Administration Guide for Oracle Self-Service E-Billing

Version 6.2

October 2013

ORACLE

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Contents

Chapter 6: Scheduling Jobs

Chapter 7: Configuring the Payments Module

Chapter 8: Payment Transactions

Index

1 What's New in This Release

What's New in Administration Guide for Oracle Self-Service E-Billing, Version 6.2

Table 1 lists changes described in this version of the documentation to support release 6.2 of the software.

Table 1. New Product Features in Administration Guide for Oracle Self-Service E-Billing, Version 6.2

Topic	Description
"StatementSummaryPDFProvider Job" on page 29 "StatementSummaryDaisyProvider Job" on page 30 "BatchReportProcessor Job" on page 31 "ReportCleanUp Job" on page 39	Modified topics. Updated the file store parameters with the option to use a content management server.
"pmtCheckSubmit Job" on page 52	Modified topic. Two parameters have been added to enable separate lines for ACH output.
"Configuring a Payment Gateway" on page 99	Modified topic. The payment module has been refactored. Parameters for configuring the Chase Paymentech Orbital Payment Gateway have been added.
"Purging Statement and Invoice Fact Data" on page 167	Modified topic. Added the Statement Content OLAP database table to the list of purged tables. This table is used for storing files on a content management server.
"Loading Billed Data" on page 177	Modified topic. Added a new parameter that lets you suppress the loading of company, account, and service data during the ETL load process. This option is for implementations that use Web Services to provision this data.
"Loading Unbilled Data" on page 183	New topic. Describes how to load an ETL file with unbilled data into Oracle Self-Service E-Billing.
"Loading Prepaid Data" on page 185	New topic. Describes how to load an ETL file with prepaid data into Oracle Self-Service E-Billing.

Table 1. New Product Features in Administration Guide for Oracle Self-Service E-Billing, Version 6.2

Topic	Description
"Rejecting an Unbilled or Prepaid Data File" on page 189	New topic. Describes how to reject unbilled or prepaid data files that have been loaded using new ETL processes.
"Purging Unbilled or Prepaid Data Records" on page 189	New topic. Describes how to purge unbilled and prepaid data records using new ETL processes.
"Resolving Checksum Errors" on page 187	New topic. Describes how to run the new ETL checksum troubleshooting process to help resolve checksum errors that can occur when loading unbilled and prepaid data.

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. This chapter includes the following topics:

Enrolling the Initial Bootstrap Administrator User

The bootstrap accesses the Command Center using the user name and password defined during the installation process. Only this user name and password can be used for the initial login to the Command Center. After accessing Oracle Self-Service E-Billing initially, the bootstrap user must immediately enroll, creating a personal administrator user ID.

After the bootstrap user enrolls, the next time she logs in, the Command Center prompts her to create a new password. When the bootstrap user has a personal Command Center ID and password, she can log in and create, or enroll, additional administrators who will have full access to the Command Center functions. Each enrolled system administrator is also prompted to create a password at first login.

You can change the default enrollment ID and password requirements. For information about setting the enrollment properties, see "Setting Enrollment Properties" on page 146. Also see *Implementation Guide for Oracle Self-Service E-Billing*.

NOTE: Preserve the bootstrap 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 bootstrap 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 administrator ID and password created for the bootstrap user during the installation process. 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 bootstrap administrator ID created during the installation.
- 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, then 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, then 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 the Initial Bootstrap Administrator User" on page 11.

Changing an Administrator's Password or Personal Information

A system administrator can change his or her password for logging in to the Command Center at any time. An administrator's password automatically expires after the configured number of days.

Administrators can also update 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 "ThresholdExceedNotify Job" on page 69.

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 Bootstrap Administrator ID in the Command Center

You can deactivate and reactivate the bootstrap administrator's Command Center ID when needed.

To deactivate or reactivate the bootstrap administrator ID

1. Log into the Oracle Self-Service E-Billing instance using SQL*Plus, not as SYSDBA:

   OLTP schema user_name/*OLTP schema password*@*EBILL TNS name*

   where:

   - *OLTP schema user_name* is the name of the OLTP schema user.
   - *OLTP schema password* is the password of the OLTP schema user.
   - *EBILL TNS name* is the TNS name for the Oracle Self-Service E-Billing instance.
2. To deactivate the bootstrap administrator's ID, run the following command:

   ```
   SQL>exec EDX_PKG_BOOTSTRAPUSER.DISABLE_CC_DEFAULT_ADMIN

   SQL>commit;
   ```

   To reactivate the bootstrap administrator's ID, run the following command:

   ```
   SQL>exec EDX_PKG_BOOTSTRAPUSER.ENABLE_CC_DEFAULT_ADMIN

   SQL>commit;
   ```

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. This chapter includes the following topics:

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.

You can optionally specify a customized billing period for your implementation. For details, see *Implementation Guide for Oracle Self-Service E-Billing*.

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`.

   For more information about data source EJBs. see"About Mapping Your Application to a Data Source EJB" on page 18.
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 Mapping Your Application to a Data Source EJB

You must specify a data source EJB for each application data definition (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.

The Data Source Name is the real, global JNDI name as opposed to the local JNDI name, as in

```
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.

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. Setting the data source EJB at the DDN level 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.

Deleting an Application

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

If you are testing in a quality assurance or development environment, then 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 and 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. For more information about the Purge Logs job, see “Purge Logs Job” on page 39.

To delete an 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, then click the box in the Delete column for each job and 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 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 billing production jobs, see "Billing Jobs" on page 23. For a list of available payment jobs, see "Payment Jobs" on page 44.

  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 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 "Command Center Jobs and Configuration Parameters" on page 23.
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 and the job status is Processing, then 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. 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 a hardware failure or data corruption.

Oracle Self-Service E-Billing uses a database instance for data storage. The database instance stores historical data for analytics, using the OLAP schema, and it stores transactional data using the OLTP schema. Oracle Self-Service E-Billing creates certain transactional data from the OLTP schema and synchronizes it with the OLAP schema.

The synchronization allows real-time analysis of the changes made in OLTP, of which the hierarchy structure is one example. Users can create the hierarchy structure. The newly created hierarchy is stored in the OLTP schema and then pushed to the OLAP schema. The analytic reports then can use the newly updated hierarchy structure.

You can set up a primary database and a standby database for the Oracle Self-Service E-Billing instance.

Data created from the ETL process is first loaded to the OLAP schema. After the data is committed and validated in the OLAP schema, some key data is populated in the OLTP exchange tables. In cases where the OLAP or OLTP schema is unavailable, the counterpart of each schema keeps all the data.

The database works 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. For data created by online users, the distributed database transactions are used to guarantee data integrity. If one of the database schemas is unavailable, then no new data is written to either schema.

4 Command Center Jobs and Configuration Parameters

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

Billing Jobs

Table 2 lists the billing production jobs available with Oracle Self-Service E-Billing.

Table 2. Billing 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 "StatementReady Job" on page 25.
Notifier	Sends the emails and SMS messages stored in the MESSENGER_GROUP_TABLE to users. You run the Notifier job after the StatementReady, Payment Reminder, Payment Due Notification, and pmtCCExpiration jobs. For details on configuring a Notifier job, see "Notifier Job" on page 25.
StatementSummaryPDF Provider	Sends an email with a statement summary to enrolled users whose accounts have new statements ready in Oracle Self-Service E-Billing. Users must also have selected the My Bill Summary is Ready via PDF notification option to receive this email. For details on configuring a StatementSummaryPDFProvider job, see "StatementSummaryPDFProvider Job" on page 29.
StatementSummaryDaisyProvider	Sends an email with a statement in DAISY audio file to enrolled users whose accounts have new statements ready in Oracle Self-Service E-Billing, and who have chosen to receive DAISY audio file statements. Individual users must request that a customer service representative enable their interface with the DAISY file option. For details on configuring a StatementSummaryDaisyProvider job, see "StatementSummaryDaisyProvider Job" on page 30.

Table 2. Billing Production Jobs

Job Name	Description
BatchReportScheduler	Schedules one-time or recurring batch reports that are set to run during a period of up to five days ahead. For details on configuring a BatchReportScheduler job, see "BatchReportScheduler Job" on page 31.
BatchReportProcessor	The BatchReportProcessor job generates all scheduled batch reports and makes them available online. For details on configuring a BatchReportProcessor job, see "BatchReportProcessor Job" on page 31.
PasswordExpNotify	Generates a warning email to system administrators whose password is set to expire within the configured number of days. For details on configuring a PasswordExpNotify job, see "PasswordExpNotify Job" on page 32.
EmailBounceBack	Flags invalid email addresses in the profiles of customers with undeliverable notifications. For details on configuring an EmailBounceBack job, see "EmailBounceBack Job" on page 32.
LoadExternalB2BUsers	Loads existing or new B2B company users to Oracle Self-Service E-Billing from an external authentication system, such as a single-sign on system. For details on configuring a LoadExternalB2BUsers job, see "LoadExternalB2BUsers Job" on page 33.
LoadExternalB2CUsers	Loads existing or new B2C users to Oracle Self-Service E-Billing from an external authentication system, such as a single-sign on system. For details on configuring a LoadExternalB2CUsers job, see "LoadExternalB2CUsers Job" on page 35.
LoadExternalCSRUsers	Loads existing or new customer service representative users to Oracle Self-Service E-Billing from an external authentication system, such as a single-sign on system. For details on configuring a LoadExternalCSRUsers job, see "LoadExternalCSRUsers Job" on page 36.
DefUsrAcctRelationship	Adds and removes user-account relationships from Oracle Self-Service E-Billing in batch mode (Consumer Edition only). For details on configuring a DefUsrAcctRelationship job, see "DefUsrAcctRelationship Job" on page 37.
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 "Purge Logs Job" on page 39.

Table 2. Billing Production Jobs

Job Name	Description
ReportCleanUp	Deletes batch report files and related records from the database. For details on configuring a ReportCleanUp job, see "ReportCleanUp Job" on page 39.
ShellJob	Provides a way to run a custom 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 "ShellJob" on page 40.

StatementReady Job

The StatementReady job sends email and SMS messages 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 or an SMS message to the corresponding users letting them know that they have a bill. Email and SMS messages are sent only after running the Notifier job. There are no configuration parameters for the StatementReady job.

Notifier Job

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

To send queued email or SMS messages, you must run Notifier after any job that was configured to generate email or SMS notifications. You can configure Notifier to run for all notification types or for just one of the individual message types described in Table 4 on page 27.

The Notifier job sends messages for entries that are stored in the MESSENGER_GROUP_TABLE table in the OLTP database, composes the correct message using message category templates, and sends the notification to the users to whom the messages are addressed.

The Notifier job consists of two tasks:

- NotificationComposer
- NotificationDispatch

Parameters for Configuring the NotificationComposer Task

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

Parameters for Configuring the NotificationComposer Task

Table 3 describes the configuration parameters for the NotificationComposer task.

Table 3. Parameters for Configuring the NotificationComposer Task

Parameter	Description
Skip task	When set to the default value N, the task is not skipped when running the Notifier job. If the value is Y, then the job is not executed.
Message type	Select the type of message that the Notifier job picks up when it runs. You can choose an individual message type or all messages. You must run the appropriate notification jobs before running Notifier.

Table 4 describes the valid message types you can specify in the Message type field for the NotificationComposer task.

Table 4. Message Types

Message Type	Notification Description
AllNotifications	All message types.
BillNotification	The user has a statement ready for viewing online.
BillNotificationwithPDF	The user has a statement ready for viewing in the attached PDF file.
BillNotificationwithDaisy	The user has a statement ready for listening in the attached DAISY audio file.
CreditCardExpiryNotification	The user's credit card is expiring. 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 or SMS message to the recipients. Users who created the batch report receive the message.
PaymentDueNotification	A payment is due. For PaymentDueNotification job notifications.
PaymentScheduledNotification	Notifications for scheduled payments from the pmtPaymentRecurringPayment job.
PaymentSuccessNotification	Notifications for successful payments from the pmtPaymentReminder job.
PaymentFailureNotification	Notifications from the pmtPaymentReminder job for failed payments.
PaymentExceedNotification	Notifications from the pmtPaymentRecurringPayment job for scheduled payments that have reached the user's threshold.

Table 4. Message Types

Message Type	Notification 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 or SMS 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 or SMS 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 or SMS 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 or SMS message to the recipients. Users who have access to the billing account through billing hierarchy receive this email.
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 or SMS message to the recipients.
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 or SMS message to the recipients.
StatementExceedNotification	The user has exceeded his or her bill threshold.
PaymentNotifyEnrollNotification	The user has enrolled to make payments.
PaymentCancelNotification	The user has canceled a payment.
ScheduledPaymentCancelNotification	The user has canceled a scheduled payment.

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. Using the task parameter value, the entry can be removed from the table and from the message repository after the message is sent successfully.

Table 5 describes the configuration parameters for the NotificationDispatcher task.

Table 5. Parameters for Configuring the NotificationDispatcher Task

Parameter	Description
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, then 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. If the value is Y, then the task is 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.

StatementSummaryPDFProvider Job

The StatementSummaryPDFProvider job sends an email with a statement summary to enrolled users whose accounts have new statements ready in Oracle Self-Service E-Billing. Users must also have selected the My Bill Summary is Ready via PDF notification option to receive this email.

This job uses information passed from the ETL process and finds all users whose account statements have recently been loaded and processed, and sends email with a statement summary as a PDF attachment to the corresponding users. Email messages are sent only after running the Notifier job.

NOTE: There is no dependence between the StatementSummaryPDFProvider job and the StatementReady job. Each job can run independently.

Parameters for Configuring StatementSummaryPDFProviderTask

Table 6 shows the configuration parameter for StatementSummaryPDFProviderTask.

Table 6. Parameters for Configuring the StatementSummaryPDFProvider Task

Parameter	Description
Bill ready PDF store location	The directory where the PDF files are stored in the local file system or on a content management server. The StatementSummaryPDFProvider job generates subfolders called /Date/AccountNumber/ under this directory, and it places the PDF files in the appropriate subfolder. This path must not contain any spaces.

StatementSummaryDaisyProvider Job

The StatementSummaryDaisyProvider job sends an email with a statement in DAISY audio file to enrolled users whose accounts have new statements ready in Oracle Self-Service E-Billing, and who have chosen to receive DAISY audio file statements. Individual users must request that a customer service representative enable their interface with the DAISY file option.

This job uses information passed from the ETL process and finds all users whose account statements have recently been loaded and processed. The job sends email with a DAISY statement summary as a ZIP file attachment to the corresponding users. Email messages are sent only after running the Notifier job.

After emailing DAISY ZIP files with the Notifier job, the administrator can purge the folder containing the DAISY files.

If the job does not successfully generate the DAISY audio file, then you can check the log file. For details about configuring the log file for the StatementSummaryDaisyProvider job, see *Installation Guide for Oracle Self-Service E-Billing*.

NOTE: The StatementSummaryDaisyProvider job and the StatementReady job can run independently.

Parameters for Configuring StatementSummaryDaisyProviderTask

Table 7 shows the configuration parameter for StatementSummaryDaisyProviderTask.

Table 7. Parameters for Configuring StatementSummaryDaisyProviderTask

Parameter	Description
Bill ready DAISY store location	The directory where the DAISY audio files are stored in the local file system or on a content management server. The StatementSummaryDaisyProvider job generates subfolders called /Date/AccountNumber/ under this directory, and it places the DAISY files in the appropriate subfolder. This path must not contain any spaces.

BatchReportScheduler Job

The BatchReportScheduler job lets you schedule batch reports with recurring batch report requests due to run. You can configure the job to include requests that are due during a period of up to five days, or 120 hours, in the future.

Scheduled batch reports are displayed on the Scheduled Business or Billing Batch Report until they are processed.

For the recurring batch requests that reach the end of the effective period, the BatchReportScheduler job generates one notification email with the subject Batch Report Request Expiry. These expired recurring batch requests are then removed from the billing or business Scheduled report views.

Parameters for Configuring the BatchReportScheduler Task

Table 8 describes the configuration parameters for the BatchReportSchedulerTask.

Table 8. Parameters for Configuring the BatchReportScheduler Task

Parameter	Description
Ahead of Hours	Select the number of hours in the future, up to 120. The BatchReportScheduler job schedules all batch reports due to run during this period.

BatchReportProcessor Job

The BatchReportProcessor job validates and generates scheduled batch reports and makes them available on the Completed Business or Billing Batch Report. The BatchReportProcessor job processes both one-time and recurring batch requests that have been scheduled by the BatchReportScheduler job. The job processes requests with the oldest scheduled dates first.

If the validation fails for a scheduled batch report request, then the request instance appears on the Failed Business or Billing Batch Report. A request can fail for one of the following reasons:

- The hierarchy parent and child node are not related to the service agreement, such as if the agreement was removed from the hierarchy.
- The user does not have access rights to the hierarchy node.

Schedule the BatchReportProcessor job to run after the ETL load.

Parameters for Configuring the BatchReportProcessor Task

Table 9 shows the configuration parameter for BatchReportProcessor task.

Table 9. Parameters for Configuring the BatchReportProcessor Task

Parameter	Description
Batch report store location	The directory where the batch report files are stored in the local file system or on a content management server. The BatchReportProcessor job generates subfolders called /Date/ under this directory, and it places the batch report files in the appropriate subfolder. This path must not contain any spaces.

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. Run 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 run AllNotifications.

Parameters for Configuring the PasswordExpNotify Task

Table 10 describes the parameters for configuring the PasswordExpNotify task.

Table 10. Parameters for Configuring the PasswordExpNotify Task

Parameter	Description
Number of days to send notification before a password is expired	The number of days to send notifications to a user whose password is set to expire.

EmailBounceBack Job

You create and configure an EmailBounceBack job to flag invalid email addresses. The EmailBounceBack job processes a UTF-8 file generated by your SMTP email server that contains the invalid email addresses of undeliverable notifications and flags the invalid addresses in the appropriate user profiles. The next time a user flagged with an invalid email address logs in, the screen prompts him or her to enter a new email address.

You must configure your SMTP server to generate a data file containing invalid email addresses from undeliverable Oracle Self-Service E-Billing notifications only.

The EmailBounceBack job consists of two tasks:

- Scanner

- ProcessEmailBouncebackFile

Parameters for Configuring the Scanner Task

Table 11 displays the configuration parameters for the Scanner task.

Table 11. Parameters for Configuring the Scanner Task

Parameter	Description
Input File Path	The path where the Scanner can find the input file generated by your SMTP server that contains the invalid email addresses from undeliverable notifications.
Input File Name	The name of the input data file for the task, located in the input file path. The default value is *.* for any file name.
Output File Path	The path where the Scanner moves the input file for use by the ProcessEmailBouncebackFile task and renames it with a timestamp.

Parameters for Configuring the ProcessEmailBouncebackFile Task

The ProcessEmailBouncebackFile task reads the email bounce back file line by line and then updates the corresponding user profiles to indicate that the email address is invalid, then generates two log files. One file contains found email addresses and the other contains email addresses not found.

Table 12 displays the configuration parameters for the ProcessEmailBouncebackFile task.

Table 12. Parameters for Configuring the ProcessEmailBouncebackFile Task

Parameter	Description
Email Bounceback File	The path and name of the output file from the Scanner task.

LoadExternalB2BUsers Job

You create and configure a LoadExternalB2BUsers job to load an input file with the existing or new B2B company users from an external authentication system, such as single-sign, to Oracle Self-Service E-Billing.

The B2B load input file requires the following minimum information, in CSV format:

- User ID
- First Name
- Last Name
- Role
- Email Address

- Company ID

The B2B load input file can contain the following optional fields:

- Preferred Language
- Mobile Phone Number for SMS
- Mobile Carrier for SMS
- Brand. This field determines which CSS file is used.

The following example shows the format for a record in a B2B input file:

