	Exception caught: {0}
MNS0010	Exception	Exception caught: {0}
MNS0011	Information	Created and starting...
MNS0012	Exception	Exception caught: {0}
MNS0013	Exception	Exception caught: {0}
MNS0014	Exception	Exception caught: {0}
MNS0015	Exception	Exception caught: {0}
MNS0016	Exception	Exception caught: {0}
MNS0017	Exception	Exception caught: {0}
MNS0018	Exception	Exception caught: {0}
MNS0019	Exception	Exception caught: {0}
MNS0020	Error	Error initializing stored procedures call strings
MNS0021	Information	Failed to send mails, tried retry number of times, Perhaps Mail servers are not working.
MNS0022	Exception	Exception caught: {0}

Service Monitoring Error Messages

The letters *MON* prefix all error message IDs that relate to the services monitoring.

Table 62 describes the service monitoring error messages.

Table 62. Services Monitoring Error Messages

Message ID	Severity	Message Text
MON0001	Exception	Exception occurred while looking up EJB: {0}
MON0002	Exception	Exception occurred while looking up EJB: {0}
MON0003	Information	Monitor created
MON0004	Exception	Exception occurred while looking up EJB: {0}
MON0005	Information	Monitor removed

Stored Procedure Error Messages

The letters *PDB* prefix all error message IDs that relate to stored procedures.

Table 63 describes the stored procedure error messages.

Table 63. Stored Procedure Error Messages

Message ID	Severity	Message Text
PDB0001	Error	Error initializing stored procedure call strings

Process Workflow Controller (PWC) Error Messages

The letters *PTK* prefix all error message IDs that relate to PWC.

Table 64 describes the PWC error messages.

Table 64. PWC Error Messages

Message ID	Severity	Message Text
PTK0001	Information	Created and starting
PTK0002	Exception	Exception caught:{0}
PTK0003	Exception	Exception caught:{0}
PTK0004	Exception	Exception caught:{0}
PTK0005	Exception	Exception caught:{0}
PTK0006	Exception	Exception caught:{0}
PTK0007	Exception	Exception caught:{0}
PTK0008	Exception	Exception caught:{0}
PTK0009	Exception	Exception caught:{0}
PTK0010	Exception	Exception caught:{0}
PTK0011	Exception	Exception caught:{0}
PTK0012	Exception	Exception caught:{0}
PTK0013	Exception	Exception caught:{0}
PTK0014	Exception	Exception caught:{0}
PTK0015	Exception	Exception caught:{0}
PTK0016	Information	Finished processing task

Purging Logs Error Messages

The letters *PUR* prefix all error message IDs that relate to purging logs.

Table 65 describes the purging log error messages.

Table 65. Purging Logs Error Messages

Message ID	Severity	Message Text
PUR0001	Information	{0} purged {1} records from the database
PUR0002	Information	The task configuration for {0} has a negative purge age of {1}. No purging will occur.
PUR0003	Error	The following exception was caught during {0} {1}

Scheduler Error Messages

The letters *SCH* prefix error message IDs that relate to Scheduler.

Table 66 describes the Scheduler error messages.

Table 66. Scheduler Error Messages

Message ID	Severity	Message Text
SCH0001	Information	Starting JobProcessor for ({0}) job instance: {1}
SCH0002	Exception	Exception in processJobs: {0}
SCH0003	Information	PWC Scheduler started
SCH0004	Exception	Reprocessing of jobs failed: {0}
SCH0005	Exception	Processing of jobs failed: {0
SCH0006	Exception	Nonrecoverable error in PWC Scheduler!: {0}
SCH0007	Information	Starting job instance thread for: {0}
SCH0008	Error	Job instance initialization failed for: {0} {1}
SCH0009	Error	Failed to get Job/Schedule Object for job instance: {0} {1}
SCH0010	Error	Failed to update failed job status for: {0} {1}
SCH0011	Information	Job instance: {0}; Starting task: {1}; order: {2}
SCH0012	Information	Job instance: {0}; Done task: {1}; order: {2}
SCH0013	Information	Job instance: {0}; Starting task: {1}; order: {2}
SCH0014	Error	Failed to update failed job status for: {0} {1}
SCH0015	Error	Failure in job processor thread for: {0} {1}
SCH0016	Error	Failed to update failed job status for: {0} {1}
SCH0017	Error	Status forced to “Failed” from {0} for hung job instance: {1}
SCH0018	Error	PWC Scheduler unable to force “Fail” hung job instances: {0}
SCH0019	Error	Failed to define next schedule for job instance: {0}
SCH0020	Information	Done job instance thread for: {0}

Scanner Error Messages

The letters *SCN* prefix all error message IDs that relate to the scheduling scanner.

Table 67 describes the scanner error messages.

Table 67. Scanner Error Messages

Message ID	Severity	Message Text
SCN0001	Error	Scanner.getTaskParams: Failed to get task config params - {0}
SCN0002	Error	Scanner.setTaskParams - missing param: {0}, for job {1}, ddn {2}, task order {3}
SCN0003	Information	Scanner.setTaskParams({0}, {1}, {2}): {3}
SCN0004	Error	Scanner.setTaskParams: Failed to set task config params - {0}
SCN0005	Error	Scanner.isTaskConfigValid: Failed to find input directory: {0}
SCN0006	Error	Scanner.isTaskConfigValid: Failed to create output directory: {0}
SCN0007	Error	Scanner.isTaskConfigValid: Failed while validating config params - {0}
SCN0008	Information	Scanner.processTask({0}, {1})
SCN0009	Error	Scanner.processTask: Failed to create ddn volume - {0}
SCN0010	Error	Scanner.processTask: Failed caused by an invalid input file: {0}
SCN0011	Information	Scanner.processTask({0}, {1}): Attempting to move {2} -> {3}
SCN0012	Error	Scanner.processTask({0}, {1}) - attempt to move: {2} -> {3} failed
SCN0013	Information	Scanner.processTask({0}, {1}) - attempt to move: {2} -> {3} succeeded
SCN0014	Error	Scanner.processTask: failed to process - {0}
SCN0015	Information	Scanner.processTask: {0} : file length is 0.

Shell Command Error Messages

The letters *SCT* prefix all error message IDs relate to the shell job task, ShellCmdTask.

Table 68 describes the shell command error messages.

Table 68. Shell Command Error Messages

Message ID	Severity	Message Text
SCT0001	Error	Exception caught: {0}
SCT0002	Error	Shell Command unable to finish: {0}
SCT0003	Information	ShellCmdTask: About to execute the following shell command: {0}
SCT0004	Information	ShellCmdTask: Return Value = {0}

Table 68. Shell Command Error Messages

Message ID	Severity	Message Text
SCT0005	Information	ShellCmdTask: Shell output: {0} DVN: {1}
SC0006	Error	ShellCmdTask: processTask: Failed to create ddn volume - {0}
SCT0007	Error	ShellCmdTask: processTask: Unable to set task output: {0}
SCT0008	Information	ShellCmdTask: DVN changed. Shell output: {0} DVN: {1}

Services Versioning Error Messages

The letters *VRS* prefix all error message IDs that relate to services versioning.

Table 69 describes the services versioning error messages.

Table 69. Services Versioning Error Messages

Message ID	Severity	Message Text
VRS0001	Exception	Exception occurred while creating IVersionSetReader object locally: {0}
VRS0002	Exception	Exception occurred whilst creating IVersionSetReader remote object: {0}
VRS0003	Exception	Unable to instantiate remote IVersionSetReader interface: {0}
VRS0004	Exception	Exception occurred whilst creating IVersionSetWriter object locally: {0}
VRS0005	Exception	Exception occurred whilst creating IVersionSetWriter remote object: {0}
VRS0006	Exception	Unable to instantiate remote IVersionSetWriter interface: {0}
VRS0007	Exception	Exception occurred whilst creating IVersionedObj object locally: {0}
VRS0008	Exception	Exception occurred whilst creating IVersionedObj remote object: {0}
VRS0009	Exception	Unable to instantiate remote IVersionedObj interface: {0}
VRS0010	Exception	Unable read data required for instantiating remote version interface implementations

Payment Error Messages

When Payment module jobs generate error messages, it stores them in the Payment error log file, com.edocs.payment.log.

The letters *PMT* prefix all error message IDs that relate to the Payment module.

Table 70 lists the error messages that can occur when running Payment jobs.