```
FSmith,Frank,Smith,Admin,Frank.Smith@AcmeCompany.com,Acme
Company,en_US,123456789,GOOGLE,Default
```

Parameters for Configuring the Scanner Task

Table 13 displays the configuration parameters for the Scanner task.

Table 13. Parameters for Configuring the Scanner Task

Parameter	Description
Input File Path	The path where the Scanner can find the input file with the new B2B users provided by the external single-sign on system.
Input File Name	The name of the input data file for the task, located in the input file path. The default value is *.* for any file name.
Output File Path	The path where the Scanner moves the input file for use by the LoadExternalB2BUsers task and renames the input file with a timestamp.

Parameters for Configuring the LoadExternalB2BUsers Task

The LoadExternalB2BUsers task loads the B2B user output file from the Scanner task.

Table 14 displays the configuration parameters for the LoadExternalB2BUsers task.

Table 14. Parameters for Configuring the LoadExternalB2BUsers Task

Parameter	Description
SSO Bulk Load B2B File	Use the output file from Scanner task. This parameter is preconfigured.
Admin Email Address (;)	The email address of your company's administrator.
Plug-in	Specify the name of the plug-in to use, if applicable.

LoadExternalB2CUsers Job

You create and configure a LoadExternalB2CUsers job to load an input file with the new B2C users from an external authentication system, such as single-sign, to Oracle Self-Service E-Billing.

The B2C load input file requires the following minimum information, in CSV format:

- User ID
- First Name Last Name
- Role
- Email Address
- Biller ID
- Billing Account Number

The B2C load input file can contain the following optional fields:

- Preferred Language
- Brand. This field determines which CSS file is used.
- Mobile Phone Number for SMS
- Mobile Carrier for SMS

The following example shows the format for a record in a B2C input file:

```
AJones,Anne,Jones,User,Anne.Jones@WirelessCompany.com,WC2,7836380WC21,en_US,,,
```

Parameters for Configuring the Scanner Task

Table 15 displays the configuration parameters for the Scanner task.

Table 15. Parameters for Configuring the Scanner Task

Parameter	Description
Input File Path	The path where the Scanner can find the input file with the new B2C users provided by the external single-sign on system.
Input File Name	The name of the input data file for the task, located in the input file path. The default value is *.* for any file name.
Output File Path	The path where the Scanner moves the input file for use by the LoadExternalB2CUsers task and renames the input file with a timestamp.

Parameters for Configuring the LoadExternalB2CUsersTask

Table 16 describes the configuration parameters for the LoadExternalB2CUsers task.

Table 16. Parameters for Configuring the LoadExternalB2CUsers Task

Parameter	Description
Use the output file from Scanner task.	Use the output file from Scanner task. This parameter is preconfigured.
The email address of your company's administrator.	The email address of your company's administrator.
Specify the name of the plug-in to use, if applicable.	Specify the name of the plug-in to use, if applicable.

LoadExternalCSRUsers Job

You create and configure a LoadExternalCSRUsers job to load an input file with the new customer service representative users from an external authentication system, such as single-sign, to Oracle Self-Service E-Billing.

The load input file requires the following minimum information, in CSV format:

- User ID
- First Name Last Name
- Role
- Email Address

The load input file can contain the following optional fields:

- Preferred Language

The following example shows the format for a record in an input file (userName,firstName,lastName,role,email,language):

```
democsradmin,CSRAdmin,SSODemo,CSRAdministrator,liaoliao.long@oracle.com,en_US
```

Parameters for Configuring the Scanner Task

Table 17 displays the configuration parameters for the Scanner task.

Table 17. Parameters for Configuring the Scanner Task

Parameter	Description
Input File Path	The path where the Scanner can find the input file with the new customer service representative users provided by the external single-sign on system.
Input File Name	The name of the input data file for the task, located in the input file path. The default value is *.* for any file name.
Output File Path	The path where the Scanner moves the input file for use by the LoadExternalCSRUsers task and renames the input file with a timestamp.

Parameters for Configuring the LoadExternalCSRUsers Task

Table 18 displays the configuration parameters for the LoadExternalCSRUsers task.

Table 18. Parameters for Configuring the LoadExternalCSRUsers Task

Parameter	Description
Use the output file from Scanner task.	Uses the output file from Scanner task. This parameter is preconfigured.
The email address of your company's administrator.	The email address of your company's administrator.
Specify the name of the plug-in to use, if applicable.	Specify the name of the plug-in to use, if applicable.

DefUsrAcctRelationship Job

The DefUsrAcctRelationship job lets you add or remove user-account relationships in Oracle Self-Service E-Billing (Consumer Edition only). The job processes a pipe-delimited TXT input file with a list of user IDs and the associated billing system ID and billing account, and a flag stating whether this user should be added to or deleted from the account.

If a user has multiple accounts, then the input file will have multiple rows for the same user. For details about the input file format for the DefUsrAcctRelationship job, see *Implementation Guide for Oracle Self-Service E-Billing*.

The DefUsrAcctRelationship job first verifies that the user and billing account exist in the Oracle Self-Service E-Billing database.

If the user-account relationship record is flagged to be removed, then the job checks whether user is associated with any other billing accounts. If the user is associated with more than one billing account, then the association is removed along with any recurring payments. If the user is associated with only one billing account, then the user is not removed. Once you confirm that you want to remove these users, you can delete B2C users and associated activities in batch mode using a SQL script at the command line. For more information about deleting users in batch mode, see "Deleting B2C Users in Batch Mode" on page 154.

After processing, the DefUsrAcctRelationship job sends an email notification to the email addresses specified in the job configuration indicating the number of successful, failed, and warning entries. The job also sends a list of bad billing accounts in an email to the email addresses specified in the job configuration.

Problems with User-Account Records That Generate Bad Entries

The DefUsrAcctRelationship job logs the bad entries, including all errors and warnings, in a log named as the input file name with the extension _Bad, and it places the file in the output directory. The following issues with user-account records are identified as bad entries:

- **Error.** The entry does not conform to the file specification.
- **Error.** The user ID does not exist in the Oracle Self-Service E-Billing database or is not a B2C user ID.
- **Error.** The account does not exist in the Oracle Self-Service E-Billing database or is not a B2C account, and includes billing ID validation.
- **Warning.** The user-account relationship is flagged to be added but already exists in the Oracle Self-Service E-Billing database.
- **Warning.** The user-account relationship is flagged to be removed but does not exist in the Oracle Self-Service E-Billing database.
- **Warning.** The user-account relationship is flagged to be removed but the user has only one account.

The DefUsrAcctRelationship job also records entries that do not conform to the file specification in the LOGS database table, and records all other types of bad entries in EDX_UMF_USERACCT_LOAD_EXCEPN database table.

Parameters for Configuring the DefUserAcctRelationshipTask

Table 19 displays the configuration parameters for the DefUserAcctRelationship task.

Table 19. Parameters for Configuring the DefUserAcctRelationship Task

Parameter	Description
Input File Path	The directory where the input file is stored.

Table 19. Parameters for Configuring the DefUserAcctRelationship Task

Parameter	Description
Input File Name	The directory where the task moves the output file after the job runs successfully.
Email Address separated by ";"	The email addresses to receive the notification when the job is complete.

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. 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 20 describes the parameters for configuring the PurgeLogs task.

Table 20. Parameters for Configuring the PurgeLogs Task

Parameter	Description
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.

ReportCleanUp Job

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

Parameters for Configuring the ReportCleanUpTask Task

Table 21 describes the configuration parameters for the ReportCleanUp task.

Table 21. Parameters for Configuring the ReportCleanUpTask Task

Parameter	Description
Number of Days to Keep Batch Files	The number of days after which Oracle Self-Service E-Billing deletes report files.

Table 21. Parameters for Configuring the ReportCleanUpTask Task

Parameter	Description
Report DB JNDI	Use `edx.report.databasePool`.
Batch report file store location	The directory where the batch report files are stored in the local file system or on a content management server. Use the same value that you specified for the BatchReportProcessor job.

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 the job to run successfully but generate nonstandard output. In this case, the return code of 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 22 displays the configuration parameters for the ShellCmdTask task.

Table 22. Parameters for Configuring the ShellCmdTask Task

Parameter	Description
Shell Command	Your shell command.
Environment Vars	Your environment variables, separated by semicolons.

Hierarchy Jobs

Table 23 lists the hierarchy-related production jobs available with Oracle Self-Service E-Billing.

Table 23. Hierarchy Production Jobs

Job Name	Description
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 "HierarchyImporter Job" on page 41.
HierarchyCopy	Replicates all published hierarchies for the periods up to the current periods. For details on configuring a HierarchyCopy job, see "HierarchyCopy Job" on page 42.
HierarchyCleanUp	Cleans up closed accounts and unsubscribed services, and removes them from hierarchies. For details on configuring a HierarchyCleanUp job, see "HierarchyCleanUp Job" on page 43.
HierarchyPurge	Cleans up closed accounts and unsubscribed services, and removes them from hierarchies. For details on configuring a HierarchyPurge job, see "HierarchyPurge Job" on page 43.

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`. In the directory, *EDX_HOME* is the location where you installed Oracle Self-Service E-Billing.

Parameters for Configuring the Scanner Task

Table 24 describes the configuration parameters for the Scanner task.

Table 24. Parameters for Configuring the Scanner Task

Parameter	Description
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.

Table 24. Parameters for Configuring the Scanner Task

Parameter	Description
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 25 describes the configuration parameters for the HierarchyImporter task.

Table 25. Parameters for Configuring the HierarchyImporter Task

Parameter	Description
XML File	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, then 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.

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, you have 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 2012 that must be populated before your implementation goes live, then you can run the ETL EBILL_DATALOAD and LOAD_PRODUCTION jobs for September 2012, then run HierarchyCopy job for October 2012, to replicate September 2012's hierarchy to October. After the October data has been replicated, October will not show up in the list. You can then run ETL load jobs again to load billing data for October 2012. The ETL jobs will then add additional data to the billing structures that are replicated from September 2012.

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. If you create a new hierarchy and publish periods later than the period for which you have run the HierarchyCopy job, then the job only replicates data for those hierarchies that do not have data published for the replicating period.

Parameters for Configuring the HierarchyCopy Task

Table 26 describes the configuration parameters for the HierarchyCopy task.

Table 26. Parameters for Configuring the HierarchyCopy Task

Parameter	Description
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 HierarchyCopy 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 HierarchyCopy job has been run for the current period.

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 determines whether 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, then 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, then 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.

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, or the threshold value.
- Records for the previously mentioned deleted or expired hierarchies.

Parameters for Configuring the HierarchyPurge Task

Table 27 describes the configuration parameters for the HierarchyPurge task.

Table 27. Parameters for Configuring the HierarchyPurge Task

Parameter	Description
Number of Periods	The number of periods that hierarchy nodes or hierarchies have been expired.

Payment Jobs

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 28 lists the available Payment Jobs.

Table 28. Payment Production Jobs

Job Name	Description
pmtRecurringPayment	Schedules payments on a recurring basis, as defined by the user. For details on configuring a pmtRecurringPayment job, see “pmtRecurringPayment Job” on page 46.
pmtSubmitEnroll	Submits enrollment information to a payment gateway. Currently this field applies to the Automated Clearing House (ACH) payment gateway only. For details on configuring a pmtSubmitEnroll job, see “pmtSubmitEnroll Job” on page 49.
pmtConfirmEnroll	Activates pending payment accounts after three days, by default, as long as there has been no error returned. Applies to the ACH payments only. For details on configuring a pmtConfirmEnroll job, see “pmtConfirmEnroll Job” on page 50.

Table 28. Payment Production Jobs

Job Name	Description
pmtPaymentReminder	Sends payment reminder email notifications to customers. For details on configuring a pmtPaymentReminder job, see "pmtPaymentReminder Job" on page 50.
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 "pmtCheckSubmit Job" on page 52.
pmtCheckUpdate	Updates a check's status according to the response from a payment gateway. For ACH payments the job also processes check returns, prenote returns and NOC returns. For details on configuring a pmtCheckUpdate job, see "pmtCheckUpdate Job" on page 55.
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 "pmtARIntegrator Job" on page 59.
pmtAllCheckTasks	Runs a sequence of Payment jobs automatically, in the required order. For details on configuring a pmtAllCheckTasks, see "pmtAllCheckTasks Job" on page 61.
PaymentDueNotification	Sends email to the account holder notifying him or her of the current account balance due before a configured number of days. For details on configuring a PaymentDueNotification job, see "PaymentDueNotification Job" on page 61.
pmtCreditCardSubmit	Submits scheduled credit card payments to a credit card gateway. For details on configuring a pmtCreditCardSubmit job, see "pmtCreditCardSubmit Job" on page 62.
pmtPaymentRefund	Processes the refund payment after a customer service representative user or an administrator authorizes a refund. For details on configuring a pmtPaymentRefund job, see "pmtPaymentRefund Job" on page 64.
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 "ThresholdExceedNotify Job" on page 69.

Table 28. Payment Production Jobs

Job Name	Description
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 "pmtCreditCardExpNotify Job" on page 69.
pmtCustom	Use to create a custom job. For details on configuring a pmtCustom job, see "pmtCustom Job" on page 69.
pmtNotifyEnroll	Sends email to Oracle Self-Service E-Billing users whose bank account registration information has been verified by the ACH and updates the payment account status. For details on configuring a pmtNotifyEnroll job, see "pmtNotifyEnroll Job" on page 70.

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, then this job sets the recurring payment to inactive.

If the customer selects a payment to be made for bills currently due within the specified number of days before the due date, or selects the amount to be the due amount, then 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

Parameters for Configuring the RecurPaymentSynchronizerTask

Table 29 describes the configuration parameters for the RecurPaymentSynchronizer Task.

Table 29. Configuration Parameters for the RecurPaymentSynchronizer Task

Configuration Parameter	Description
Implementation of interface IRecurringPaymentPlugIn.	The name of the Java class that is called before the pmtRecurringPayment job schedules a payment. This class currently does nothing, but you can replace this class with one of your own to process additional business logic and possibly modify how the payment is scheduled, or to cancel the payment. 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 your Oracle sales representative to request assistance from Oracle's 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, the default, enables synchronization. To ignore synchronization and start scheduling immediately, change the value to Y.

Parameters for Configuring the RecurPaymentSchedulerTask

Table 30 describes the configuration parameters for the RecurPaymentScheduler task.

Table 30. Parameters for Configuring the RecurPaymentScheduler Task

Parameter	Description
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 your Oracle sales representative to request assistance from Oracle's Professional Services. The default value for this plug-in is: `com.edocs.common.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 canceled. 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 dependent on this condition.

Table 30. Parameters for Configuring the RecurPaymentScheduler Task

Parameter	Description
Cancel recurring payment if payment account is invalid?	The account information, contained in a prenote for the ACH payments, payment gateway sent to the Originating Depository Financial Institution (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, then 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, then the account will be marked invalid. Because the customer's enrollment failed, the customer 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 canceled.

pmtSubmitEnroll Job

The ACH optionally accepts enrollment information to verify the customer's check routing number and check account number. The 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 payments 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 63.

The pmtSubmitEnroll job has only one task, called pmtSubmitEnroll.

Parameters for Configuring the SubmitEnrollTask

Table 31 describes the configuration parameters for the SubmitEnroll Task.

Table 31. Parameters for Configuring the SubmitEnroll Task

Parameter	Description
Skip Holidays	Determines whether to send the ACH payment batch file to the ACH 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.

pmtConfirmEnroll Job

The pmtConfirmEnroll job only applies to ACH payments. The pmtConfirmEnroll job checks the txn_date field in the payment_accounts table to find each new account, where the account_status field is pnd_wait. If the specified number of days have passed since the user signed up for enrollment (txn_date), then the pmtConfirmEnroll job 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 ACH payments. There are no configuration parameters for the pmtConfirmEnroll job.

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 value is Returned, Failed, or Processed. The Processed status indicates a payment was sent for processing. After sending the email, the Reminded field is updated to Y.

  The check status values are shown in Table 32.

Table 32. Check Status Values

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 33 describes the parameters you configure for the PaymentReminder task.

Table 33. Parameters for Configuring the PaymentReminder Task

Parameter	Description
Implementation of Interface IPaymentReminderPlugin	Optional class that modifies the IPaymentReminderPlugin plug-in. You can modify whether a payment reminder is sent and other actions dependent on the type of reminder. For information about implementing this class, contact your Oracle sales representative to request assistance from Oracle's 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, or cleared.

Table 33. Parameters for Configuring the PaymentReminder Task

Parameter	Description
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.
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.

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.

The pmtCheckSubmit job 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, then the check's account information is used for check submission.

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

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

For ACH payments, 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 following directory:

- **UNIX.** *EDX_HOME*/payment/lib/payment_resources/ach/template/
- **Windows.***EDX_HOME*\payment\lib\payment_resources\ach\template

In the path, *EDX_HOME* is the directory where you installed Oracle Self-Service E-Billing. For help modifying ACH batch format, contact your Oracle sales representative to request assistance from Oracle's Professional Services.

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

Table 34 shows the fields that the pmtCheckSubmit job updates in the check_payments table.

Table 34. Columns Updated in the Check Payments Table by the pmtCheckSubmit Job

Column	Description
last_modify_time	Current time
status	7
action_code	For ACH payments: 27 for checking and 37 for saving.
txn_number	For ACH payments: 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 the ACH to a check in the Payment database. For more information, contact your Oracle sales representative to request assistance from Oracle's Professional Services.

About 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.

The ACH is closed on bank holidays, but it is allowed to send a file to the ACH which requires transfer of money on holidays. The 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 the 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. Otherwise, the batch entry effective date 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, then modify the Payment Settings.

The CheckSubmit job consists of the CheckSubmit task.

Parameters for Configuring the CheckSubmitTask

Table 35 describes the configuration parameters for the CheckSubmitTask.

Table 35. Parameters for Configuring the CheckSubmit Task

Parameter	Description
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, then all the checks whose pay dates are tomorrow or earlier will be selected to send to the ACH. 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 canceled. Normally, the value is Y. Use N if the plug-in is going to take actions dependent on this condition.
Cancel payment if account information is invalid?	The account information, contained in a prenote for the 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, the customer 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, then 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 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.

Table 35. Parameters for Configuring the CheckSubmit Task

Parameter	Description
Do you want to create ACH records in separate lines?	Indicates whether to create ACH output in separate lines. The values can be Y-Yes or N- No. The default is N.
Length of ACH Record (used for line separator)	Indicates the number of characters after which to create a new ACH record line. The default is 94. This parameter is applicable only when you set the parameter for creating ACH records in separate lines to Y.

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) Payment Gateway Logic

The pmtCheckUpdate job processes return files from the ACH, which can include checks from multiple DDNs, or applications. 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. If the names are the same, then the pmtCheckUpdate job continues processing the ACH return file. If the names are not the same, then the job ignores the check.

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

For each successful batch header record match, the pmtCheckUpdate job updates the status according to Table 36.

Table 36. Status Changes with the pmtCheckUpdate Job

Status Change	Status Code	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.	Check error

The pmtCheckUpdate job also updates the status to Paid if there is no return from the 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, according to the ACH Template Directory parameter.

About ACH Return Types

One ACH 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 37 lists the columns that are updated in the check_payments table after a check return is processed.

Table 37. 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 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 in the account_status column, and txn_message is set to the ACH return code.

Table 38 lists the columns that are updated in the payment_accounts table after a prenote is processed.

Table 38. 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, or Y in the Payment Settings, then the corresponding payment routing number, account number, or account type is updated dependent on the NOC code. If the flag is false, or 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 the ACH. This is the content from position 36 to 64 of the 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 the notify_status field 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 39 lists the columns that are updated in the payment_accounts table after a NOC is processed.

Table 39. 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
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 40 must return correctly.

Table 40. 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.

Table 40. ACH Return File Format

Column Name	Record	Description
addendaTypeCode	addenda	The values can be one of the following: ■ 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 the 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.
totalDebitAmount	fileTrailer	Total debit amount in the file.
totalCreditAmount	fileTrailer	Total credit amount in the file.

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 the PaymentIntegratorTask Task

For the PaymentIntegratorTask task, configure the parameters described in Table 41.

Table 41. Parameters for Configuring the PaymentIntegratorTask

Parameter	Description
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 for 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.

Table 41. Parameters for Configuring the PaymentIntegratorTask

Parameter	Description
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.
Directory for Output AR File	Specify a directory to put the output file, *EDX_HOME*/payment/Output/AR, or another location.
Transform Output AR File to Another Format?	A flag, Y or N. This parameter is ignored unless an XML template is chosen. If the value is Y, then 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 dependent 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. In the path, *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.

Table 41. Parameters for Configuring the PaymentIntegratorTask

Parameter	Description
Flexible Parameter 1	Optional parameter used in the AR query.
Flexible Parameter 2	Optional parameter used in the AR query.

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:

- “Parameters for Configuring the RecurPaymentSynchronizerTask” on page 47
- “Parameters for Configuring the RecurPaymentSchedulerTask” on page 48
- “Parameters for Configuring the SubmitEnrollTask” on page 49
- “Parameters for Configuring the PaymentReminderTask” on page 51
- “Parameters for Configuring the PaymentIntegratorTask Task” on page 59
- “Parameters for Configuring the CheckSubmitTask” on page 54

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

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 42 displays the configuration parameters for the PaymentDueNotification task.

Table 42. Parameters for Configuring the PaymentDueNotification Task

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

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, or 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, then the payment is approved, money is guaranteed to be transferred, and the status of the payment is set to Settled, or 8. If there is a problem submitting the payment, then its status is set to Failed-authorize, or -4.

The Payment module supports the PayPal Payflow Pro and uses HTTP to communicate with it. If there is a network problem, then 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, then the Payment module stores the confirmation number from PayPal Payflow Pro in the txn_number field of the creditcard_payments table.

Table 43 describes the columns in the CHECK_PAYMENTS table updated after a credit card is submitted.

Table 43. 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. The status remains Scheduled if there is a network error.
log_id	The value in payment_log, which represents a report ID.

Parameters for Configuring the CreditCardSubmitTask

Table 44 describes the configuration parameters for the CreditCardSubmitTask.

Table 44. Parameters for Configuring the CreditCardSubmit Task

Parameter	Description
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?	Y or N. If the credit card account used in a payment has expired, then cancel the payment. If you specify N, then the task tries to make the payment using the old account. If the task fails to make the payment using the old account, then 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, or 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, then schedule your job run twice, back to back.

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 canceled only when a payment has a status of Scheduled. If a payment is in another state, then the payment cannot be canceled.
- **Voids.** If a payment has a status of Authorized, then it can be voided. An authorized payment must not be canceled 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 45 describes the configuration parameters you must specify in the XML file for batch refunds.

Table 45. Parameters for Configuring the Batch Refund XML File

Parameter	Description
Payee Id	DDN Name of the Biller. This parameter is required.
User Id	The user's registration ID used to identify the user. This parameter is required.
Payment Account Number	The user's credit card or checking account number where the refund must be made. This parameter is required.
Payer Account Number	The account number that the user is paying for. This parameter is required for refunds.
Amount	The amount to be refunded. If the Amount tag does not contain an amount, then the job refunds the original amount of the payment. This parameter is optional.
Payment Type	The refund payment mode for credit card. This parameter is required.
Refund Type	Refund type R–Refund. This parameter is required.
Billing System Transaction Number	The transaction number, or payment ID, provided by Oracle Self-Service E-Billing. This parameter is required.

Table 45. Parameters for Configuring the Batch Refund XML File

Parameter	Description
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. This parameter is optional.
Source System Payment Initiator	The ID of the refund payment initiator and is used for security. This parameter is optional.
Transaction Description	The description for the refund justification. This parameter is optional.
Notify Required	Whether a notification is required for the refund. Notification can be sent to the biller as well as the payer. This parameter is optional.
Flex Field One	Flexible field for storing custom data. This parameter is optional.
Flex Field Two	Flexible field for storing custom data. This parameter is optional.
Flex Field Three	Flexible field for storing custom data. This parameter is optional.
Flex Field Date	Flexible field for storing custom data. This parameter is 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 45 on page 65.

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, then the updated refund record is written to the database with the status as Scheduled. If the plug-in rejects the refund, then 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 46 describes the parameters you must configure for the LoadsRefunds task.

Table 46. Parameters for Configuring the LoadsRefunds Task

Parameter	Description
Payment selection criteria for refunds	Select the unit of time in days, weeks, or months, to use for payment refund criteria.
Selection criteria value (Greater than 0 value)	Enter a specific number in 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, then 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 canceled immediately.

  If a scheduled payment is canceled, then Oracle Self-Service E-Billing processes it. If the status of the payment is Scheduled, then 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, then 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 payment status is in the Authorized state.

  If the status is in the Authorized state, then 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, then the refund status updates to Rejected, and is stored in the database.

  If the plug-in accepts the refund and the ID of the new payment is captured through the refund, then the old payment which is being voided is canceled 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, then 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, or 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, then 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 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, then Oracle Self-Service E-Billing changes the refund status to Rejected.

Parameters for Configuring the SubmitRefund Task

Table 47 describes the parameters you must configure for the SubmitRefundTask.

Table 47. Parameters for Configuring the SubmitRefund Task

Parameter	Description
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.

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.

pmtCreditCardExpNotify Job

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

Parameters for Configuring the CreditCardExpNotifyTask Task

Table 48 describes the configuration parameters for the CreditCardExpNotifyTask task.

Table 48. Parameters for Configuring the CreditCardExpNotifyTask Task

Parameter	Description
Number of days before card expiration date to send email	Specifies the number of days before the customer's 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 your Oracle sales representative to request assistance from Oracle's Professional Services. The default value of the pmtCreditCardExpNotify job is:

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

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.

pmtNotifyEnroll Job

The pmtNotifyEnroll job sends email to Oracle Self-Service E-Billing users whose bank account registration information has been checked by the ACH, and updates the payment account status. If the bank account information is valid, then the payment account status is recorded as Active. If the account information is invalid, then the account status changes to bad_active.

Email messages are sent only after running the Notifier job.

The pmtNotifyEnroll job has one task, NotifyEnroll, which has no configurable parameters.

Payment Consolidator Jobs

Configuring the payment consolidator jobs are steps in "Process of Configuring a Payment Consolidator" on page 112.

Table 49 lists the jobs necessary for using a payment consolidator with Oracle Self-Service E-Billing.

Table 49. Payment Consolidator Production Jobs

Job Name	Description
PCAccountEnrollment	Processes the Enrollment Data File provided by a payment consolidator to bring account information into Oracle Self-Service E-Billing for B2C customers who successfully enrolled or unenrolled to use a payment consolidator. The PCAccountEnrollment job can produce an enrollment response file with details on the accounts that were processed successfully. For details on configuring a PCAccountEnrollment job, see "PCAccountEnrollment Job" on page 71.
PCBillSummary	Generates a file with bill summary information for users who have successfully enrolled to use a payment consolidator with a particular biller. For details on configuring a PCBillSummary job, see "PCBillSummary Job" on page 72.
PCBillSummaryAcknowledgement	Processes the bill summary acknowledgement, or confirmation, file received from a payment consolidator, updating the summary records accepted and rejected by the consolidator into the Oracle Self-Service E-Billing database. For details on configuring a PCBillSummaryAcknowledgement job, see "PCBillSummaryAcknowledgement Job" on page 74.

PCAccountEnrollment Job

You use the PCAccountEnrollment job to bring account information into Oracle Self-Service E-Billing for B2C customers existing who successfully enrolled or unenrolled to use a payment consolidator, or who changed their account information. The job uses an XML template to read the format of a consolidator's enrollment file and map it to the corresponding data in Oracle Self-Service E-Billing.

The PCAccountEnrollment job can be configured to produce an enrollment response file with details on the accounts that were processed successfully. The format of the enrollment response file is also determined using an XML template.

The PCAccountEnrollment job consists of the following tasks:

- Scanner
- AccountEnrollment
- EnrollmentResponse

If account information is rejected by the biller or consolidator, or if errors are generated by either the AccountEnrollment or EnrollmentResponse task, then an email notification is automatically sent to all administrators. Each task also generates a log file which is linked in the notification. For information about configuring an alert profile to narrow the list of administrators who receive email notifications from this and other payment consolidator jobs, see "Creating Alert Profiles" on page 91.

Parameters for Configuring the Scanner Task

The Scanner task reads the consolidator's input enrollment file and it places it in the output file location configured for this task.

Table 50 describes the configuration parameters for the Scanner task.

Table 50. Parameters for Configuring the Scanner Task

Parameter	Description
Input File Path	The name of the directory where the consolidator's input enrollment data file is located. You must place the input file in this location before running the PCAccountEnrollment job.
Input File Name	The name of the input data file for the task, located in the input file path. The default value is *.* for any file name.
Output File Name	The name of the directory where the you want the scanner task to place the consolidator's enrollment file.

Parameters for Configuring the AccountEnrollment Task

The AccountEnrollment task uses consolidator's enrollment file from the output file location and updates the appropriate data in Oracle Self-Service E-Billing.

Table 51 describes the configuration parameters for the AccountEnrollment task.

Table 51. Parameters for Configuring the AccountEnrollment Task

Parameter	Description
Input File Type	Select the file type for the consolidator's enrollment input file. The file type determines which template the task uses to read the file from the Payment Consolidator's configuration.

Parameters for Configuring the EnrollmentResponse Task

The EnrollmentResponse task generates the enrollment response output file which is provided to the consolidator.

Table 52 describes the configuration parameters for the EnrollmentResponse task.

Table 52. Parameters for Configuring the EnrollmentResponse Task

Parameter	Description
Generate Response File	Indicates whether to generate an enrollment response file. The value can be: Y - Yes or N - No. If you specify N, then the Enrollment Response task does not run.