Table 70. Payment Job Error Messages

Error Message	Description
NoPaymentAccountException	An ACH customer cannot be activated (the ACH payment gateway only supports up to three payment accounts for each customer) and can result in the failure of the payment application. This type of error typically indicates a problem with the Enrollment JSP page and must be discovered during initial setup and testing.
FileIOException	There is a problem regarding the reading and writing of ACH files. Verify the existence of the ACH file output directory and file input directory, and make sure the permissions on them are correct.
CassetteException	There is a general problem regarding an ACH operation and the interpretation of the error depends on the specific error message. This error message is sometimes used as a wrapper for other errors.
AchFormatException	There is a problem with the ACH file format. One common source of this error is that the length of supplied data is greater than the length allowed by ACH.
TemplateException	One or more ACH template files failed during processing by the Template engine. In some cases, this error can be corrected during setup and configuration. Also check the ACH template formats to make sure they are valid.
SQLException	A generic exception accessing the Payment module database. This message can be generated by a variety of errors, but one common source is the failure to establish connection pools in the Payment database. In this case, check whether the database server is still running, and check the application server configuration.
OperationNotSupported Exception	The current payment operation is not supported by ACH. This error message typically indicates faulty coding and must be corrected by a developer immediately.
RemoteException	A generic exception which typically serves as the wrapper for other exceptions, and typically indicates a failed payment operation, which cannot be corrected by running the operation again.

Table 70. Payment Job Error Messages

Error Message	Description
DuplicateKeyException	An exception that occurs when there are two check payments in the database with the same payment ID. The Payment module uses a timestamp (in milliseconds) to assign payment IDs. This type of error typically does not require manual intervention. Instead, the customer will receive an error message and then be prompted to submit the check payment again.
NoDataFoundException	There is no data being extracted from the Payment module database. This type of error is typically corrected by the application during processing.

Index

A

ACH
- addenda records 97
- change codes 95
- effective date 97
- Federal holidays 88
- file 97
- file input and output directories 87
- immediate destination 86
- immediate origin 87
- logic 41
- multiple DDNs in ACH files 97
- prenote 40
- return codes 95, 96
- return directory 87
- return file format 44
- return types 42
- settlement date 97
- submit directory 87
- template directory 87

AchFormatException 160
Action column 58
adding a new customer account for check payment services 93
Address Verification Service 101
administering jobs 57
administrator activity
- audit 120
- purging 140

alert groups
- creating 70
- deleting 71
- editing 70

alert profiles
- creating 71
- deleting 73
- updating 72

alert service 73
- plug-in 71, 74
- status 71

alert settings 71, 74
alert types 71, 74
alerts
- applying to a job schedule 73
- process of configuring 69

applications
- creating 15
- deleting 126
- listing jobs 59

applying a blackout calendar to a job schedule 69
applying alerts to a job schedule 73
audits
- administrator activity 120
- purged data 142
- purging administrator activity history 140

Automated Clearing House (ACH) logic 41
automatic payment 103

B

bank holidays 39, 88
BatchReportScheduler job 19, 25
blackout calendars
- applying to a job schedule 69
- creating 68
- deleting 69
- editing 68
- process of configuring 67

C

canceling and rerunning failed jobs
- jobs
  - canceling 62

Cancelled status 60
cartridges 76, 84
CassetteException 160
changing a job configuration 17
changing an administrator's password 13
changing an administrator's personal information 13
check and credit card gateway configuration parameters (regular and external payee DDNs) 84
check cartridge 84
check gateway configuration parameters (external payee DDNs only) 88
check gateway configuration parameters (regular payee DDNs only) 85
check gateway parameters 83
- regular payee DDN 85

check payment status flow 98
check payments
- clearing checks 85, 88, 102

gateway 79
transaction cycle 94
check returns 42
Command Center
logging in 13
Main Console 57
monitoring production 57
secure password 11
task status 60
Company ID 92
configuring a payment gateway 79
configuring the batch refunds XML 50
contacts, email notification 70
creating a blackout calendar 68
creating a custom job using the pmtCustom job 55
creating a new application 15
creating a new job 16
creating a system administrator password 12
creating an alert group 70
creating an alert profile 71
creating and configuring jobs 19
credit card gateway configuration parameters (external payee DDNs only) 91
credit card gateway configuration parameters (regular payee DDNs only) 89
credit card payment status 101
credit card payment transactions 98
credit cards
cartridge 84
gateway parameters 83
overview 75
transaction overview 99
user options 98
credit reversals 98
custom alert service
plug-in 72
custom jobs 20
creating 55
payment 32
cycle of an ACH check payment transaction 93

D

data
purging 127
purging in batch 141
purging locked administrators in Command Center 140
purging payment account and transactional data 128
purging user data 136
purging validation codes 139
data protection 17
database
backup and recovery 125
maintaining tables 125
sizing tables 123
database tables affected by recurring payments 106
DDN
and payment_profile 124
as payee name 92
configuration parameters 82
creating job alert profiles 71
in XML format 51
mapping to a data source EJB 15
multiple 41, 42, 97
payee 84
payment gateway 84
payment portal 75
regular payee 89
selected 72
deactivating the secure administrator ID 14
deleting a blackout calendar 69
deleting a job 17
deleting an alert group 71
deleting an alert profile 73
deleting an application 126
deleting applications 126
directory, history 41
displaying current job status 59
Done, recurring status 60
DuplicateKeyException 161