Output File Type	Select the type of output file for the consolidator. The file type determines which template the task uses to generate the file from the Payment Consolidator's configuration.
Response File Drop Location	Specify the directory where the task places the enrollment response output file.
Response File Extension	Specify the extension to use for the output file.

PCBillSummary Job

You use the PCBillSummary job to generate a file with bill summary information for users who have successfully enrolled to use a payment consolidator. The file is provided to the payment consolidator. The output content and file format are determined by an XML template.

The summary file includes all active, enrolled accounts whose statement load date is in the time period specified in the job configuration. You can limit the bill summary to specific accounts only by creating a CSV file with the biller and account IDs and specifying the file name in the job configuration.

The job configuration also lets you choose to regenerate multiple bill summaries within a billing period, which lets you rerun the job if an error occurs. The PCBillSummary job only generates the bill summary for accounts that enrolled with a biller using the payment consolidator's site. The PCBillSummary job consists of the BillSummary task.

If an error occurs while generating a summary record, then the BillSummary task automatically sends an email notification to all administrators. This job also generates a log file, which is linked in the notification. For information about configuring an alert profile to narrow the list of administrators who receive email notifications from this and other payment consolidator jobs, see "Creating Alert Profiles" on page 91.

Parameters for Configuring the BillSummary Task

Table 53 describes the configuration parameters for the BillSummary task.

Table 53. Parameters for Configuring the BillSummary Task

Parameter	Description
Bill Summary File Type	The bill summary file type. The file type determines which template is used to generate the summary file.
Biller Name	Select the biller to use with this job.
Date Selection Criteria	The billing period date range: Today, Yesterday - Today, Fixed Date, or N days ago - Today. With the Fixed Date option, you must specify the from and to dates in the MM/dd/yyyy format.
From Date: MM/DD/YYYY	The from date, if you selected Fixed Date for the Date Selection Criteria parameter.
To Date: MM/DD/YYYY	The to date, if you selected Fixed Date for the Date Selection Criteria parameter.
N Days Ago - Today	If using the N Days Ago - Today range, then specify the number of days: 1-31.
List of Accounts: CSV File	To limit the bill summary to specific accounts only, specify the biller and account IDs in a CSV file and specify the file name here.
Send Duplicate	Indicates whether to generate and send a statement summary multiple times within the same billing period: ■ **Y**. Generate the bill summary for all active enroll accounts when the job runs. ■ **N**. Generate the bill summary only for the accounts whose bill summary in the period has not been generated yet. N is the default.
File Drop Location	The name of the directory where the task places the bill summary output file.
File Extension	The name of the extension to add to the bill summary output file.

PCBillSummaryAcknowledgement Job

You use the PCBillSummaryAcknowledgement job to process the bill summary acknowledgement file received from a payment consolidator. The PCBillSummaryAcknowledgement job uses the relevant consolidator XML template to read the bill summary acknowledgement file and update the bill summary records that were accepted and rejected by the consolidator into Oracle Self-Service E-Billing.

The PCBillSummaryAcknowledgement job consists of the following tasks:

- Scanner
- BillSummaryAcknowledgement

If the PCBillSummaryAcknowledgement job generates errors when processing the bill summary confirmation file, then it automatically sends an email notification to all administrators. This job also generates a log file which is linked in the notification. For information about configuring an alert profile to narrow the list of administrators who receive email notifications from this and other payment consolidator jobs, see “Creating Alert Profiles” on page 91.

Parameters for Configuring the Scanner Task

The Scanner task reads the consolidator’s bill summary acknowledgement file from the input directory configured for this task and it places the file in the output file location with a timestamp in the file name.

Table 54 describes the configuration parameters for the Scanner task.

Table 54. Parameters for Configuring the Scanner Task

Parameter	Description
Input File Path	Name of the directory where the bill summary acknowledgement file is located. You must move the input file to this location before running the job.
Input File Name	The name of the input file for the task, located in the input file path. The default is the wildcard (*.*) for any file name.
Output File Path	The name of the directory where the job places the bill summary acknowledgement file. The output file is used by the next task, BillSummaryAcknowledgement.

Parameters for Configuring the BillSummaryAcknowledgement Task

The BillSummaryAcknowledgement task updates the Oracle Self-Service E-Billing database for bills accepted and rejected by the consolidator. The task also generates a CSV log file with the rejected account information, and sends an email to all administrators regarding the rejected accounts.

Table 55 describes the configuration parameters for the BillSummaryAcknowledgement task.

Table 55. Parameters for Configuring the BillSummaryAcknowledgement Task

Parameter	Description
Input File Type	The type of bill summary acknowledgement input data file provided by the payment consolidator.
Rejected Accounts File Drop Location	The directory where the task places the CSV file with the accounts rejected by the payment consolidator.

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. This chapter includes the following topics:

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 85.
- **Continuous monitoring of job and task status.** For details about monitoring jobs, see “Monitoring Production Jobs” on page 78.
- **Starting a job, using the Run Now feature.** For details see “Starting a Job Manually” on page 83.
- **Monitoring services.** For details on monitoring services, see “Monitoring Service Status” on page 146.

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 is 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 90.

- Check production Error, Information, Warning, and Debug production logs weekly or more frequently. For details on viewing log reports, see "About Message Log Files" on page 145.
- Run job reports to view job activity. For details, see "About Job Reports" on page 143.

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 79.
- Which tasks completed successfully. For details on task status, see "Viewing and Verifying Task Status" on page 80
- 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 82.

Table 56 describes each field on the Command Center Main Console.

Table 56. 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 alphabetically by job name, 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 57 in the order used for ranking purposes.

Table 57. Job Status Values

Job Status	Description
Failed	Job failed
Processing	Job is currently executing

Table 57. 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.
Canceled	Job run failed and was canceled.
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, then the job fails, and the status changes to Failed. All failed jobs display in red on the Main Console. If a job fails, then 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, 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.

Processing order remains unchanged.

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 with 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 58.

Table 58. 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. In this task, N is the maximum number of instances allowed for the job.
- Any instances in Processing, Failed, Reprocessing, or Reprocess status

If you are not using concurrency (N=1), then 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, then 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, then 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, then cancel and run it again.

If the task has not been started, then 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 canceled, 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, then 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. This chapter includes the following topics:

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, and not recur. For information about when to schedule Payment jobs, see "Scheduling Payment Jobs" on page 97.

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 calendar.** 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 88.
- **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 89.
- **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.

   Some jobs require a resource or input data file. In these jobs, the scanner looks for a file one time only. If the file is not there, then the job cannot run. If you are scheduling a job to run one time only, and not recur, then 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, or scan, and the time to stop scan attempts.

   NOTE: To run a job immediately, click Run Now on the Job Schedule page.

4. 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, for the number 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, then the next job process will be skipped, so only that one job process runs at a time.

5 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 88.

6 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, or global alert settings. This is the default setting.
- **Use Alert Group with Alert Type.** Select an alert group and alert type to use for the job, overriding the global alert settings: Success, Failure or both.
- **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.

7 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, then 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.

8 Click Save Schedule.

CAUTION: If you try to save schedule changes while a job is running in the production queue (the job status indicates Processing), then 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 88

2 "Applying a Blackout Calendar to a Job Schedule" on page 89

Creating a Blackout Calendar

To mark days when processing does not occur, 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 87.

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 88

"Deleting a Blackout Calendar" on page 89

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 87.

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, or both.

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

1. “Creating Alert Groups” on page 90
2. “Creating Alert Profiles” on page 91

3 “Applying Alerts to a Job Schedule” on page 93.

Creating Alert Groups

Alert groups let you define a list of contact email addresses 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 89.

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 90

“Deleting Alert Groups” on page 91

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 in the profile.
- **Selected Job Types.** Lets you select a list of specific job types for all applications.
- **Selected DDNs.** Lets you select a list of applications for all job types. Application data definitions are referred to in the Command Center as DDNs.
- **Selected Job Types in a Particular DDN.** Let you choose particular job types for a specific DDN.

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

If you use the payment consolidator feature, then creating an alert profile for the payment consolidator jobs lets you target a list of administrators to receive alerts. Without an alert profile, the payment consolidator jobs automatically send alerts to all administrators. For a list of payment consolidator jobs, see “Payment Consolidator Jobs” on page 70.

This task is a step in “Process of Configuring Job Alerts” on page 89.

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. If you are creating an alert profile for the Payment Consolidator feature, to target a list of administrators to receive alerts, then select this option.

     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. If you are creating an alert for use with the payment consolidator feature, then you must uncheck both Success and Failure alert types for each payment consolidator job. Alert types are:
   - **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, then 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 93

"Deleting an Alert Profile" on page 93

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, which is 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 configured for this job in the alert profiles: success only, failure only, or both.

- Manually enter a list of email contacts and alert types to use, overriding the alert groups and alert types configured for this job in the alert profiles: success only, failure only, or both.
- 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 89.

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.** Use the alert groups and types configured for this job in the alert profiles, or Global alert settings. This is the default.
   - **Use Alert Group with Alert Type.** Select an alert group and alert type to use for the job, overriding the global alert settings: Success, Failure or both.
   - **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. This chapter includes the following topics:

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. ACH is the electronic network for financial transactions in the United States.

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 99.

Credit card payments are supported for immediate or 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. Oracle Self-Service E-Billing supports PayPal Payflow Pro, which processes real-time online authorization and payment.

Using Chase Paymentech Orbital Payment Gateway or Other Payment Processors

The cartridge for Chase Paymentech Orbital Payment Gateway has been tested for use with Oracle Self-Service E-Billing.

You must download the required Chase Paymentech files as part of the application server configuration process. For more information about installing Chase Paymentech Orbital Payment Gateway, see *Installation Guide for Oracle Self-Service E-Billing*.

In addition, you can create cartridges to support other payment processors. For help with cartridges, contact your Oracle sales representative to request assistance from Oracle's Professional Services.

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, then 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 59.

Table 59. Email Notification Jobs

Job	Description
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.
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 are controlled by the email template files. The path to the template files is defined by the Email template status notification parameter in Payment Settings. The default templates are stored in the following directory:

- **UNIX.** `$EDX_HOME/config`
- **Windows.** `%EDX_HOME%\config`

The default email template files must be customized, which is usually done by Oracle Professional Services during the installation. For more information, contact your development team or your Oracle sales representative to request assistance from Oracle's Professional Services.

Configuring a Payment Gateway

You must configure at least one payment gateway for ACH or credit card payments for the biller application that you defined on the Main Console. You can 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 the payment gateway. You must have this information in-hand before configuring.

You must 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 Application Configuration. Click Create for the DDN, or application, you are configuring. Specify an implementation of Implementation of com.edocs.payment.imported.IBillDepot and click Add.
2. Click Global Configuration. Select the payee DDN application. Select the bililng application, and choose the same application from the DDN for External Payment field. Specify the global configuration parameters shown in Table 60 on page 100 and click Update.
3. Click Create next to the check payment type. Select the gateway, either paypal or orbital, and click Add. Specify the payment gateway settings for checks shown in Table 63 on page 105. The parameters in this table are for either PayPal Payflow Pro or Chase Paymentech Orbital Payment Gateway. Click Add.

   If you are migrating, then 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, or credit card, payment type. Select the gateway, either paypal or orbital, and click Add. Specify the credit card gateway settings for the gateway you are using:
   - **PayPal Payflow Pro**. Use the parameters in Table 63 on page 105.
   - **Chase Paymentech Orbital Payment Gateway.** Use the parameters in Table 64 on page 108.
5. Click Add.
6. You must configure at least one payee bank account for each payee DDN. Click Payee Account Configuration, then Click New. Select the biller payee DDN. Specify bank account details for the application. For details on configuring parameters for a payee bank account, see Table 65 on page 111.
7. Click Add to save the settings.

8 Click Payee Biller Configuration, then click New. Choose the Biller ID to map with the payee bank account, the payee application, and the payee bank account to use. Click Add. Repeat this step for any additional Biller IDs you're using.

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

Parameters for Configuring Global Payment Settings

Table 60 describes the global parameters required when configuring a payment gateway for a Billing and Payment application.

Table 60. Parameters for Global Payment Configuration

Parameter	Description
Payee DDNs	Select the application DNN that you defined on the Main Console for use with your Billing and Payment application. In the Command Center, an application is also referred to as a payee Data Definition Name (DDN).
Payment Payee Account Plugin Implementation	Select com.edocs.common.payment.plugin.DefaultPayeeAccountPlugin or select Custom if you're using a custom plugin.
Payment Notification Service Implementation	Always select com.edocs.common.notification.payment.ConsolidatedPaymentNotificationService' class.
DDN for External Payment	This field is for use with the DDN for external payments feature. It is possible to use multiple payee bank accounts for the biller application. For help using the DDN for external payments feature, contact your Oracle sales representative to request assistance from Oracle's Professional Services. Otherwise, leave as the default value.
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.

Check Gateway Configuration Parameters

Table 61 describes the parameters you must specify when configuring an ACH check gateway. These parameters are used for both PayPal Payflow Pro and Chase Paymentech Orbital Payment Gateway.

Table 61. Parameters for Configuring a Check Gateway

Parameter	Description
DDN	The selected application DNN.
Payment Type	check
Gateway	ach
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 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.payment.plugin.DefaultUserAccountAccessor.

Table 61. Parameters for Configuring a Check Gateway

Parameter	Description
Implementation of IPaymentAccountAccessor	The name of the class that handles getting payment user information. Select the class com.edocs.payment.payenroll.payacct.DefaultPaymentAccountAccessor.
Make Authorize Reversal Payments	Y, Yes or N, No
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 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.
Batch Submit Transaction Time Out	The number of minutes after which batch submit transactions time out.
Number of Business Days After Pay Date to Clear the Check	The number of business days for a check to be marked as paid after submitting to the payment gateway without returns or failures. The default is five business days.
Days to Activate Pending Subscribers	The number of business days to wait before approving a customer for online payment. If a prenote file is returned before this time, then the customer is rejected for the causes stated in the prenote file. The default is three days.
Batch File Name Prefix	The text used as a prefix to each batch file name.
Batch File Name Time Stamp	The format used for each batch file time stamp.
Batch File Name Suffix	The text used as a suffix to each batch file name.
ACH Prenote File Name Prefix	The text used as a prefix to each ACH prenote file name.

Table 61. Parameters for Configuring a Check Gateway

Parameter	Description
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, then you must pad the entry with trailing blanks to reach a total length of 10 characters.
Immediate Destination Name	The name of the Biller. This information is assigned to you by your bank, and follows a format specific to the needs of the ODFI. The pmtCheckSubmit job inserts this information into the ACH File Header record's Immediate Destination Name field. The pmtCheckUpdate job uses this value to validate entries in the ACH return file.

Table 61. Parameters for Configuring a Check Gateway

Parameter	Description
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. Oracle Self-Service E-Billing does not create this directory.
ACH Template File Directory	The directory where the ACH XML files are stored.
Implementation of IAchPlugIn	The plug-in allows modification of whether a check payment is submitted, plus other actions dependent 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 value of this parameter is com.edocs.common.payment.cassette.bank.ach.DefaultAchPlugin.
Flexible Field 1 (for plugin)	You can specify an additional field for the IAchPlugin plug in.
Flexible Field 2 (for plugin)	You can specify an additional field for the IAchPlugin plug in.

ACH Federal Holidays

You can configure an ACH check payment gateway to skip nonbusiness 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.

Nonbusiness days in the Payment module include the U.S. Federal holidays listed in Table 62.

Table 62. 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 for PayPal Payflow Pro

Table 63 describes the parameters you must specify when configuring a credit card gateway to PayPal Payflow Pro.

Table 63. Parameters for Configuring a Credit Card Gateway for PayPal Payflow Pro

Parameter	Description
DDN	Name of the DDN
Payment Type	ccard
Gateway	paypal
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 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.

Table 63. Parameters for Configuring a Credit Card Gateway for PayPal Payflow Pro

Parameter	Description
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.payment.plugin.DefaultUserAccountAccessor.
Implementation of IPaymentAccountAccessor	The name of the class that handles getting Payment user information. Select the class com.edocs.payment.payenroll.payacct.DefaultPaymentAccountAccessor.
Make Authorize Reversal Payments	Y, Yes or N, No
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 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.
Gateway Host Name	The URL to the PayPal Payflow Pro host that processes credit card transactions for this payment gateway: pilot-payflowpro.paypal.com

Table 63. Parameters for Configuring a Credit Card Gateway for PayPal Payflow Pro

Parameter	Description
Gateway Host Port	The TCP port number to be used when contacting the PayPal Payflow Pro host. The default is 443.
Gateway 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. If you are using the proxy network connection, then enter your proxy setting. This parameter is optional.
Proxy Port	The port number of the proxy server: 80. If you are using the proxy network connection, then enter your proxy setting. This parameter is optional.
Gateway User	The case-sensitive vendor name from PayPal Payflow Pro. See your PayPal representative for this value. This parameter must match the PayPal Vendor parameter value.
Gateway Password	The case-sensitive password. See your PayPal Payflow Pro representative for this password.
PayPal Vendor	The case-sensitive vendor name assigned by registering with PayPal Payflow Pro. See your PayPal Payflow Pro representative for this value.
PayPal Partner	The partner name used by PayPal: PayPal.
URL StreamHandler class of application server	The name of the class used for the application server StreamHandler. For Oracle WebLogic, use sun.net.www.protocol.https.Handler.
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 enables address verification for credit card payments. AVS support must be set up with PayPal Payflow Pro. Y is the default.

Table 63. Parameters for Configuring a Credit Card Gateway for PayPal Payflow Pro

Parameter	Description
Implementation of IPaypalPlugIn	The plug-in allows modification of whether a credit card payment is submitted, plus other actions dependent on the payments selected for settlement. For example, to deny a credit card payment dependent on additional business rules. For information about implementing this class, contact Oracle Professional Services. The default is: com.edocs.common.payment.cassette.creditcard.DefaultCreditCardPlugin
Flexible Field 1 (for Plug-in)	You can specify an additional field for use with the plug-in. This parameter is optional.
Flexible Field 2 (for Plug-in)	Can specify an additional field for use with the plug-in. This parameter is optional.

Credit Card Gateway Parameters for Chase Paymentech Orbital Payment Gateway

Table 64 describes the parameters you must specify when configuring a credit card gateway for Chase Paymentech.

Table 64. Parameters for Configuring a Chase Paymentech Orbital Payment Gateway

Parameter	Description
DDN	Name of the selected application DDN.
Payment Type	ccard
Gateway	orbital
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 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.

Table 64. Parameters for Configuring a Chase Paymentech Orbital Payment Gateway

Parameter	Description
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.payment.plugin.DefaultUserAccountAccessor.
Implementation of IPaymentAccountAccessor	The name of the class that handles getting Payment user information. Select the class com.edocs.payment.payenroll.payacct.DefaultPaymentAccountAccessor.
Make Authorize Reversal Payments	Y, Yes or N, No
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 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.
Gateway Host Name	The URL to the Chase Paymentech Orbital Payment Gateway host that processes credit card transactions for this payment gateway: orbitalvar1.paymentech.net

Table 64. Parameters for Configuring a Chase Paymentech Orbital Payment Gateway

Parameter	Description
Gateway Host Port	The TCP port number to be used when contacting the Chase Paymentech Orbital Payment Gateway host. Use the default value, 443.
Gateway Timeout Period for Transaction	The number of seconds the pmtCreditCardSubmit job will wait for a transaction to complete with the Chase Paymentech Orbital Payment Gateway host before timing out. Use the default value of 60.
Proxy Address	The address of the proxy server. If you are using the proxy network connection, then enter your proxy setting. This parameter is optional.
Proxy Port	The port number of the proxy server: 80. If you are using the proxy network connection, then enter your proxy setting.This parameter is optional.
Gateway User	The case-sensitive vendor name from Chase Paymentech . See your Chase Paymentech representative for this value.
Gateway Password	The case-sensitive password. See your Chase Paymentech representative for this password.
Orbital Merchant ID	Your Orbital merchant identification number. See your Chase Paymentech representative for this value.
Orbital Terminal ID	Your Orbital terminal ID. See your Chase Paymentech representative for this value.
Orbital BIN	Your Orbital BIN number. See your Chase Paymentech representative for this value.
Enable Orbital Address Verification Service	The value Y enables address verification for credit card payments. AVS support must be set up with Chase Paymentech. Use the default value, Y.
Paymentech Orbital Config File Location	The default location of the Paymentech properties configuration file: *EDX_HOME*\payment\paymentech\config\linehandler.properties
Implementation of IOrbitalPlugIn	The plug-in allows modification of whether a credit card payment is submitted, plus other actions dependent on the payments selected for settlement. For example, to deny a credit card payment dependent on additional business rules. For information about implementing this class, contact Oracle Professional Services. The default is: com.edocs.common.payment.cassette.creditcard.DefaultCreditCardPlugin
Flexible Field 1 (for Plug-in)	You can specify an additional field for use with the plug-in. This parameter is optional.

Table 64. Parameters for Configuring a Chase Paymentech Orbital Payment Gateway

Parameter	Description
Flexible Field 2 (for Plug-in)	Can specify an additional field for use with the plug-in. This parameter is optional.
Number of Threads	Specifies the number of connections to open with the Chase Paymentech Orbital 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. Use the default value of 1.

Parameters for Configuring a Payee Bank Account

Table 65 shows the parameters you configure when adding a new payee bank account.

Table 65. Configuration Parameters for a New Payee Bank Account

Parameter	Description
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.

Table 65. Configuration Parameters for a New Payee Bank Account

Parameter	Description
Vertical Field 1, Vertical Field 2, Vertical Field 3	Fields that can be customized for use with your application. These parameters are optional.
Flexible Field 1, Flexible Field 2, Flexible Field 3	Fields that can be customized for use with your application. These parameters are optional.

Process of Configuring a Payment Consolidator

To enable your consumer customers to pay their bills through a payment consolidator, you must customize and configure Oracle Self-Service E-Billing to receive account enrollment information and provide a summary bill using the particular consolidator's data formats. For information about customizing the preconfigured templates and transmission data files provided by Oracle Self-Service E-Billing, see *Implementation Guide for Oracle Self-Service E-Billing*.

For each consolidator you want to make available to your customers, you must follow this process to configure the payment consolidator, biller, and related production jobs in the Command Center.

To configure a payment consolidator, perform the following tasks:

1. "Registering a Payment Consolidator" on page 112
2. "Registering a Biller" on page 114
3. "Creating a New Job" on page 19. Configure the following jobs:
   - "PCAccountEnrollment Job" on page 71
   - "PCBillSummary Job" on page 72
   - "PCBillSummaryAcknowledgement Job" on page 74
4. "Configuring XMA Settings for Payment Consolidation Jobs" on page 115 (Optional)
5. "Creating Alert Profiles" on page 91. (Optional) Creating an alert profile lets you target a list of administrators to receive alerts after the payment consolidator jobs run. Without an alert profile, the payment consolidator jobs automatically send alerts to all administrators.

Registering a Payment Consolidator

To use a payment consolidator with Oracle Self-Service E-Billing, you must register the payment consolidator and specify the custom templates and other related information in the Command Center.

Templates are used by the corresponding payment consolidation jobs you configure for the consolidator to parse the consolidator's input data file and generate the output file required by the consolidator. You must create custom templates for use with your payment consolidator. For details, see *Implementation Guide for Oracle Self-Service E-Billing*.

You define multiple file types for a consolidator to specify both the input and output templates.

This task is a step in "Process of Configuring a Payment Consolidator" on page 112.

To register a payment consolidator

1. Click Settings on the Command Center Main Menu.
2. Click the Payment Consolidator Settings tab, then click the Consolidator Registration tab.
3. Click Add New Consolidator, enter the following information, and then click Add Consolidator.

Field	Description
Consolidator Name	Specify a unique name for the payment consolidator.
Consolidator ID	Specify a unique ID number for the payment consolidator. This number is used in the file header record and can be a maximum of 32.
Consolidator Alternate Name	Specify an alternate name for the payment consolidator.

4. Click the new consolidator name that appears next to the word Edit.
5. Click New File Type. Configure the enrollment input file for the consolidator by selecting Input File for the File Category field, specifying the appropriate information described in the following table, and clicking Add File Type. Repeat this step, selecting Output File for the File Category and specifying the enrollment output file details for the consolidator. If you intend to use the PCBillSummaryAcknowledgement job with this consolidator, then also define a new file type for that input file.

Field	Description
File Type Name	Specify a name for the consolidator's file type, such as 2040, 2050, and so on. The input file type will appear as an option with the PCAccountEnrollment job and PCBillSummaryAcknowledgement job configurations. The output file type will appear as an option with the PCAccountEnrollment job configuration only.
File Category	Select the type of file you are configuring for the consolidator: ■ Input File ■ Output File
Template File	The full path and name of the template XML file to use for processing the selected file type, input or output, for the consolidator.
Cartridge Plug-in	The name of the implementation class plug-in to use for processing this file type.

Registering a Biller

To use a payment consolidator with Oracle Self-Service E-Billing, you must register your organization as a biller in Oracle Self-Service E-Billing.

This task is a step in "Process of Configuring a Payment Consolidator" on page 112.

To register a biller

1. Click Settings on the Command Center Main Menu.
2. Click the Payment Consolidator Settings tab, then click the Biller Registration tab.
3. Click Add New Biller, enter the following information, and then click Add Biller.

Field	Description
Biller Name	The name of the biller.
Biller ID	A unique ID number for the biller, assigned by the payment consolidator.
Billing System ID Plug-in	The name of the billing system plug-in that maps the Billing System ID defined by the biller to the Biller ID defined in Oracle Self-Service E-Billing.
Biller Alternate Name	An alternate name for the biller.
E-Bill URL	The billing detail URL to display to the consumer, in the following format: https://*hostname*:ebillingAppPort/ebilling/login.do?redirectTo=statementSummary For example: https://localhost:7030/ebilling/login.do?redirectTo=statementSummary
Teaser URL	The URL where the marketing teaser ad is stored. This URL is used to display the teaser ad on the user interface.
Teaser Ad Image URL	The URL where the teaser ad image is stored. This URL is used to display the image on the user interface.
Teaser Ad Text	A text URL that can be used instead of the image.
E-Bill Category URL	The URL associated with the E-bill's category.
E-Bill Category Text	The text associated with the E-bill's category.

Configuring XMA Settings for Payment Consolidation Jobs

The paymentConsolidator.xma.xml file controls how to generate the summary or enrollment response file. You can configure the following parameters for the payment consolidation jobs:

- **logFilePath.** The customized log file location. By default, log files are stored in the following location:
  - **UNIX.** *EDX_HOME*/Data/*applicationName*/*jobName*
  - **Windows.** *EDX_HOME*\Data*applicationName**jobName*
- **batchSize.** The maximum number of records in a batch in a Bill Summary or Enrollment Response file.
- **trunkSize.** The total number of accounts that each job can process at a time. This parameter impacts job performance.

This task is a step in "Process of Configuring a Payment Consolidator" on page 112.

To configure XMA settings for payment consolidation jobs

1. Open the paymentConsolidator.xma.xml file, located in the following directory:
   - **UNIX.** *EDX_HOME*/xma/config/modules/paymentConsolidator
   - **Windows.** *EDX_HOME*\xma\config\modules\paymentConsolidator
2. Configure the parameters found in the following file content:

```
<bean id="configBean" class="com.edocs.common.pc.core.ConfigBean"
scope="prototype">

    <property name="logFilePath">

       <value> </value>

    </property>

    <property name="billSummaryCartridge">

       <ref local="billSummaryCartridgeImp"/>

    </property>

    <property name="billSummaryAckCartridge">

       <ref local="billSummaryAckCartridgeImp"/>

    </property>

    <property name="userAcctEnrollCartridge">

       <ref local="userAcctEnrollCartridgeImp"/>

    </property>
```

```
    <property name="enrollResponseCartridge">
       <ref local="enrollResponseCartridgeImp"/>
    </property>
</bean>


 <bean id="billSummaryCartridgeImp"

class="com.edocs.common.pc.core.cartridge.billsummary.BillSummaryCartridge"
      scope="prototype">
    <property name="batchSize">
       <value>300</value>
    </property>
    <property name="trunkSize">
       <value>500</value>
    </property>
 </bean>
 <bean id="billSummaryAckCartridgeImp"

class="com.edocs.common.pc.core.cartridge.billsummary.BillSummaryConfirmationCa
rtridge"
      scope="prototype">
    <property name="batchSize">
       <value>500</value>
    </property>
    <property name="trunkSize">
       <value>500</value>
    </property>
 </bean>
 <bean id="enrollResponseCartridgeImp"
class="com.edocs.common.pc.core.cartridge.enrollment.EnrollmentResponseCartridg
e" scope="prototype">
      <property name="trunkSize">
```

```
        <value>100</value>
         </property>
</bean>
```

8 Payment Transactions

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

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 a 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.

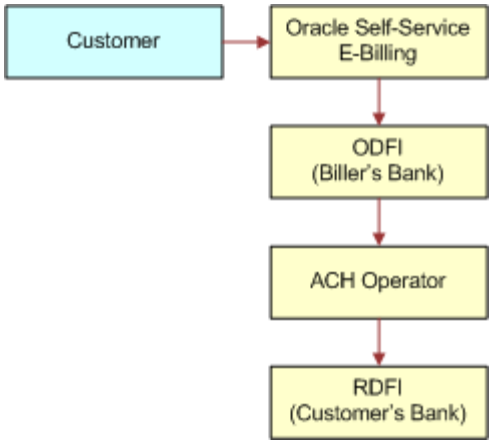

Figure 1. Check Payment Transactions

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, then the check status is changed to Canceled, but the payment remains in the database for the customer to view as a canceled 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, then 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, then the 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 have passed, then the pmtCheckUpdate job changes the status of the check from Processed to Paid.
- If the payment gateway returns the check, then the pmtCheckUpdate job updates the check 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, then the pmtCheckUpdate job 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 the pmtPaymentReminder job is configured, then it sends an email to the customer about the status of the check payment.

Supported SEC Codes

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

- **Web.** Internet Initiated Entry. This is the default.

  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 using ANSI ASC X12 standards. Can be used either with the credit or debit application.

ACH Change Codes

Table 66 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 66. 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 67 lists some of the ACH return codes that can appear in the returns file after running the pmtCheckUpdate job.

Table 67. 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. These are 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. These are 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 nonbusiness 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 nonbusiness 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 nonbusiness 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 whether it is on or before the end of today. If it is, then the computed date is the customer-scheduled date plus one. If it is not, then the computed date is the customer-scheduled date.

Nonbusiness 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, then all the DDNs must use the same template. If you need to use different templates, then 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.

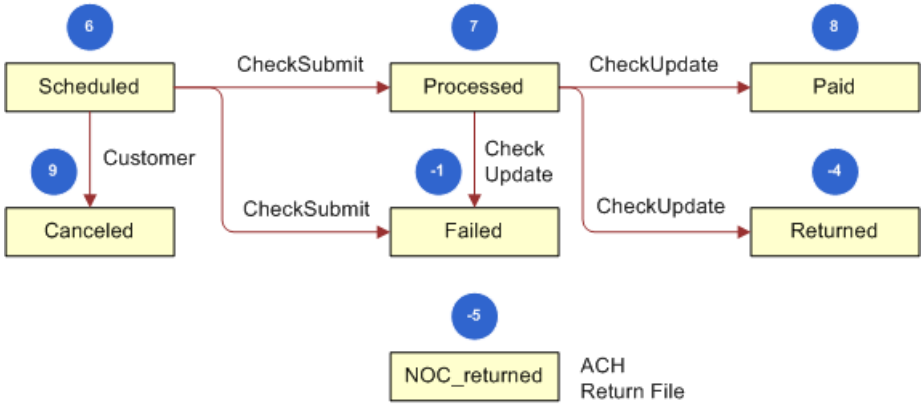

Figure 2. Check Payment Status Flow

Table 68 lists the status values that can occur during a check payment transaction cycle. The code is the actual value saved in the Oracle Self-Service E-Billing database.

Table 68. Status Values for Check Payment Transactions

Transaction Status	Code	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.
Canceled	9	The customer canceled 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, or 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.

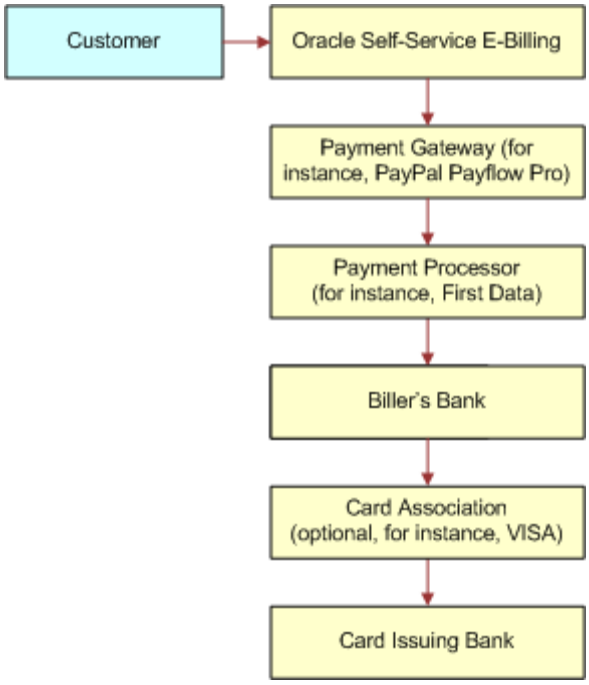

Figure 3. Credit Card Payment Transactions

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, then the state is set to Settled.
- If the card failed to authorize, then the state is set to Failed_authorize.
- If there is a network problem, then the state is set 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 ICreditCardPlugIn has been implemented in Payment Settings, then 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. Thfe 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, then the state is set to Settled.
- If the card fails to authorize, then the state is set to Failed_authorize.
- If there is a network problem, then 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 69 lists the status values that can occur during a credit card payment transaction cycle. The code is the actual value saved in the Oracle Self-Service E-Billing database.

Table 69. Credit Card Payment Transaction Cycle

Transaction Status	Code	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.
Canceled	9	A credit card payment was canceled 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. 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, such as PayPal Payflow Pro, for address comparison. If AVS is turned on by an Oracle Self-Service E-Billing administrator, then 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 to PayPal Payflow Pro indicating whether the information matched. The 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, or voids, the transaction depending on the filter set in the Payment module for both street address and zip code. There is also a filter option to set the international AVS code to determine whether 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, then 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, then 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 depending 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 merchant also must specify to PayPal Payflow Pro during setup and that the customer 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 canceled, when the check is processed, and other check actions.

The Payment module also logs additional important information, warnings, and errors. The format of the logging messages determines whether the Payment module can connect with other monitoring software, such as 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 in 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. This chapter includes the following topics:

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, depending on the amount and pay date. To configure recurring payments, see “pmtCreditCardSubmit Job” on page 62.

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 the 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, which is when the 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, then 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, then 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.

About the Multiple Recurring Payment Feature

The multiple recurring 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 the 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 70 shows the columns updated in the recurring_payments table by the pmtRecurringPayment job.

Table 70. 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, or inserted into the check_payments or creditcard_payments columns. This parameter 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 using 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 one of the following actions:

- The recurring payment is configured to pay a fixed amount
- The recurring payment is configured to pay if less than this amount
- The recurring payment is configured to 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 or creditcard_payments 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 04/09/2012, 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 04/10/2012, and the end date is 06/10/2012. The following table gives an example of a recurring payment created with Amount Due and Before Due Date.

Column Name	Value
payer_account_number	act1111
bill_scheduled	Y
status	Active
last_process_time	04/10/2012. 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.

Column Name	Value
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 on 03/10/2012. The doc ID is bill1, in this example. On 04/10/2012, the ETL load runs again and indexes two more bills: bill2 and bill3. The following table gives an example of ETL load producing bills, and 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	03/10/2012	100.01	04/15/2012
acct1111	bill2	04/10/2012	50.00	04/25/2012
acct1111	bill3	04/10/2012	100.00	05/15/2012

3 The pmtRecurringPayment job runs on 04/10/2012 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 04/10/2012 the value of last_process_time, and 04/10/2012 23:59:00PM, the 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. The last bill info can be retrieved from payment_bill_summaries using the bill_id. Otherwise, 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.

Column Name	Value
last_process_time	04/10/2012 23:59:00PM. This value changes to the job run time.
last_pay_date	01/01/1970, unchanged
next_pay_date	05/14/2012. This value is one day before the due date, 05/15/2012.
bill_id	bill3

6 If the pmtRecurringPayment job runs between 04/11/2012 and 05/10/2012, then nothing happens to this recurring payment because synchronization and scheduling does not happen. The table remains unchanged.

7 On 05/11/2012 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 05/14/2012, which is 05/11/2012 + three 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 05/14/2012. 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	04/10/2012 23:59:00PM. This value is unchanged since there was no synchronization.
last_pay_date	05/14/2012. This date changes to the check pay date.
next_pay_date	05/14/2012. 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 05/12/2012 23:59:00PM, the pmtRecurringPayment job runs again and finds bills with a doc date between 04/10/2012 11:59:00PM and 05/12/2012 23:59:00PM. No bills exist, and the last process time is updated to 05/12/2012 23:59:00PM. Everything else remains the same.

9 On 05/13/2012, 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	03/10/2012	100.01	04/15/2012
acct1111	bill2	04/10/2012	50.00	04/25/2012
acct1111	bill3	04/10/201	100.00	05/15/2012
acct1111	bill4	05/13/2012	80.00	06/15/2012

10 On 05/13/2012 23:59:00PM, the pmtRecurringPayment job runs again, and retrieves bills with dates between 05/12/2012 23:59:00PM and 05/13/2012 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	act1111
bill_scheduled	N. The bill has not been paid.
status	Inactive, because the next pay date is beyond the effective period.
last_process_time	05/15/2012 23:59:00PM. The value changes to the job run time.
last_pay_date	05/14/2012. This value is unchanged.
next_pay_date	06/14/2012. This value is one day before the due date, 06/15/2012.
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 04/09/2012, 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 04/10/2012, 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	04/10/2012
last_pay_date	01/01/1970
next_pay_date	4/30/2012. This is the first available pay date after 04/10/2012. 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. 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	03/10/2012	100.01	04/15/2012
acct1111	bill2	04/10/2012	50.00	04/25/2012
acct1111	bill3	04/10/2012	100.00	05/15/2012

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 04/10/2012 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	acct111
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	04/10/2012 23:59:00PM. This column changed to the job run time.
last_pay_date	01/01/1970. This value is unchanged.
next_pay_date	04/30/2012. There is no 31st of April.
bill_id	bill3
curr_num_payments	0

3 On 04/27/2012, three days before the next pay date, the pmtRecurringPayment job runs again. There is no synchronization, since the value of 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 04/30/2012. 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	04/10/2012 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	04/10/2012 23:59:00PM. This value is unchanged since there has been no synchronization.

4 Step 1, Step 2, and Step 3 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 04/30/2012 to 05/31/2012 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 04/09/2012, 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 04/10/2012 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	04/10/2012
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	03/10/2012	04/15/2012
acct1111	bill2	04/10/2012	04/25/2012
acct1111	bill3	04/10/2012	05/15/2012

Amount due is not required for this case.

2 The pmtRecurringPayment job runs on 04/10/2012 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	04/10/2012 23:59:00PM. This value changes to the job run time.
last_pay_date	01/01/1970. The value is unchanged.
next_pay_date	05/14/2012. This value is one day before the due date, 05/15/2012.
bill_id	bill3
curr_num_payment	0

3 On 05/11/2012, 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 05/14/2012. 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	04/10/2012 23:59:00PM. This value is unchanged since there was no synchronization.
last_pay_date	05/11/2012. This value changes to the value in the next_pay_date column.
next_pay_date	05/11/2012. 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 Step 1, Step 2, and Step 3 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 04/09/2012, 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 04/10/2012 and ends at 06/10/2012. 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	04/10/2012
last_pay_date	01/01/1970
next_pay_date	05/01/2012
bill_id	Null
payment_id	06/10/2012
curr_num_payments	0. No payment has been made yet.

2 On 04/28/2012, 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 05/01/2012.

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	04/10/2012. This value is unchanged since there was no synchronization.
last_pay_date	05/01/2012. This value changes to the next pay date.
next_pay_date	06/01/2012. 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. This chapter includes the following topics:

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 71 describes the columns on the Job Report for Email Notification.

Table 71. 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.

Table 71. Columns on the Email Notification Job Report

Column	Description
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
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

Viewing Job Reports

Follow these steps to view a job report.

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, then 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 72.

Table 72. 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.

Viewing Production Log Messages

Follow these steps to view production log messages.

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, then 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, then 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 169.

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. This property is the 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. This property is 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.

11 Administering the Database

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

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, or 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 must add a unique key constraint on the bill_id field of the check_payments table using the set_unique_bill_id.sql script. 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, then the customer receives 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, then the payment fails and an email notification message is 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 the customer 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, then 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.

To set the unique constraint

- Run the set_unique_bill_id.sql script, located in the following directory. In the path, *EDX_HOME* is the directory where you installed Oracle Self-Service E-Billing.
  - **UNIX.** *EDX_HOME/db/DB_NAME*
  - **Windows.** *EDX_HOME\db\DB_NAME*

UI Actions and Database Changes

Table 73 lists user actions and describes their impact on the Payment module database.

Table 73. 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, in the format yyyyMMddHHmm.
Run pmtConfirmEnroll	Changes account_status to bad_active if returned or to active if there is no return after three days. Updates notify_status to N.
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 canceled 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 checks with Paid, Returned, Failed and Canceled status.

Payment Table Sizing

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

Table 74 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 74. 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 74. 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, but not all customers will use reminders. Each reminder creates one row in the table.
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 of 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, then 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, then 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.

Migrating B2C Users in Batch Mode

Oracle Self-Service E-Billing provides a SQL command script that lets you migrate B2C users from your billing system to Oracle Self-Service E-Billing in batch mode.

You must create a UTF8 input file with the required user information to use this script. For details about the input file, see *Implementation Guide for Oracle Self-Service E-Billing*.

To migrate B2C users to Oracle Self-Service E-Billing in batch mode

1. Using SQL*Plus, log in to the Oracle Self-Service E-Billing instance as the OLTP schema owner.
2. Upload the input file to database server.
3. Run the following commands using SQL*Plus:

```
SQL> exec EDX_PKG_MIG_END_USER.main('Input_File_Directory',
'Migrate_End_Users_Filename.dat');