E

editing a blackout calendar 68
editing an alert group 70
email
notification for job processing 69
notifications 77
template 37
enrolling a system administrator user 11
enrollment
model 84
setting properties for Payment 126
XML file 84
errors
and warnings in payment reports 102
log files, payment error messages 153
messages 153, 160
payment module 159
ETL jobs

data protection 18
OLAP_ACCEPT_REJECT 147
running the ETL_SUPER_PF_2 (ETL Loader) job 147
ETL_SUPER_PF_2 147

F

failed job 59, 62
alerts 69
Federal holidays 89, 97
FileIOException 160

G

gateway configuration parameters
check (external payee DDNs only) 88
check (regular payee DDNs only) 85
check and credit card (regular and external payee DDNs) 84
credit card (external payee DDNs only) 91
credit card (regular payee DDNs only) 89
gateways
configuring 79
parameters 83
global alert settings 74
global configuration parameters for payment 80

H

hierarchies and hierarchy assignments
purging 134
HierarchyCleanUp job 20, 29
HierarchyCopy job 20, 28
HierarchyImporter job 20, 27
HierarchyPurge job 20, 30
history directory 41
holidays
ACH gateway parameters 86
bank 88
payment scheduling 39
sending payment batch files on 35
U.S. Federal 89, 97

I

IAlertServicePlugin 71, 72, 74
Immediate Destination Name parameter 87
Immediate Destination parameter 86
Immediate Origin Name parameter 87
Immediate Origin parameter 86
Implementation of ICheckSubmitPlugIn 87
indexer DDN 82
parameters 82
instant payments 99
IPaymentAccountAccessor 85
IPaymentAccountUserAccessor 85
IUserAccountUserAccessor 84

J

job alerts
process of configuring 69
job reports 117
job schedule
applying alerts 73
Job Type column 58
job types
payment 30
production 19
jobs
about 16
administering 57
alert notifications 69
applying alerts to a schedule 73
BatchReportScheduler 19, 25
cancelling 62
changing a configuration 17
changing configuration 17
creating 16
custom 20
deleting 17
displaying current status 59
email notification 69
error messages 153
ETL_SUPER_PF_2 (ETL Loader) 147
failed 62
HierarchyCleanUp 20, 29
HierarchyCopy 20, 28
HierarchyImporter 20, 27
HierarchyPurge 20, 30
history and statistics 117
listing for an application 59
monitoring 57, 58
Notifier 19, 20, 21
OLAP_ACCEPT_REJECT 147
PasswordExpNotify 19, 25
payment
job types 17
PaymentDueNotification 31, 47
pmtAllCheckTasks 31, 46, 48
pmtARIntegrator 31, 45
pmtCheckSubmit 31, 38
pmtCheckUpdate 31, 41
pmtConfirmEnroll 31, 36
pmtCreditCardExpNotify 32, 49, 54
pmtCreditCardSubmit 31
pmtCustom 32, 55
pmtPaymentRefund 32, 49

pmtPaymentReminder 31, 36, 37
pmtRecurringPayment 31, 32
pmtSubmitEnroll 31, 35, 54
production job types 19
Purge Logs 19, 26
recurring 60
ReportCleanUp 20, 26
rerunning 62
scheduling 65
scheduling payment jobs 77
Shell 27
ShellJob 20, 27
sorting on the Main Console 59
starting manually 63
StatementReady 19, 20, 30
status 59
ThresholdExceedNotify 54
viewing status 59

L

Last Run column 58
listing jobs for an application 59
log files 102, 118
message 118
payment history 102
logging in to the Command Center 13

M

Main Console 57, 59
managing the payment module database 121
mapping your application to a data source EJB 15
Master Key Update
running 143
message log files 118
monitoring production jobs 58
monitoring service status 119
multiple DDNs in ACH files 97
multiple recurring payments 104

N

Next Run column 58
No operation status 60
No operation, recurring status 60
NOC
check transaction status 98
codes 95
returns 43
NOC returns 42
NOC transactions 97
NoDataFoundException 161
NoPaymentAccountException 160
Notifier job 19, 20, 21