SQL> exit;
```

   where:

   - *Input_File_Directory* is the directory that contains the input file.
   - *Migrate_End_Users_Filename.dat* is the name of the input file.

Deleting B2C Users in Batch Mode

Oracle Self-Service E-Billing provides a SQL command script that lets you delete B2C users along with their associated billing accounts and payment configurations.

You must create a UTF8 input file with the required user information to use this script. For details about the input file, see *Implementation Guide for Oracle Self-Service E-Billing*.

For each user IDs defined in the ASCII file, the delete B2C users script soft deletes the users from the Oracle Self-Service E-Billing database. In addition, the script also performs the following tasks:

- Disassociates the billing accounts from each user.
- Cancels all scheduled and recurring payments each user has configured.
- Removes all payment accounts associated with each user.

To delete B2C users and associated activity

1. Using SQL*Plus, log in to the Oracle Self-Service E-Billing instance as the OLTP schema owner.
2. Upload the input file to database server.

3 Run the following commands using SQL*Plus:

```
SQL>set serveroutput on size 100000

SQL>exec edx_pkg_del_user_act.main('Input_File_Directory',
'Delete_B2C_Users_Filename.dat');

SQL>exit;
```

where:

- *Input_File_Directory* is the directory that contains the input file.
- *Delete_B2C_Users_Filename.dat* is the name of the input file.

Records that are invalid or generate an error display on screen.

Canceling Payments for B2C Billing Accounts

Oracle Self-Service E-Billing provides a SQL command script that lets you cancel, or delete, all payment types configured for one or more billing accounts. The script reads the billing account numbers defined in the UTF8 input file and cancels the following payment types configured for this account.

- **ACH.** One-time, scheduled, and recurring payments.
- **Credit Card.** Scheduled and recurring payments.

For details about the input file, see *Implementation Guide for Oracle Self-Service E-Billing*.

To cancel payments for B2C billing accounts

- Using SQL*Plus, log in to the Oracle Self-Service E-Billing instance as the OLTP schema owner.
- Upload the input file to the database server.
- Run the following commands using SQL*Plus:

```
SQL>set serveroutput on size 100000

SQL> exec edx_pkg_cxl_pymnt.main('Input_File_Directory',
'Cancel_Payments_Filename.dat')

SQL>exit;
```

where:

- *Input_File_Directory* is the directory where the input file is located.
- *Cancel_Payments_Filename.dat* is the name of the input file.

Records that cannot process successfully are displayed in a list on the screen.

About Purging Data

Oracle Self-Service E-Billing provides a script for purging data from the databases:

- **UNIX.** *EDX_HOME*/bin/purge_data.sh
- **Windows.** *EDX_HOME*\bin\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:

- "Viewing Help for Running the Purge Script" on page 156
- "Purging Payment Account and Transactional Data" on page 157
- "Purging Hierarchies and Hierarchy Assignments" on page 162
- "Purging User Data" on page 165
- "Purging Statement and Invoice Fact Data" on page 167
- "Purging Validation Codes" on page 168
- "Purging Administrator Activity" on page 169
- "Purging Locked Administrators in Command Center" on page 169
- "Process of Purging Sample Data" on page 171

You can also purge all related transactional data in batch mode. For information about purging data in batch mode, see "Purging Data in Batch Mode" on page 170.

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 171.

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 shell script as follows:
  - **UNIX.** ./purge_data.sh -help
  - **Windows.** purge_data.bat -help

  The script is located in the following directory. In the path, *EDX_HOME* is the directory where you installed Oracle Self-Service E-Billing.
  - **UNIX.** *EDX_HOME*/bin

- **Windows.** *EDX_HOME*\bin

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 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

1. Go to the following directory:
   - **UNIX.** *EDX_HOME*/bin
   - **Windows.** *EDX_HOME*\bin
2. Run the following command:

- **UNIX.** `./purge_data.sh -connstr` *`OLTP schema user_name/OLTP schema password@EBILL TNS name Input`*
- **Windows.** `purge_data.bat -connstr` *`OLTP schema user_name/OLTP schema password@EBILL TNS name Input`*

where:

- *OLTP schema user_name* is the name of the OLTP schema user.
- *OLTP schema password* is the password of the OLTP schema user.
- *EBILL TNS name* is the TNS name for the Oracle Self-Service E-Billing 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 users

Purging payment accounts removes data from the PAYMENT_ACCOUNTS OLTP table.

Complete the following task to purge payment accounts.

To purge payment accounts

1. Go to the following directory:
   - **UNIX.** *`EDX_HOME`*`/bin`
   - **Windows.** *`EDX_HOME`*`\bin`
2. Run the following command:

- **UNIX.** ./purge_data.sh -connstr *OLTP schema user_name/OLTP schema password@EBILL TNS name Input*
- **Windows.** purge_data.bat -connstr *OLTP schema user_name/OLTP schema password@EBILL 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 171.

To purge payment history

1 Go to the following directory:

- **UNIX.** *EDX_HOME*/bin
- **Windows.** *EDX_HOME*\bin

2 Run the following command:

- **UNIX.** `./purge_data.sh -connstr` *`OLTP schema user_name/OLTP schema password@EBILL TNS name Input`*
- **Windows.** `purge_data.bat -connstr` *`OLTP schema user_name/OLTP schema password@EBILL 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 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

1 Go to the following directory:

- **UNIX.** *EDX_HOME*`/bin`
- **Windows.** *EDX_HOME*`\bin`

2 Run the following command:

- **UNIX.** ./purge_data.sh -connstr *OLTP schema user_name/OLTP schema password@EBILL TNS name Input*
- **Windows.** purge_data.bat -connstr *OLTP schema user_name/OLTP schema password@EBILL 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 users

Purging payment invoices removes data from the PAYMENT_INVOICES table.

Complete the following task to purge payment invoices

To purge payment invoices

1. Go to the following directory:
   - **UNIX.** *EDX_HOME*/bin
   - **Windows.** *EDX_HOME*\bin
2. Run the following command:
   - **UNIX.** ./purge_data.sh -connstr *OLTP schema user_name/OLTP schema password@EBILL TNS name Input*
   - **Windows.** purge_data.bat -connstr *OLTP schema user_name/OLTP schema password@EBILL 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 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

1. Go to the following directory:
   - **UNIX.** *EDX_HOME*`/bin`
   - **Windows.** *EDX_HOME*`\bin`
2. Run the following command:
   - **UNIX.** `./purge_data.sh -connstr` *OLTP schema user_name/OLTP schema password@EBILL TNS name Input*
   - **Windows.** `purge_data.bat -connstr` *OLTP schema user_name/OLTP schema password@EBILL 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 a screen. Information removed from the OLAP hierarchy tables does not display a message because dbms_output does not work for 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

1. Go to the following directory:
   - **UNIX.** *EDX_HOME*/bin
   - **Windows.** *EDX_HOME*\bin
2. Run the following command:
   - **UNIX.** ./purge_data.sh -connstr *OLTP schema user_name*/*OLTP schema password*@*EBILL TNS name* -hierarchy billing -company "*Company ID*"
   - **Windows.** purge_data.bat -connstr *OLTP schema user_name*/*OLTP schema password*@*EBILL 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

1. Go to the following directory:
   - **UNIX.** *EDX_HOME*/bin
   - **Windows.** *EDX_HOME*\bin
2. Run the following command:
   - **UNIX.** ./purge_data.sh -connstr *OLTP schema user_name/OLTP schema password@EBILL TNS name* -hierarchy business -company "*Company ID*"
   - **Windows.** purge_data.bat -connstr *OLTP schema user_name/OLTP schema password@EBILL 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

1. Go to the following directory:
   - **UNIX.** *EDX_HOME*/bin

- **Windows.** *EDX_HOME*\bin

2 Run the following command:

- **UNIX.** ./purge_data.sh -connstr *OLTP schema user_name/OLTP schema password@EBILL TNS name* -hierarchy assign -user *User ID*
- **Windows.** purge_data.bat -connstr *OLTP schema user_name/OLTP schema password@EBILL 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

1 Go to the following directory:

- **UNIX.** *EDX_HOME*/bin
- **Windows.** *EDX_HOME*\bin

2 Run the following command:

- **UNIX.** ./purge_data.sh -connstr *OLTP schema user_name/OLTP schema password@EBILL TNS name* -hierarchy service -company "*Company ID*"
- **Windows.** purge_data.bat -connstr *OLTP schema user_name/OLTP schema password@EBILL 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 users
- Inactive users for a specific company ID

- Inactive Consumer Edition 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_ATTRIBSF
- 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_UMF_EMAIL_VALIDATION
- 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

1. Go to the following directory:
   - **UNIX.** *EDX_HOME*/bin
   - **Windows.** *EDX_HOME*\bin
2. Run the following command:
   - **UNIX.** ./purge_data.sh -connstr *OLTP schema user_name/OLTP schema password@EBILL TNS name Input*
   - **Windows.** purge_data.bat -connstr O*LTP schema user_name/OLTP schema password@EBILL 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 users
- All Consumer Edition users for a specific period

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

- EDX_RPT_STATEMENT_FACT
- EDX_RPT_STATEMENT_CONTENT
- EDX_RPT_STATEMENT_PAYMENT_FACT
- EDX_RPT_STATEMENT_ADJUST_FACT
- EDX_RPT_CONSUM_FACT
- EDX_RPT_SERVICE_MISC_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_TOP_100_LONG_CALL
- EDX_RPT_AVG_BILLED_SUMM (Consumer Edition only)

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

1 Go to the following directory:

- **UNIX.** *EDX_HOME*/bin
- **Windows.** *EDX_HOME*\bin

2 Run the following command:

- **UNIX.** ./purge_data.sh -connstr *OLAP schema user_name/OLAP schema password@EBILL TNS name Input*
- **Windows.** purge_data.bat -connstr *OLAP schema user_name/OLAP schema password@EBILL 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 user enrolls.

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

1 Go to the following directory:

- **UNIX.** *EDX_HOME*/bin
- **Windows.** *EDX_HOME*\bin

2 Run the following command:

- **UNIX.** ./purge_data.sh -connstr *OLTP schema user_name/OLTP schema password@EBILL TNS name* -validationcode -keepdays *Number of days*

- **Windows.** purge_data.bat -connstr *OLTP schema user_name/OLTP schema password@EBILL 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

1 Go to the following directory:

- **UNIX.** *EDX_HOME*/bin
- **Windows.** *EDX_HOME*\bin

2 Run the following command:

- **UNIX.** ./purge_data.sh -connstr OLTP schema *user_name*/OLTP schema password@EBILL TNS name *Input*
- **Windows.** purge_data.bat -connstr *OLTP schema user_name/OLTP schema password@EBILL 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, or 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, or purge data, for administrators

1 Go to the following directory:

- **UNIX.** *EDX_HOME*/bin
- **Windows.** *EDX_HOME*\bin

2 Run the following command:

- **UNIX.** ./purge_data.sh -connstr *OLTP schema user_name/OLTP schema password@EBILL TNS name Input*
- **Windows.** purge_data.bat -connstr *OLTP schema user_name/OLTP schema password@EBILL 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 Mode

You can purge all related transactional data in batch mode for:

- A specific company ID
- A specific user ID
- All Consumer Edition users

NOTE: Payment history data is excluded, as some clients have restrictions on deleting any payment for a particular number of months. The scheduled payments will be canceled 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 (canceled)
- 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 mode.

This task is a step in "Process of Purging Sample Data" on page 171.

To purge data in batch mode

1. Go to the following directory:
   - **UNIX.** *EDX_HOME*/bin
   - **Windows.** *EDX_HOME*\bin
2. Run the following command:
   - **UNIX.** ./purge_data.sh -connstr *OLTP schema user_name/OLTP schema password@EBILL TNS name Input*
   - **Windows.** purge_data.bat -connstr *OLTP schema user_name/OLTP schema password@EBILL TNS name Input*

   where *Input* is one of the following:
   - Purge data in batch mode for a specific company ID: -company "*Company ID*"
   - Purge data in batch mode for a specific user ID: -user *User ID*
   - Purge data in batch mode for all B2C users: -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 Mode" on page 170.
2. "Purging Payment History" on page 159.

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.

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 was purged in the tables because the output function does not work through the database link. For details about the information removed, see the EDX_PURGE_LOG file.

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:

- For Oracle Database 11*g* only, the ojdbc5.jar file, found on your database server in the $ORACLE_HOME\jdbc\lib directory.
- For Oracle Database 10*g* only, the ojdbc14.jar file, found on your database server in the $ORACLE_HOME\jdbc\lib directory.
- For all versions of Oracle Database, 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, then the following message appears in the log: *Master Key Update process is completed*.

b Check the MasterKeyStore.properties file, located in either the AES128 or blowfish subdirectory under the following directory:

- **UNIX.** *EDX_HOME*/keystore
- **Windows.** *EDX_HOME*\keystore

The script generates a new master key in the MasterKeyStore.properties 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*.

c Distribute the new MasterKeyStore.properties file to all application servers to guarantee master key consistency.

d Verify that the following data has been updated:

- The SECURE_SUBKEY column in the EDX_SECURE_SUBKEY table
- The Code in the EDX_UMF_SEC_VALIDATIONCODE table
- The KEY_UPDATE_FLAG column in the EDX_SECURE_SUBKEY table will be NULL

10 If the master key update script ran successfully, then restore the original persistence.xma.xml file to the following directory and restart application server, and log in with your ID and password:

- **UNIX.** *EDX_HOME*/xma/config/modules
- **Windows.***EDX_HOME*\xma\config\modules

11 If the master key update script did not run successfully, then follow these steps to troubleshoot the problem:

a If the script did not generate the new master key in the MasterKeyStore.properties file, then 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, then 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 located in the either the AES128 or blowfish subdirectory under the following directory with the backup file:

- **UNIX.** *EDX_HOME*/keystore
- **Windows.** *EDX_HOME*\keystore

If the column contains any value other than NULL, then you can run the script again.

c If it is not possible to run the master key update script successfully, then 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) processes in Oracle Workflow Manager and the Design Center in Oracle Warehouse Builder. This chapter includes the following topics:

Loading Billed Data

You must load your billing data into Oracle Self-Service E-Billing, using the EBILL_DATALOAD ETL loader process. The EBILL_DATALOAD job loads the ETL data into the OLAP database, then by default automatically publishes the data load into the OLAP production tables and proceeds to update the OLTP database using the LOAD_PRODUCTION process.

You run the EBILL_DATALOAD process 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.

You can optionally run the EBILL_DATALOAD job using the manual publish option. The manual option keeps the data in the OLAP work-in-process (WIP) tables and does not publish the data to production. If you use the manual option, then you must verify the data and run either the Reject_Group_Bill_File or Reject_Master_Bill_File job to reject the data, or run the LOAD_PRODUCTION file to accept the data and complete the ETL publishing process.

CAUTION: If you are running a catch-up load, then you must load past bill files sequentially and check log files to verify that each load is successful before proceeding to the next period. Also check that multiple files in the same billing period have loaded successfully. If a bill file does not load successfully, then that billing period data might not appear on analytic reports because the Hierarchy Node table might not load correctly.

You also have the option to perform a validation on the billing file to be loaded. This option validates the company-to-group key or account-to-group key for the file. By default, the job does not perform a validation.

CAUTION: If you run the EBILL_DATALOAD job using the validation option, then you might experience a degradation in performance.

If you load unbilled data into Oracle Self-Service E-Billing, then the EBILL_DATALOAD job also removes unbilled data records. For more information about unbilled data, see "Loading Unbilled Data" on page 183.

To run the EBILL_DATALOAD process

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 EBILL_DATALOAD node and click Start.
5. Specify the following input parameters.

Parameter	Description
IN_01_FILE_NAME_PATTERN	The file name pattern for identifying input file names.
IN_02_INPUT_PATH	The full directory path where the billing data input file to be loaded to Oracle Self-Service E-Billing is located.
IN_03_OUTPUT_PATH	The full directory path where you want Oracle Self-Service E-Billing to place the input file for processing.
IN_04_REJECT_PATH	The full directory path where you want Oracle Self-Service E-Billing to place an input file that the EBILL_DATALOAD job rejects.
IN_05_PUBLISH	Specify whether the file will be automatically published to production. To automatically publish the load to production, specify A, which is the default. To use the manual publish option, specify M.

Parameter	Description
IN_06_REGULAR_LOAD	Specify whether the ETL-Bill file load is for the regular, or current, billing period. If the load is for the regular, or current, billing period, then specify True. The default is True. If this is a catch-up load for a past billing period, then specify False. Catch-up loads are used in the Business Edition of Oracle Self-Service E-Billing only.
IN_07_DATA_AUDIT_FLAG	Specify whether to run data auditing. To audit the data, specify True. To bypass data auditing, specify False. The default is False.
IN_08_FROM_ADDRESS	The email address of the address to appear in the from field on email notifications sent by the job.
IN_09_TO_ADDRESS	The email address to send any notifications sent by the job, such as admin@example.com.
IN_10_CC_ADDRESS	An email address to copy on any notifications sent by the job, such as admin2@example.com.
IN_11_SMTP_PORT	The number of the SMTP port to use for sending email notifications.
IN_12_SMTP_SERVER	The name of the SMTP server to use for sending email notifications.
IN_13_VALIDATE_PARTN_KEY	Specify whether you want to validate the associations between the company or account data and group keys. To perform key validation, specify Y. To bypass key validation, specify N. The default is N.

Parameter	Description
IN_14_AUTO_REJECT	Specify whether to automatically reject the current bill if the EBILL_DATALOAD fails and clear the data from the associated staging tables. If the load fails, then the ETL log will show the discrepancy between the number of records inserted in the staging table and production fact tables. If a failure occurs when using this option, then you will no longer be able to reference the data in the staging tables. To automatically reject the load, specify Y. If you specify N to suppress this feature, then if a failure occurs you can reference the staging table data. After successfully determining the cause of the failure, you must manually reject the ETL_Bill file before processing your next ETL_Bill file.
IN_15_PROVISIONING_INUSE	Specifies whether the application is using provisioning, or loading company, account, and service data using Web Services. If the value is Y, the EBILL_DATALOAD job skips the processes that load data into company, account, and service dimension tables. If you specify N, the EBILL_DATALOAD job loads the company, account, and service data from the billing load file. The default is N.

6 A dialog box might appear stating that the object must be deployed before execution. If this dialog appears, then click OK.

Rejecting a Group Bill File

You can use the Reject_Group_Bill_File process to perform one of the following:

- Reject a group bill data file and remove loaded data
- Reject the master DIM file only
- Reject one or more statements

You might want to reject a group bill data file and remove loaded data in the following circumstances:

- You want to remove data from the production tables after the load is published.
- The EBILL_DATALOAD job failed and you want to reject the partial load.
- You ran the EBILL_DATALOAD job using the manual publishing option and want to reject the unpublished load.

NOTE: If you divided your data load file into multiple files and want to reject the entire master bill file, then see “Launch the Control Center from the Tools menu.” on page 181.

To reject a group bill file load using the Reject_Group_Bill_File 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, REJECT, and REJECT_GROUP_BILL_FILE nodes.
4. Right click the REJECT_GROUP_BILL_FILE node and click Start.
5. Specify or verify the following input parameters, then click OK.

Parameter	Description
P_01_FILE_NAME	Enter the full name of the Oracle Self-Service E-Billing data file, for example, EBILLING_B2B-DATA-FILE-20121210.DAT. This can be a group or master DIM file name.
P_02_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.

Rejecting a Master Bill File

Use the Reject_Master_Bill_File job to manually reject the master bill file if you have divided your data load file into multiple files by group assignment. This job rejects all group bill files associated with the master file.

The Reject_Master_Bill_File job performs the following functions on the master and individual group files:

- Validates the files.
- Determines the associated group file names associated with each file.
- Determine whether the data files have been published.
- Performs the reject process for each group file depending on the published state.

NOTE: If you did not divide your data load file into multiple files, then use the Reject_Group_Bill_File job, described in "To reject a group bill file load using the Reject_Group_Bill_File job" on page 181.

To reject a master bill file load using the Reject_Master_Bill_File 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, REJECT, and REJECT_MASTER_BILL_FILE nodes.
4. Right click the REJECT_MASTER_BILL_FILE node and click Start.
5. Specify the name of the master bill file for the P_01_MASTER_FILENAME parameter.

Running the LOAD_PRODUCTION Process Manually

The LOAD_PRODUCTION process runs automatically as part of the EBILL_DATALOAD. However, if you run the EBILL_DATALOAD job with the IN_05_PUBLISH parameter set to M for the manual publish option, then you must manually run the LOAD_PRODUCTION job to complete the ETL process and update the OLTP database.

CAUTION: If you are running a catch-up load, then you must load past bill files sequentially and check log files to verify that each load is successful before proceeding to the next period. Also check that multiple files in the same billing period have loaded successfully. If a bill file does not load successfully, then that billing period data might not appear on analytic reports because the Hierarchy Node table might not load correctly.

The LOAD_PRODUCTION job moves the data into the OLAP production tables and 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 subloaders as described in Table 75.

Table 75. Subloader Actions

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.
Company loader	Inserts rows into the company table, edx_bsl_cmf_company, selecting distinct company ID's.
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.

To run the LOAD_PRODUCTION process

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. Expand the EBILLING_ETL_LOCATION, PROD_LD, and LOAD_PRODUCTION nodes.

4 Right click the LOAD_PRODUCTION node and click Start.

5 Specify the following input parameters.

Parameter	Description
IN_06_REGULAR_LOAD	Specify whether the ETL-Bill file load is for the regular, or current, billing period. If the load is for the regular billing period, then specify True. The default is True. If this is a catch-up load for a past billing period, then specify False. Catch-up loads are used in the Business Edition of Oracle Self-Service E-Billing only.
IN_07_FILENAME	The full name of the data file, for example, EBILLING_B2B-DATA-FILE-20110710.DAT
IN_08_FROM_ADDRESS	The email address of the address to appear in the from field on email notifications sent by the job.
IN_09_TO_ADDRESS	The email address to send any notifications sent by the job, such as admin@example.com.
IN_10_CC_address	An email address to copy on any notifications sent by the job, such as admin2@example.com.
IN_11_SMTP_PORT	Port number. The default value is 25.
IN_12_SMTP_SERVER	The name of the SMTP server to use for sending email notifications.

Loading Unbilled Data

You can optionally load unbilled data into Oracle Self-Service E-Billing, using the EBILL_UBDATALOAD ETL process in Oracle Workflow Builder. The EBILL_UBDATALOAD job loads the unbilled ETL data into the OLAP staging tables, then automatically publishes the data load into the OLAP unbilled fact table.

Before running the EBILL_UBDATALOAD process, the unbilled company, account, service agreement, and hierarchy (for B2B) data must exist in Oracle Self-Service E-Billing. For information on how to provision the required data using Web Services, see *Web Services Reference for Oracle Self-Service E-Billing*.

The EBILL_UBDATALOAD process performs a checksum error evaluation and logs missing or incorrect data.

You use the IN_11_AUTO_REJECT parameter to specify whether to automatically or manually reject an unbilled data file that fails to load correctly:

- **Automatic rejection of an unbilled file**. Using automatic rejection directs the EBILL_UBDATALOAD job to automatically clear the data and staging tables for an unbilled file that fails to load properly.

- **Manual rejection of an unbilled file**. Using manual rejection requires that you manually run a rejection process for the failed unbilled file. For details on manually rejecting an unbilled file that failed to load properly, see "Rejecting an Unbilled or Prepaid Data File" on page 189.

Successfully loaded unbilled data is eventually removed by the EBILL_DATALOAD job. When loading a billed file, the EBILL_DATALOAD job also removes unbilled records for the service number account number combination being loaded. To be removed, the unbilled data must also be less than or equal to the bill cycle end date of the incoming billed file. (If the bill cycle end date is not present, then statement date is used.) If, for any reason, old unbilled data records do not get removed by the EBILL_DATALOAD job, you can purge the data records. For details, see "Purging Unbilled or Prepaid Data Records" on page 189.

To run the EBILL_UBDATALOAD process

1. Verify that the required company, account, service agreement, and hierarchy (for B2B) data for the unbilled data file have been provisioned in Oracle Self-Service E-Billing.
2. Log in to the Oracle Workflow Builder Design Center as the repository owner.
3. Launch the Control Center from the Tools menu.
4. Expand the EBILLING_ETL_LOCATION, ETL_PF_MODULE, and ETL_SUP nodes.
5. Right click the EBILL_UBDATALOAD node and click Start.
6. Specify the following input parameters.

Parameter	Description
IN_01_FILE_NAME_PATTERN	The file name pattern for identifying unbilled input file names.
IN_02_INPUT_PATH	The full directory path where the data input file to be loaded to Oracle Self-Service E-Billing is located.
IN_03_OUTPUT_PATH	The full directory path where you want Oracle Self-Service E-Billing to place the input file for processing.
IN_04_REJECT_PATH	The full directory path where you want Oracle Self-Service E-Billing to place an input file that the EBILL_UBDATALOAD job rejects.
IN_05_DATA_AUDIT_FLAG	Specify whether to run data auditing. To audit the data, specify True. To bypass data auditing, specify False. The default is False.
IN_06_TO_ADDRESS	The email address to send any notifications sent by the job, such as admin@example.com.
IN_07_CC_ADDRESS	An email address to copy on any notifications sent by the job, such as admin2@example.com.
IN_08_FROM_ADDRESS	The email address of the address to appear in the from field on email notifications sent by the job.

Parameter	Description
IN_09_SMTP_PORT	The number of the SMTP port to use for sending email notifications.
IN_10_SMTP_SERVER	The name of the SMTP server to use for sending email notifications.
IN_11_AUTO_REJECT	Specify whether to automatically reject the current unbilled data if the EBILL_UBDATALOAD fails and clear the data from the associated staging tables. If you specify N to suppress this feature, then if a failure occurs you can reference the staging table data and use the EDX_ETL_CHECKSUM_TROUBLESHOOT process. The ETL log will show the discrepancy between the number of records inserted in the staging table and production fact tables. After successfully determining the cause of the failure, you must manually reject the file before processing your next file. If you specify Y to use the auto reject feature, you will not be able to reference the data that was in the staging tables.

7 A dialog box might appear stating that the object must be deployed before execution. If this dialog appears, then click OK.

8 Review the checksum result in the ETL log table, EDX_RPT_ETL_LOG.

For more information about resolving checksum errors, see “Resolving Checksum Errors” on page 187.

Loading Prepaid Data

You can optionally load prepaid billing data into Oracle Self-Service E-Billing, using the EBILL_PPDATALOAD ETL process in Oracle Workflow Builder.

Before loading a prepaid data file, the prepaid account and service agreement numbers must exist in Oracle Self-Service E-Billing. For information on how to provision the required data using Web Services, see *Web Services Reference for Oracle Self-Service E-Billing*.

The EBILL_PPDATALOAD job loads the prepaid ETL data into the OLAP staging tables, then automatically publishes the data load into the OLAP prepaid fact table.

The EBILL_PPDATALOAD process performs a checksum error evaluation and logs missing or incorrect data.

You use the IN_11_AUTO_REJECT parameter to specify whether to automatically or manually reject a prepaid data file that fails to load correctly:

- **Automatic rejection of an unbilled file**. Using automatic rejection directs the EBILL_PPDATALOAD job to automatically clear the data and staging tables for an unbilled file that fails to load properly.
- **Manual rejection of an unbilled file**. Using manual rejection requires that you manually run a rejection process for the failed prepaid file. For details on manually rejecting a prepaid file that failed to load properly, see "Rejecting an Unbilled or Prepaid Data File" on page 189.

It is recommended that you keep not more than three months of prepaid data, as this could result in less efficient reporting analysis on the data. For details on how to purge prepaid data, see "Purging Unbilled or Prepaid Data Records" on page 189.

To run the EBILL_PPDATALOAD process

1. Verify that the required account and service agreement data for the prepay data file has been provisioned in Oracle Self-Service E-Billing.
2. Log in to the Oracle Workflow Builder Design Center as the repository owner.
3. Launch the Control Center from the Tools menu.
4. Expand the EBILLING_ETL_LOCATION, ETL_PF_MODULE, and ETL_SUP nodes.
5. Right click the EBILL_PPDATALOAD node and click Start.
6. Specify the following input parameters.

Parameter	Description
IN_01_FILE_NAME_PATTERN	The file name pattern for identifying prepay input file names.
IN_02_INPUT_PATH	The full directory path where the data input file to be loaded to Oracle Self-Service E-Billing is located.
IN_03_OUTPUT_PATH	The full directory path where you want Oracle Self-Service E-Billing to place the input file for processing.
IN_04_REJECT_PATH	The full directory path where you want Oracle Self-Service E-Billing to place an input file that the EBILL_PPDATALOAD job rejects.
IN_05_DATA_AUDIT_FLAG	Specify whether to run data auditing. To audit the data, specify True. To bypass data auditing, specify False. The default is False.
IN_06_TO_ADDRESS	The email address to send any notifications sent by the job, such as admin@example.com.
IN_07_CC_ADDRESS	An email address to copy on any notifications sent by the job, such as admin2@example.com.
IN_08_FROM_ADDRESS	The email address of the address to appear in the from field on email notifications sent by the job.
IN_09_SMTP_PORT	The number of the SMTP port to use for sending email notifications.

Parameter	Description
IN_10_SMTP_SERVER	The name of the SMTP server to use for sending email notifications.
IN_11_AUTO_REJECT	Specify whether to automatically reject the current unbilled data if the EBILL_PPDATALOAD fails and clear the data from the associated staging tables. If you specify N to suppress this feature, then if a failure occurs you can reference the staging table data and use the EDX_ETL_CHECKSUM_TROUBLESHOOT process. The ETL log will show the discrepancy between the number of records inserted in the staging table and production fact tables. After successfully determining the cause of the failure, you must manually reject the file before processing your next file. If you specify Y to use the auto reject feature, you will not be able to reference the data that was in the staging tables.

7 A dialog box might appear stating that the object must be deployed before execution. If this dialog appears, then click OK.

8 Review the checksum result in the ETL log table, EDX_RPT_ETL_LOG.

For more information about resolving checksum errors, see “Resolving Checksum Errors” on page 187.

Resolving Checksum Errors

The processes for loading unbilled and prepaid data, EBILL_UBDATALOAD and EBILL_PPDATALOAD, perform a checksum error analysis. For each process, the checksum determines whether the number of records loaded into the staging table is the same as the number finally loaded into the production table.

The ETL log table, EDX_RPT_ETL_LOG, shows the checksum result. For prepaid data loads, the checksum error log also provides information about what was missing or incorrect.

You can run the EDX_ETL_CHECKSUM_TROUBLESHOOT process to determine which records caused the checksum errors in the unbilled or prepay data load files. The EDX_ETL_CHECKSUM_TROUBLESHOOT process populates the EDX_RPT_ETL_CHECKSUM_ERRORS table, which you can then use to review the problem records.

You can use the EDX_ETL_CHECKSUM_TROUBLESHOOT process only if the IN_11_AUTO_REJECT process parameter in the unbilled or prepaid load process is set to N.

To determine which records caused the checksum errors

1 Log into SQL*Plus as the schema owner.

2 Run the EDX_ETL_CHECKSUM_TROUBLESHOOT process as follows:

```
set serveroutput on


declare


vresult int;
vtablenamevarchar2(30);
vfilenamevarchar2(255);
begin
  -- Call the procedure
  edx_etl_checksum_troubleshoot(pfilename => vfilename,
                                ptablename => vtablename,
                                presult => vresult);
dbms_output.put_line(vresult);


end;
/
```

where:

- *pfilename* is the name of the unbilled or prepaid data input file that produced the checksum error.
- *ptablename* is the table name to perform the checks against, either the EDX_RPT_PREPAID_DETAIL_FACT or EDX_RPT_UNBILLED_DETAIL_FACT table. You can also specify no value.
- *presult* is the return code: 0 = success; -1 = failure. Check the EDX_RPT_ETL_LOG file in the OLAP database for details about the failure.

3 Review the EDX_RPT_ETL_CHECKSUM_ERRORS table, which was populated by the EDX_ETL_CHECKSUM_TROUBLESHOOT process.

Rejecting an Unbilled or Prepaid Data File

You can manually reject an unbilled or prepaid data file that the EBILL_UBDATALOAD or EBILL_PPDATALOAD process failed to load properly. The REJECT_UNBILLED_FILE and REJECT_PREPAY_FILE processes remove all the data associated with the loaded data file that you specify. You run the data rejection processes in Oracle Workflow Builder.

You can only reject loaded unbilled or prepaid files manually if the IN_11_AUTO_REJECT parameter was set to N when you ran the EBILL_UBDATALOAD or EBILL_PPDATALOAD process.

The can alternately set the unbilled or prepaid file rejection process to occur automatically. If you set the IN_11_AUTO_REJECT to Y, then if a failure occurs during the file load, the process automatically rejects the file. However, automatically rejecting a failed file load also clears staging tables, preventing any analysis from being done on those tables.

To manually reject a loaded unbilled or prepaid ETL file

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 REJECT nodes.
4. Right click the process for the type of loaded data file you want to reject:
   - REJECT_UNBILLED_FILE
   - REJECT_PREPAY_FILE
5. Click Start.
6. Specify the name of the file you want to reject.

Purging Unbilled or Prepaid Data Records

Oracle Self-Service E-Billing provides a procedure for purging unbilled and prepaid data records, if necessary. You can purge all unbilled or prepaid data records older than a date that you specify.

You might need to purge any unbilled data records that did not get removed from the Oracle Self-Service E-Billing database by either the EBILL_DATALOAD or REJECT_UNBILLED_FILE process. Unbilled data records can remain if, for some reason, unbilled records never get billed.

It is recommended that you keep not more than three months of prepaid data, as this could result in less efficient reporting analysis on the data.

To purge loaded unbilled or prepaid data records

1. Log into SQL*Plus as the schema owner.
2. Run the EDX_PURGE_UNBILLED_DATA or the EDX_PURGE_PREPAY_DATA procedure as shown in the following example