O

ODFI
ACH payment gateway setting 92
ACH return code 96
and cancel recurring payment 35
and check payment transactions 94
and immediate destination 86
and immediate origin 86
and immediate origin name 87
configuring payee bank account 92
routing number 41
OLAP 18, 108, 109, 111
OLAP_ACCEPT_REJECT job 147
OLTP 18
OLTP_LD_2 job 150
OperationNotSupportedException 160
Oracle Data Guard 17

P

parameters
check and credit card gateways 83
global payment configuration 80
indexer DDN 82
payee bank account 92
PasswordExpNotify job 19, 25
passwords 11
payment user 123
system administrator, changing 13
system administrator, creating 12
payee bank account parameters 92
payee DDN 84
payment
ACH check payment transaction cycle 93
cartridges 76
credit card transactions 98
error messages 159
features 75
gateways 93
instant credit card payments 99
jobs 17
managing the database 121
processor 91
purging account and transactional data 128
reports 102
using credit cards 37, 99
viewing reports 102
payment job types 30
PaymentDueNotification job 31, 47
payments
automatic 103
check status flow 98

instant 99
multiple recurring 104
recurring 32, 103
scheduled credit card 100
scheduling 39
PayPal Payflow Pro 91
and AVS 101
confirmation number 48
setting up certification 78
threads 49
plug-ins
ACH gateway parameters 87
AVS 101
credit card payment configuration 91
custom alert service 72
IAlertServicePlugin 74
scheduled credit card payment 100
pmtAllCheckTasks job 31, 46, 48
pmtARIntegrator job 31, 45
pmtCheckSubmit job 31, 38
ACH company name 92
ACH immediate origin 86
and ACH effective date 97
date 97
empty ACH configuration 86
immediate destination 86
ODFI configuration 92
pmtCheckUpdate job 31, 41
ACH company name 92
ACH immediate destination 86
ACH immediate origin 86
and ACH change codes 95
and ACH return codes 96
pmtConfirmEnroll job 31, 36
pmtCreditCardExpNotify job 32, 49, 54
pmtCreditCardSubmit job 31, 100
pmtCustom job 32, 55
pmtPaymentRefund job 32, 49
pmtPaymentReminder job 31, 36, 37
pmtRecurringPayment job 31, 32
pmtSubmitEnroll job 31, 35, 54
prenote returns 42
processes
configuring a blackout calendar 67
configuring job alerts 69
Processing status 59
production job types 19
Purge Logs job 19, 26
purging data 127
administrator activity 140
auditing 142
B2B user hierarchy assignments 135
billing accounts and services 136
billing hierarchies 134
business hierarchies 135
hierarchies and hierarchy assignments 134
in batch 141
locked administrators in Command Center 140
payment account and transactional 128
payment accounts 129
payment history 130
payment invoices 132
payment notifications 133
pending payments 128
recurring payments 131
user data 136
validation codes 139

Q

queries
database 122
job reports 118
name 45
payment transactions 52
template file 45
XML file 45

R

reactivating the secure administrator ID 14
recurring payments 103
about 103
configuring 32
database tables 106
examples 106
job 31
multiple 104
transaction cycle 104
RemoteException 160
ReportCleanUp job 20, 26
reports 117
payment 102
retrying 150
retrying the OLTP_LD_2 job process 150
reversals, credit 98
Run Now button 63
Run Time column 58
running the ETL_SUPER_PF_2 (ETL Loader) job 147
running the Master Key Update 143
running the OLAP_ACCEPT_REJECT job 147

S

scheduled credit card payments 100
scheduling jobs 65
applying a blackout calendar 67, 69
applying alerts 73

payment 77
secure administrator ID and password 11
deactivating and reactivating 14
service status, monitoring 119
services 20
setting enrollment properties (Payment) 126
setting up PayPal Payflow Pro certification 78
ShellJob job 20, 27
sorting jobs on the Main Console 59
SQLException 160
starting a job manually 63
StatementReady job 19, 20, 30
status 58, 59, 60
successful job email notification 69
supported SEC codes 95

T

task status 60
TemplateException 160
templates 76
ACH 42, 87
and front-end GUIs 76
and multiple DDNs 97
AR 45
email 37, 78
query 45
XML 46
XSLT 45, 46
threads, multiple PayPal Payflow Pro 49
ThresholdExceedNotify job 54

U

U.S. Federal holidays 97
Update Payment Enrollment in Case of NOC parameter 86
updating an alert profile 72
user data, purging 136

V

viewing job status 59
viewing payment reports 102
viewing task status 60

W

warnings 102

X

XML 45, 46
configuring batch refunds 50
XSLT 45, 46