```
set serveroutput on


declare
vdate  varchar2(30) := to_date('05/01/2013','mm/dd/rrrr');
vval int;
begin
  EDX_PURGE_UNBILLED_DATE(vdate ,vval);
  dbms_output.put_line(vval);
end;
/
```

where:

- *EDX_PURGE_UNBILLED_DATA* is the purge process. Use EDX_PURGE_UNBILLED_DATA to purge unbilled data and EDX_PURGE_PREPAY_DATA to purge prepaid data.
- *05/01/2013* is the older-than date.

3 A return value of -1 indicates possible errors. Check the edx_rpt_etl_log file for more information.

Database Partitioning Architecture

Oracle Self-Service E-Billing preconfigures some database tables with a default partition and a sub-partition template for storing data.

Oracle Self-Service E-Billing uses group keys to identify partitions. The default partition is assigned group key 0. You must create a group for each partition you want to add, and include the group key in the data input file for each company or account. All of the companies or accounts in a load input file must be associated with a single group key.

If you are not using grouping, then you must specify a group key value of zero. If you do not specify a group key value for a company or account in your data load file, then Oracle Self-Service E-Billing does not assign the default group key value of zero.

The ETL load job, EBILL_DATALOAD, provides an option to validate the group provided for each company or account in the data input file. This helps maintain your company or account data in the correct partition. For the ETL load to validate the group key indicated in your input file, you must create data-group associations in Oracle Self-Service E-Billing. Each time you add companies or accounts, make these assignments before loading data files. For instructions on creating new groups and associating companies or accounts with a group, see “Process of Setting Up New Database Partitions” on page 191.

Partition Naming Convention

Oracle Self-Service E-Billing uses the convention P_*GroupKey*_*PeriodNumber* for partitions that do not have subpartitions

where:

- P indicates the top-level partition.
- *GroupKey* is the group key number.
- *PeriodNumber* is the billing period key from the period table.

For example, P_1_90 refers to top-level partition with group key 1 for billing period 90, such as January.

For partitions with subpartitions, Oracle Self-Service E-Billing uses the naming convention P_*GroupKey*_*PeriodNumber*_*SP*_*ETLKey*, where the additional parameters for the subpartition are as follows:

- SP indicates the subpartition.
- *ETLKey* is the subpartition number, which is also the ETL key.

For example, P_1_91_SP_999 refers to top-level partition with group key 1 for billing period 91, such as February, and subpartition for ETL key 999.

It is also possible to have a table with top-level partition GROUP_KEY and SUB-PARTITION of period key, in that case you would see P_1_SP_90.

Some tables are preconfigured with GROUP_KEY as the top-level partition, and always use P_1 for group key 1, no period partition.

Process of Setting Up New Database Partitions

In Oracle Self-Service E-Billing, you can set up new database partitions.

To set up a new database partition, perform the following tasks:

1. “Creating a New Partition Group” on page 191
2. “Associating Data with a Group” on page 192

Creating a New Partition Group

For each new database partition you want to create in Oracle Self-Service E-Billing, you must create a new group.

This task is a step in “Process of Setting Up New Database Partitions” on page 191.

To create a new partition group

- Using a SQL tool such as Toad, Oracle SQL Developer, or SQL*Plus, execute the OLAP database function fn_Create_Group located in the package pkg_olap_group_mgmt. Specify the following input parameters with the function:
  - **P_GROUP_KEY.** A value for the group key you are creating.
  - **P_TABLESPACE.** The tablespace to be associated with any partitions and subpartitions going forward.
  - **P_START_DATE.** The start date of the group, which determines the start period for partitions and subpartitions that are period-key driven.
  - **P_DESC.** Description for the group being created. This parameter is optional\.

  The function generates one output parameter indicating whether the process was successful. A value of 0 (zero) indicates success. Any other value is an error. For errors, see the OLAP log table, EDX_RPT_ETL_LOG and the OLTP log table, EDX_OLTP_LOADER_LOG.

Associating Data with a Group

You must associate a company (Business Edition) or an account (Consumer Edition) data with a particular group so that the ETL load can assign the data to the defined group and the ETL load validation can work properly. For details about setting the parameter in the EBILL_DATALOAD job that validates the group associations, see "Loading Billed Data" on page 177.

A company or account can be associated with a single group key only. A group key can be associated with multiple companies or accounts, however. Do not combine business (company) and consumer (account) data in the same group.

Whenever you have new companies or accounts, you must add associations for that data before loading the data file.

Oracle Self-Service E-Billing maintains the data-group associations you create in the partition mapping table.

This task is a step in "Process of Setting Up New Database Partitions" on page 191.

Associating a Company with a Partition Group in the Business Edition

In the Business Edition of Oracle Self-Service E-Billing you can associate a company with a partition group.

To associate a company with a partition group in the Business Edition

- Using a SQL tool such as Toad, Oracle SQL Developer, or SQL*Plus, run the MapCompanytoPartition.sql script, located in the following directory:
  - **UNIX.** *EDX_HOME*/applications/db/oracle/olap/etl/owb
  - **Windows.** *EDX_HOME*\applications\db\oracle\olap\etl\owb

  Specify the following input parameters with the function:

- **COMPANY ID.** The company_cd value in the EDX_RPT_COMPANY_DIM table.
- **GROUP KEY.** The group_key value in the EDX_RPT_PARTN_MGMT table.

This script inserts the associations to the EDX_RPT_PARTN_COMP_MAP table. Any errors display on-screen using the Oracle dbms_output.put_line function.

Associating an Account with a Partition Group in the Consumer Edition

In the Consumer Edition of Oracle Self-Service E-Billing you can associate an account with a partition group.

To associate an account with a partition group in the Consumer Edition

- Using a SQL tool such as Toad, Oracle SQL Developer, or SQL*Plus, run the MapAccounttoPartition.sql script, located in the following directory:
  - **UNIX.** *EDX_HOME*/applications/db/oracle/olap/etl/owb
  - **Windows.***EDX_HOME*\applications\db\oracle\olap\etl\owb
- Specify the following input parameters with the function:
  - **BILLER ID.** The biller_id value in the EDX_RPT_ACCOUNT_DIM table.
  - **ACCT NUM.** The account_num value in the EDX_RPT__ACCOUNT_DIM table.
  - **GROUP KEY.** The group_key value in the EDX_RPT_PARTN_MGMT table.

  inserts the associations to the EDX_RPT_PARTN_ACCT_MAP table. Any errors display on-screen using the Oracle dbms_output.put_line function.

Moving Data Between Partitions

You can redistribute data between existing partitions, or groups, such as when a particular partition becomes over-crowded or to provide certain companies with their own dedicated partition.

It is recommended to move data between partitions during off-hours or times of low application usage. While a move function is in process, Oracle Self-Service E-Billing prevents you from running an EBILL_DATALOAD job.

Do not mix line-of-business and billing systems during a move.

Once you complete a successful move, you cannot move a company or accounts back to the original group.

Table 76 shows the Oracle Self-Service E-Billing database tables where data is inserted or updated whenever you move data to a new partition.

Table 76. Database Tables Modified by Moving Data Between Partitions

Schema	Table	Description
OLAP	EDX_RPT_TOP_100_LONG_CALL	Inserts to the work-in-process (WIP) table and then performs Partition Exchange Loading to the production table.
OLAP	EDX_RPT_TOP_100_EXPENSIVE_CALL	Inserts to the WIP table and then performs Partition Exchange Loading to the production table.
OLAP	EDX_RPT_STATEMENT_PAYMENT_FACT	Inserts to the WIP table and then performs Partition Exchange Loading to the production table.
OLAP	EDX_RPT_STATEMENT_FACT	Updates the group key and ETL key with new values.
OLAP	EDX_RPT_STATEMENT_ADJUST_FACT	Inserts to the WIP table and then performs Partition Exchange Loading to the production table.
OLAP	EDX_RPT_SERVICE_USAGE_FACT	Inserts to the WIP table and then performs Partition Exchange Loading to the production table.
OLAP	EDX_RPT_SERVICE_TARIFF_FACT	Inserts to the WIP table and then performs Partition Exchange Loading to the production table.
OLAP	EDX_RPT_SERVICE_PRODUCT_FACT	Inserts to the WIP table and then performs Partition Exchange Loading to the production table.
OLAP	EDX_RPT_SERVICE_MISC_FACT	This table is used in the Utilities application.
OLAP	EDX_RPT_SERVICE_FACT	Inserts to the WIP table and then performs Partition Exchange Loading to the production table.
OLAP	EDX_RPT_SERVICE_DIM	Updates the group key.
OLAP	EDX_RPT_SERVICE_DETAIL_FACT	Inserts to the WIP table and then performs Partition Exchange Loading to the production table.
OLAP	EDX_RPT_SERVICE_CHARGE_FACT	Inserts to the WIP table and then performs Partition Exchange Loading to the production table.
OLAP	EDX_RPT_HIERARCHY_XREF_DIM	Updates the group key.

Table 76. Database Tables Modified by Moving Data Between Partitions

Schema	Table	Description
OLAP	EDX_RPT_HIERARCHY_TREE_DIM	Updates the group key.
OLAP	EDX_RPT_HIERARCHY_NODE_ PERIOD	Inserts to the WIP table and then inserts to the production table.
OLAP	EDX_RPT_ETL_LOAD_FACT	Inserts a new record to log the move by period and new group key.
OLAP	EDX_RPT_CONSUM_FACT	Inserts to the WIP table and then inserts to the production table. This table is used in the Utilities application only.
OLAP	EDX_RPT_AVG_BILLED_SUMM	Inserts to the WIP table and then inserts to the production table. This table is used in the Utilities application only.
OLAP	EDX_RPT_ACCOUNT_WSPACE	Inserts to the WIP table and then inserts to the production table.
OLAP	EDX_RPT_ACCOUNT_FACT	Inserts to the WIP table and then performs Partition Exchange Loading to the production table.
OLAP	EDX_RPT_ACCOUNT_CHARGE_FACT	Inserts to the WIP table and then performs Partition Exchange Loading to the production table.
OLTP	ETL_GROUP_SOURCE	Inserts a new record to log the move by the new ETL key, load datetime.
OLTP	EDX_OMF_SERVICEAGREEMENT	Updates the group key.
OLTP	EDX_HIER_NODE_PERIOD	Insert the group and ETL keys with new values.
OLTP	EDX_HIER_HNODE	Updates the group key.
OLTP	EDX_HIER_HIERARCHY	Updates the group key.

To move data for one company at a time to a new partition group (Business Edition)

1 Verify the following information:

- The B2B data exists and is in a published state.
- The new group exists and there is an entry in the EDX_RPT_PARTN_MGMT table for the partition group.
- The current partition group the company is in has an entry in the mapping table EDX_RPT_PARTN_COMP_MAP.

2 Using a SQL tool such as Toad, Oracle SQL Developer, or SQL*Plus, execute the function fn_GroupMove_Company, located in the pkg_olap_group_mgmt package. Specify the following input parameters with the function:

- **p_new_group_key.** The key of the group you are moving the data to.
- **p_old_group_key.** The key of the group you are moving the data from.
- **p_company_cd.** The code of the company whose data you want to move.

To move data for one or more accounts to a new partition group (Consumer Edition)

1 Verify the following information:

- The B2C data exists and is in a published state.
- The new group exists and there is an entry in the EDX_RPT_PARTN_MGMT table for the group.
- The current groups the accounts are in each have an entry in the mapping table EDX_RPT_PARTN_ACCT_MAP.

2 Create an input file listing the account numbers you want to move. Use the following format for the data file, using pipe delimiters (|). Place the input file in the same folder you configured for the GRP_MOVE_INDIR function in the ETL properties file during installation. For details, see *Installation Guide for Oracle Self-Service E-Billing*.

Field Name	Position	Type
Biller ID	COL1	VARCHAR2(20)
Account Number	COL2	VARCHAR2(255)
From Group Key	COL3	NUMBER(7)
To Group Key	COL4	NUMBER(7)

3 Using a SQL tool such as Toad, Oracle SQL Developer, or SQL*Plus, execute the function fn_GroupMove_Account located in the pkg_olap_group_mgmt package. Specify the following input parameters with the function:

- **p_file_name.** The name of the input file with the account numbers to move.
- **p_in_location.** The directory path where the file to be moved is currently located.
- **p_out_location.** The directory path where the file will be moved to for processing.
- **p_new_group_key.** The new group to move the data to.
- **p_old_group_key.** The group the data is to move from.

Removing an Empty Partition Group Key

If you experience an issue while creating a new partition group, then you can remove that empty group key and any partitions that have been created for that group.

To remove an empty partition group key

- Using a SQL tool such as Toad, Oracle SQL Developer, or SQL*Plus, execute the pr_cleanup_partn function, located in the pkg_olap_group_mgmt package. For the input parameter p_group_key, specify the group key associated with the partition you want to drop.

  If the function does not detect data in the partition, then it drops all the partitions associated with the specified group key. If the function detects data in the partition, then the function fails and logs the reason. The function generates one output parameter, p_status, indicating whether the process was successful. A value of zero indicates that the process was successful. Any other value is an error. For errors, see the OLAP log table, EDX_RPT_ETL_LOG and the OLTP log table, EDX_OLTP_LOADER_LOG.

Rejecting a Data Move Between Partition Groups

If you experience an issue while moving data between partition groups, then you can reject the move.

You might need to reject a move if the move failed, was performed incorrectly, or it was done at an inconvenient time.

You can reject a move for either of the following types of data:

- By Company in the Business Edition
- By Account in the Consumer Edition

When you reject a move, Oracle Self-Service E-Billing reverses all the data that was modified or inserted during the move. Table 76 on page 194 shows a list of tables updated when moving data between partitions.

To reject a move of data between partition groups (Business Edition)

- Using a SQL tool such as Toad, Oracle SQL Developer, or SQL*Plus, call the pr_RejectCompMove function, located in the pkg_olap_group_mgmt package. Specify the following input parameters with the function:
  - **p_new_group_key.** The key of the group you moved the data to.
  - **p_old_group_key.** The key of the group you moved the data from.
  - **p_company_cd.** The code of the company whose data you want to move.

  The function generates one output parameter, p_success, indicating whether the process was successful. A value of zero indicates that the process was successful. Any other value is an error. For errors, see the OLAP log table, EDX_RPT_ETL_LOG and the OLTP log table, EDX_OLTP_LOADER_LOG.

To reject a move of data between partition groups (Consumer Edition)

- Using a SQL tool such as Toad, Oracle SQL Developer, or SQL*Plus, call the function pr_RejectAcctMove, located in the pkg_olap_group_mgmt package. Specify the following input parameters with the function:
  - **p_filename.** The name of the input file containing a list of account numbers used with the move you are rejecting.
  - **p_in_location.** The directory path where the file is currently located.
  - **p_out_location.** The directory path where the file will be moved to for processing.
  - **p_new_group_key.** The key of the group you moved the data to.
  - **p_old_group_key.** The key of the group you moved the data from.

  The function generates one output parameter, p_success, indicating whether the process was successful. A value of zero indicates that the process was successful. Any other value is an error. For errors, see the OLAP log table, EDX_RPT_ETL_LOG and the OLTP log table, EDX_OLTP_LOADER_LOG.

Running the ETL Purge Process

Oracle Self-Service E-Billing provides the ETL PURGE process for removing data from the Oracle Self-Service E-Billing databases for a specified number of months.

To run the PURGE 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 following directory:
   - **UNIX.** `ebilling_etl_location/purge/purge_process`
   - **Windows.** `ebilling_etl_location\purge\purge_process`
4. Click Start to run the job. Specify one of following input parameters:
   - **P_MONTH.** The month name in MON-YYYY format. The PURGE job will purge all data older than this month.
   - **P_MONTHS.** The number of months of statements you want to remain in the database.

A 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. This chapter includes the following topics:

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 145.

Application Error Messages

The letters *APP* prefix all error message IDs that relate to applications.

Table 77 describes the application error messages.

Table 77. 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 78 describes the mailer error messages.

Table 78. 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 79 describes the service merger error messages.

Table 79. 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 80 describes the messages.

Table 80. 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 80. 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 81 describes the service monitoring error messages.

Table 81. 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 82 describes the stored procedure error messages.

Table 82. 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 83 describes the PWC error messages.

Table 83. 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 84 describes the purging log error messages.

Table 84. 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 85 describes the Scheduler error messages.

Table 85. 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 86 describes the scanner error messages.

Table 86. 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 87 describes the shell command error messages.

Table 87. 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 87. 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 88 describes the services versioning error messages.

Table 88. 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, they store 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 89 lists the error messages that can occur when running Payment jobs.

Table 89. 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 89. 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.

Payment Consolidator Error Messages

When the payment consolidator jobs generate error messages, they store them in the payment error log file, com.edocs.payment.log, or in the Logs table.

The letters *PST* prefix all error message IDs that relate to the payment consolidator feature.

Table 90 lists the error messages that can occur when running payment consolidator jobs.

Table 90. Payment Consolidator Error Messages

Message ID	Severity	Message Text
PST0208	Information	Job {2}:{3} Cartridge start at {0} and finish at {1}.
PST0209	Error	Please configure cartridge of {0} for bill summary task
PST0210	Error	Please configure cartridge of {0} for bill summary confirmation task
PST0381	Error	Failed to insert the bill summary for account: {0}, {1} to EDX_PC_BILLSUMMARY_LOG table and exception is:{2}
PST0382	Error	Failed to update the bill summary for account: {0}, {1} to EDX_PC_BILLSUMMARY_LOG table and exception is:{2}
PST0384	Error	Failed to access the enrollment table, exception is {0}
PST0385	Error	Failed to read the enrollment table, exception is {0}
PST0386	Error	Failed to get configuration bean
PST0387	Error	Failed to read the bill summary confirmation file
PST2000	Error	Caught exception: {0}
PST2001	Error	Caught SQLException, error code is {1} and exception is: {0}

Table 90. Payment Consolidator Error Messages

Message ID	Severity	Message Text
PST2003	Error	Caught ProcessException: {0}
PST2004	Error	Caught ConfigureException: {0}
PST2020	Error	Please register a consolidator name: {0}
PST2021	Error	Please register a biller name: {0}
PST2022	Error	Biller sender id is null or empty for biller: {0}
PST2023	Error	The Consolidator sender id is null empty for consolidator: {0}
PST2027	Error	Please configure a biller.
PST2035	Error	File extension is empty.
PST2037	Error	Specified Summary output directory doesn't exist.
PST2039	Error	Invalid From date {0}
PST2040	Error	Invalid To date {0}
PST2041	Error	Fixed date from must be at an earlier time than to.
PST2042	Error	Failed to move file {0} to directory {1}
PST2043	Error	Date selection criteria is {0}, please set the {1} field to NA.
PST2044	Error	Unknown date selection criteria.
PST2045	Error	Date selection criteria is {0}, please clear out the {1} field.
PST2046	Error	Date selection criteria is {0}, please select a number from {1} field.
PST2050	Error	Not set bill summary ACK file directory.
PST2051	Error	Specified bill summary ACK file directory does not exist.
PST2052	Error	Bill Processing Center ID is null or empty for biller: {0}
PST2053	Error	Caught exception when parse bill summary template: {0}
PST2054	Error	Bill summary Accounts list file:{0} does not exist
PST2055	Error	Please configure the file type.
PST2056	Error	Please configure the bill summary cartridge plugin for consolidator {0}
PST2057	Error	Input data file {0} does not exist.
PST2058	Error	File Type {0} does not exist.
PST2059	Error	Log file path {0} does not exist.
PST2060	Error	Please configure cartridge for user account enrollment task.
PST2061	Error	Please configure cartridge for enrollment response task.

Table 90. Payment Consolidator Error Messages

Message ID	Severity	Message Text
PST2062	Error	Response file drop location {0} does not exist.
PST2063	Error	Biller {0} does not exist.
PST2064	Error	Please configure the response file drop location.
PST2065	Error	Consolidator {0} does not exist.
PST2066	Error	Please set up the template file in file type settings.
PST2067	Error	Caught exception while parsing the input template: {0}
PST2068	Error	Data record type is null or empty.
PST2069	Error	Caught TemplateFormatException while parse file: {0}
PST2070	Error	Caught TemplateException while parse file: {0}
PST2071	Error	Caught Exception while parse file: {0}
PST2072	Error	Caught Exception while write the log file: {0}
PST2073	Error	Enrollment plugin {0} of consolidator {1} create failed - {3}
PST2074	Error	Caught Exception while invoking plugin method: {0} - {1}
PST2075	Error	Caught Exception while writing file header: {0}
PST2076	Error	Caught Exception while get enrolled user account list: {0}
PST2077	Error	Caught TemplateException while writing batch header: {0}
PST2078	Error	Caught TemplateException while writing detail record: {0}
PST2079	Error	Caught TemplateException while writing batch trailer: {0}
PST2080	Error	Caught TemplateException while writing file trailer: {0}
PST2081	Error	Output file to directory is empty.
PST2082	Error	Account list file does not exist.
PST2083	Error	Cannot get Billing System ID {0}
PST2084	Error	Plugin implementation for user account enrollment is not set, please change the file type settings for Enrollment Plugin Implementation.
PST2085	Error	Caught Exception while checking whether account {0} exists in the system: {1}
PST2086	Error	Caught Exception while creating billing account {0}: {1}
PST2087	Error	Caught MandatoryMissingException: {0}
PST2088	Error	Date selection criteria is {0}, please set the from date
PST2089	Error	Date selection criteria is {0}, please set the to date

Table 90. Payment Consolidator Error Messages

Message ID	Severity	Message Text
PST2090	Error	Status code in File header record is {0}, status message is {1}, and all records in the file are rejected.
PST4001	Error	Failed to send payment consolidator failure message for job instance:{0}, and the processing detail information is {1}

Index

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X